Université Sidi Mohamed Ben Abdellah

Faculté des Sciences et Techniques Fès

Département Génie Electrique

Mémoire de Projet de Fin d'Etudes

Préparé par

Omar DIOURI

Pour l'obtention du diplôme Ingénieur d'Etat en

SYSTEMES ELECTRONIQUES ET TELECOMMUNICATIONS

Intitulé

Etude comparative d'architecture entre FPGA et DSP pour l'implémentation des Algorithmes de Traitement de signal

Encadré par :

Mr. Mhammed LAHBABI

Mr. Farid ABDI

Soutenu le Mardi 25 Juin 2013, devant le jury composé de:

Pr.Mhammed LAHBABI :Encadrant

Pr. Farid ABDI: Encadrant

Pr. Abdellah MECHAQRANE : Examinateur

Pr. Ali AHAITOUF : Examinateur

Pr. Mouhcine RAZI : Examinateur

ANNEE UNIVERSITAIRE 2012-2013

Notice Biographique :

Auteur : Omar DIOURI

Année : 2013

Etablissement : Faculté des Sciences et Techniques de Fès

Spécialité : Systèmes Electroniques et Télécommunications

($3^{ème}$ année du cycle ingénieur)

Lieu de stage : Laboratoire Signaux Systèmes &Composants, Faculté des Sciences et Techniques de Fès

Cadre du rapport : Stage de Fin d'Etude

Titre du projet de stage :

« **Etude comparative d'architecture entre FPGA et DSP pour l'implémentation des Algorithmes de Traitement de signal »**

Encadrants:

MrMhammed LAHBABI

MrFarid ABDI

Date de début du stage : Le 03 Mars 2013

Date de fin du stage : Le 22 Juin 2013

Résumé

Ce mémoire qui s'inscrit dans le cadre des projets menés par le laboratoire signaux systèmes et composants, visant l'étude des architectures Hardwares des différentes plateformes et essentiellement le DSP (Digital Signal Processor) et le FPGA (Field ProgramambleGateArray) pour des applications de traitement du signal, est intitulé :

« *Etude comparative d'architecture entre FPGA et DSP pour l'implémentation des Algorithmes de Traitement de signal* »

Au regard de ce contexte,je décline mon travail en trois créneaux :

- Une analyse détaillée sur les architectures du FPGA et DSP ainsique le comportement des différents blocs et périphériques constituant chaque plateforme.
- Faire tourner des algorithmes de traitement de signal sur ces plateformes en réalisant du "Profiling".
- Elaborer une analyse temporelle et fonctionnelleconcise, ainsi qu'une estimation de la vitesse maximale, de la consommation en puissance et des performances.

Abstract

This memory is part of the projects undertaken by the laboratory signals systems and components for the study of architecture Hardwares for different platforms and essentially the DSP (Digital Signal Processor) and FPGA (Field Programamble Gate Array) for signal processing applications, is entitled:

« ***Comparative study of architecture for FPGA and DSP implementation of signal processing algorithms***»

Given this context, I assume my work into three segments:

- A detailed analysis of the FPGA and DSP architectures and behavior of the individual blocks and devices constituting each platform.
- Rotating the signal processing algorithms on these platforms by performing the "profiling".
- Develop a concise temporal and functional analysis, and an estimate of the maximum speed, power consumption and performance.

Remerciements

Avant d'entamer la description de mon travail, jeveux exprimer avec beaucoup de plaisir ma profonde gratitude, ma fidèle reconnaissance, mesrespects et mes remerciements les plus sincères à mesencadrants Mr. MhammedLAHBABI et Mr. FaridABDI qui n'ont pas cessé de me prodiguer leurs conseils et leurs recommandations ainsi pour leurs générosité en matière d'encadrement.

Mes remerciements se destinent aussi aux personnels de la Faculté des Sciences et Techniques de Fès pour leurs générosités, leurs gentillesses et leur présence permanente pour me servir.

De même, jetiens à témoigner ma reconnaissance à tous mes enseignants qui ont contribué à l'enrichissement de mes connaissances.

Enfin, mes remerciements sont adressés à tous ceux qui ont collaboré du loin ou de proche ce travail.

Merci à tous…

"Le meilleur moyen pour apprendre à se connaître,
c'est de chercher à comprendre autrui."

André Gide

"Les choses devraient être rendues aussi simples
que possible. Mais pas plus simples."

Albert Einstein

"Etudier sans réfléchir est une occupation vaine ;
réfléchir sans étudier est dangereux."

Confucius

Liste des abréviations :

DMA :Direct Memory Access

DSK :DSP Starter Kit

DSP :Digital Signal Processor.

DSP :Digital signal Processing.

EMIF :External Memory InterFace

FIR :Finite Impulse Response filter

FPGA :Field Programamble Gate Array

IIR : Infinite Impulse Response filter

MAC :Multiply and Accumulate

MFLOPS :Million Floating Operations Per Second.

MIPS :Million Instruction Per Second.

SOC :System-On-Chip

VHDL :Very High Speed Integrated Circuit Hardware Description Langage

VLIW :Very Long Instruction Word

Table des matières

INTRODUCTION GENERALE 11

Chapitre 1 : Contexte du stage 13

Introduction : 14

I. Présentation du projet : 14

1. Cahier des charges : 14

2. Description du projet : 14

II. Description des différentes Architectures Hardwares des FPGA et DSP : 15

1. FPGA : 15

a. Généralités : 15

b. Architecture des FPGA : 17

2. DSP : 19

a. Généralités : 19

b. Architecture interne du DSP : 21

Conclusion : 24

Chapitre 2 : Implémentation du filtre numérique FIR sur FPGA et DSP 25

Introduction : 26

I. Filtre Numérique : 26

1. Définition : 26

2. Filtre FIR 27

3. Pourquoi un filtre FIR 29

II. Implémentation sur FPGA et DSP : 29

1. La carte TSW6011 du FPGA: 29

a. Description de la carte TSW6011 du FPGA: 29

b. Les difficultés de l'implémentation : 31

c. Les solutions proposées : 32

d. Design et simulation du filtre : 32

e. Implémentation sur FPGA : 34

2. La carte C6713 DSK du DSP : 37

a. Description de la carte C6713DSK : 37

b. Design et simulation du filtre : 39

c. Implémentation et test : 41

Conclusion : 42

Chapitre 3 : Comparaison de deux implémentations sur les différentes cibles 43

Introduction : 44

I. Différentes contraintes pour différents domaines 44

1. Performance : 44

2. Consommation électrique : 45

3. Le coût des circuits numériques : 45

II. Interprétation des résultats de l'implémentation : 46

1. Interprétation sur FPGA : 46

a. Vitesse d'exécution : 47

b. Consommation électrique : 47

c. Coût : 47

2. Interprétation sur DSP : 48

a. Benchmarking (Profiling) sans optimisation : 48

b. Benchmarking (Profiling) avec optimisation : 49

c. Vitesse d'exécution : 50

d. Consommation électrique : 51

e. Coût : 51

III. Comparatif : 51

1. Matériel ou Logiciel 51

2. FPGA ou DSP 53

Conclusion 54

Conclusion Générale 55

Annexe 1 : Description des différents blocs sur la carte TSW6011EVM : 56

Annexe 2 : Description du DSP TMS320C6713 : 59

Bibliographie 62

Liste des Figures :

Figure 1 : Flot classique de conception FPGA. 16
Figure 2 : Architecture îlot de calcul....18
Figure 3 : Exemple d'Architecture hiérarchique à quatre niveaux (circuit, tuiles, clusters, éléments configurables)....19
Figure 4 : Différent famille de Texas Instruments. 21
Figure 5 : Architecture interne du DSP.....22
Figure 6 : l'organisation en blocs d'un DSP basé sur une architecture VLIW et Harvard.....23
Figure 7 : Structure générale d'un filtre FIR. 28
Figure 8 : Schéma synoptique de la carte TSW6011 31
Figure 9 : design du filtre sur FDATOOL. 33
Figure 10 : Set quantization parameters....33
Figure 11 : Simulation du filtre sur ModelSim. 34
Figure 12 : Schéma synoptique du projet réalisé sur FPGA. 35
Figure 13 : Schéma synoptique du projet sur la carte.....36
Figure 14 : Simulation sur ModelSim. 37
Figure 15 : Visualisation en temps réel du signal de sortie sur Signal Tap.....37
Figure 16 : la carte C6713DSK. 38
Figure 17 : Schéma synoptique de la carte et du Codec.....39
Figure 18 : Programmation sur Code Composer Studio 3.1....40
Figure 19 : Simulation dans le domaine temporel. 41
Figure 20 : Simulation dans le domaine fréquentiel. 41
Figure 21 : Simulation du filtre sur oscilloscope. 42
Figure 22 : Estimation Fréquence / Complexité pour différents types d'application 44
Figure 23 : Rapport de compilation sur Quratus II.....46
Figure 24 : Estimation de consommation de la puissance sur FPGA.....47
Figure 25 : Equilibre puissance, performance, fonctionnalité et coût.....48
Figure 26 : Configuration du compilateur sans optimisation. 49
Figure 27 : Nombre de cycle 'exécution du boucle 'for' sans optimisation. 49
Figure 28 : Configuration du compilateur avec optimisation. 50
Figure 29 : Nombre de cycle 'exécution du boucle 'for' avec optimisation.....50
Figure 30 : Solutions matérielles pour systèmes DSP 51
Figure 31 : Performance, consommation et flexibilité des différentes architectures matérielles 52

Figure 32 : Consommation et performance par type de processeur 53
Figure 33 : Schéma de principe du programme original. 57
Figure 34 : Les blocs diagramme de TMS320C6713. 59
Figure 35 : Répartition de la mémoire interne L2. 60

INTRODUCTION GENERALE

Apparu dans les années 60-70 en même temps que les premiers calculateurs, le traitement numérique du signal est pendant longtemps resté cantonné à quelques domaines d'application particuliers: radar, industrie pétrolière, et imagerie médicale. Ceci s'explique principalement par le coût prohibitif des ordinateurs à cette époque. L'arrivée des ordinateurs personnels dans les années 80 a contribué au développement de la technologie numérique, occasionnant d'énormes progrès dans la maîtrise des processus de fabrication, l'évolution des architectures matériels et la performance des circuits intégrés. Mais ces efforts ont avant tout portés sur les microprocesseurs généralistes utilisés dans l'industrie des PC. Les circuits spécialisés en traitement du signal ont eux aussi profité de l'évolution technologique mais sont globalement restés « à la traîne » par rapport aux microprocesseurs et microcontrôleurs, qui sont les moteurs de l'industrie de l'époque. Les quelques circuits implémentant des fonctions de traitement numérique du signal sont des blocs câblés non programmables réalisant des traitements simples de type Filtre ou Transformée de Fourier.

La véritable révolution survient au début des années 90 avec l'émergence des applications multimédias, de l'Internet et de la téléphonie mobile. La complexité de ces applications et les fortes contraintes en terme de performance liées à ces domaines obligent les concepteurs de circuits spécialisés pour le traitement numérique du signal à envisager de nouvelles solutions matérielles et logicielles. C'est ainsi qu'on a vu apparaître dans les processeurs de traitement du signal des techniques architecturales modernes tels les jeux d'instructions VLIW (Very Long Instruction Word), les architectures superscalaires et fortement pipelinées, les architectures multiprocesseurs, etc.

Les contraintes liées aux systèmes embarqués associées à la puissance de calculs nécessitée par les applications de traitement du signal conduisent lors de l'implémentation, à l'utilisation de processeurs spécialisés en traitement du signal. Leur architecture est optimisée afin d'exécuter un nombre important d'opérations arithmétiques sur des tableaux de données et donc toutes les combinaisons d'opérations intervenantdans les applications de traitement du signal.

Aujourd'hui, les domaines d'application dans lesquels intervient le traitement de signal sont très nombreux, et la diversité des solutions matérielles d'implémentation elle est aussi très grande.

Ce rapport concerne le domaine traitement de signal. Le 1er chapitre définit les différentes cibles avec leurs architectures et leurs caractéristiques spécifiques au traitement du signal. Dans le 2ème chapitre, le système de traitement de signal qui est mis en œuvre pour la comparaison doit traiter le fonctionnement d'un filtre FIR (Finite Impulse Responsefilter) passe bande dans le domaine de traitement de signal. Le développement FPGA s'effectue à l'aide de MATLAB FDATOOL, avec cet outil, on obtient une description complète du filtre FIR oùles valeurs des coefficients sont converties en VHDL (Very High Speed Integrated Circuit Hardware Description Langage).Le développement de processeur DSP est accompli par le Code Composer Studio qui cible un TMS320C6713 de Texas Instruments. Le dernier chapitre détaille les différentes solutions d'implémentation en précisant leurs performances en termes de vitesse, consommation électrique et coût.

L'objectif principal de cette étude est d'effectuer une comparaison entre les étapes de développement d'une application sur FPGA et ses performances, et entre une solution de traitement du signal s'appuyant sur les caractéristiques d'un processeur DSP.

Chapitre 1 : Contexte du stage

Introduction :

Le 1er chapitre présente une vue générale sur le projet,commençant par une petite description du projet de stage et les différentes démarches suivies pendant cette période, puis décrivantl'architecture des deux cibles et les différents blocs qui les constituent.

I. Présentation du projet :

1. Cahier des charges :

Le but du sujet est de réaliser une étude d'architecture 'Hardware' pour l'intégration d'algorithme de traitement de signal. En effet, on veut comprendre quelle serait la meilleure architecture'Hardware'à utiliser (DSP ou FPGA) pour intégrer des algorithmes très gourmands en opérations de Traitement du Signal (MAC, multiplication en virgule fixe et en virgule flottantes, la vitesse d'exécution ainsi que de la mémoire RAM).

Le travail consisteà faire :

- Une analyse détaillée sur les architectures et le comportement des différents blocs et périphériques qui constituent chaque plateforme.
- Faire tourner des algorithmes de traitement de signal sur ces plateformes en réalisant du "Profiling".
- Faire une analyse temporelle et fonctionnelle ainsi qu'une estimation de la vitesse maximale,de la consommation en puissance et des performances.

2. Description du projet :

Ce projet a pour objectif de faire une étude détaillée sur les architectures Hardwares de DSP et FPGA ensuite de réaliser une comparaison entre les deux plateformes en vued'atteindre une meilleure implémentation tout en gardant d'augmenter les performances et obtenir une meilleure qualité de traitement de signal.

Donc pour faire toutes ces démarches, on a voulu réaliser un filtre numérique FIR d'ordre 40 et de nature passe bandeavec une bande bien déterminée et d'une architecture parallèle symétrique du filtre FIR.

Pour ce faire, nous allonsutiliser le FDATOOL de Matlab pour concevoir notre filtre et entrer les différentes caractéristiques décrites précédemment. Cet outil nous permet de générer le

code VHDL du filtre pour l'implémenter sur un FPGA d'Altera Cyclone III qui est disposée sur lacarte TSW6011 de Texas Instruments et nous permetausside générer tous les coefficients de notre filtre dans un fichier Header en C afin de les utiliser dans un programme du filtre FIR écrit en C qui sera à la suite chargé sur le DSP TMS320C6713.

Cette réalisation nous permet de faire la distinction entre les différentes architectures Hardwares et faire une comparaison en termes de la vitesse d'exécution, la consommation d'énergie et le coût.

II. Description des différentes Architectures Hardwares des FPGA et DSP :

1. FPGA :

a. Généralités :

Un FPGA est un circuit en silicium reprogrammable. À l'aide de blocs logiques préconstruits et de ressources de routage programmables, nous pouvons configurer ce circuit afin de mettre en œuvre des fonctionnalités matérielles personnalisées, sans avoir jamais besoin d'utiliser une maquette ou un fer à souder. En outre, les FPGA sont totalement reconfigurables et peuvent adopter instantanément une nouvelle « personnalité ». Toutefois, la généralisation des outils de conception de haut niveau est en train de modifier les règles de la programmation de FPGA, grâce à des nouvelles technologies permettant de convertir des diagrammes graphiques ou même du code C en circuits matériels numériques[1].

Si les FPGA rencontrent un tel succès dans tous les secteurs, c'est parce qu'ils réunissent le meilleur des ASIC et des systèmes basés processeur. Ils sont plus rentables que les ASIC personnalisés. Les circuits reprogrammables jouissent également de la même souplessed'exécution logicielle qu'un système basé processeur, mais ils ne sont pas limités par le nombre de cœurs de traitement disponibles. Contrairement aux processeurs, les FPGA sont vraiment parallèles par nature, de sorte que plusieurs opérations de traitement différent ne se trouvent pas en concurrence pour l'utilisation des ressources. Chaque tâche de traitement indépendante est affectée à une section spécifique du circuit, et peut donc s'exécuter en toute autonomie sans dépendre aucunement des autres blocs logiques. En

conséquence, vous pouvez accroître le volume de traitement effectué sans que les performances d'une partie de l'application n'en soient affectées pour autant.

En plus de ces ressources, un FPGA est composé d'une mémoire interne de configuration. Chaque point de cette mémoire correspond à la configuration d'un élément d'une des ressources opérationnelles. Cette mémoire est, dans la plupart des cas, réalisée avec une des trois technologies suivantes : ANTIFUSIBLE (la plus ancienne, configurable une seule fois), FLASH (non-volatile) ou SRAM (volatile, la plus utilisée, représente plus de 80 % du marché).

Comme le montre la figure 1, pour réaliser une application avec un FPGA il faut décrire le circuit électronique à réaliser avec un langage de description matérielle comme le VHDL (Very High Speed Integrated Circuit Hardware Description Langage). Puis il faut synthétiser cette description en circuit électronique. Cette étape et les suivantes peuvent se faire avec des logiciels gratuits fournis par le fabricant de circuit. Enfin après une étape de placement et routage qui prend en compte l'architecture du FPGA, un fichier de configuration appelé bitstream est généré. Celui-ci permet de spécifier au FPGA lors de la configuration la position des points de la mémoire de configuration[1].

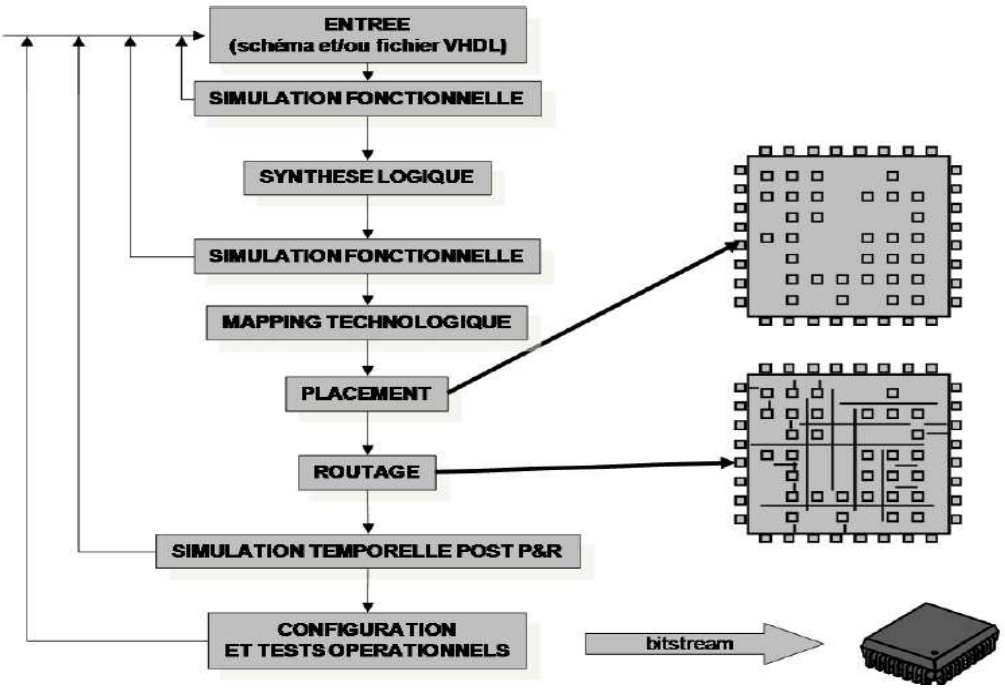

Figure 1 : Flot classique de conception FPGA.

Parmi les principaux fabricants de FPGA dans le monde on peut citer : Xilinx, Altera, Actel (n°1 du marché des FPGA Antifusibles et FLASH), Atmel, QuickLogic, Lattice.

b. Architecture des FPGA :

L'architecture d'un FPGA est principalement décrite par la topologie des ressources de routages et des éléments logiques configurables de base. Il existe deux architectures classiques, l'architecture îlot de calcul (initialement utilisée dans les composants Xilinx) et l'architecture hiérarchique (initialement utilisée dans les composants Altera). Cependant, une tendance apparait avec les dernières générations de circuits, les architectures sont principalement de style îlots de calculs. Les sections suivantes donnent quelques détails sur ces architectures. Dans le passé, d'autres architectures ont été utilisées. On peut citer l'architecture de routage logarithmique utilisée par Xilinx pour son circuit XC6000. Malheureusement, les outils de placement routage n'étaient pas adaptés à cette architecture pour permettre au concepteur d'en tirer pleinement partie.

Architecture îlot de calcul :

L'architecture la plus communément utilisée pour réaliser ces circuits est de type îlot de calcul. Dans ce cas les ressources configurables sont disposées sous formes de matrice, comme on peut le voir sur la figure 2. Des lignes de routage sont disposées horizontalement et verticalement autour des ressources configurables. Des blocs de connexion relient les ressources configurables aux lignes de connexion. Des matrices de connexion relient les lignes de routage horizontales et verticales[1].

L'utilisation de matrices de connexions configurables est indispensable pour assurer la connectivité des modules, mais les matrices de connexions configurables dégradent les caractéristiques des signaux, diminuent les performances (fréquence de fonctionnement et consommation de puissance) et nécessitent des outils de placement-routage efficaces. Sur la figure 2, on peut voir en gras des liaisons point à point entre deux éléments configurables. Ces liaisons utilisent : les ports d'entrées/sorties des éléments configurables, les connexions configurables qui permettent la connexion des éléments configurables au réseau de routage, les lignes de routages et les matrices de connexions configurables. Autant d'éléments parcourus qui dégradent les performances du circuit mais qui permettent une flexibilité importante.

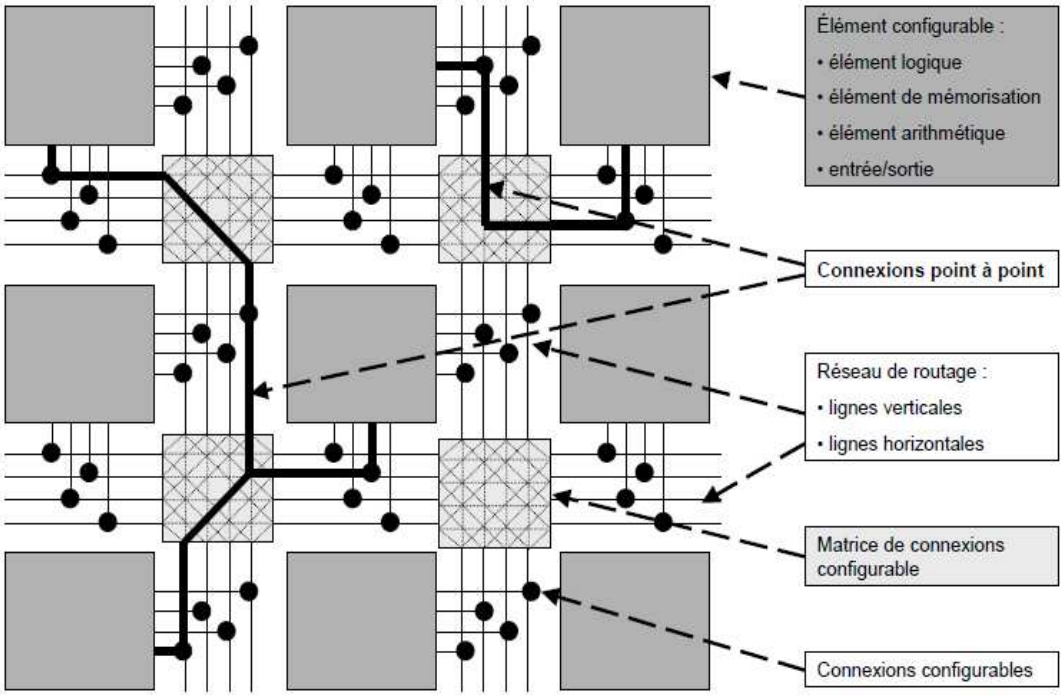

Figure 2 : Architecture îlot de calcul

Architecture hiérarchique :

L'architecture hiérarchique couramment utilisée pour les circuits FPGA est constituée de quatre ou trois niveaux. A chacun de ces niveaux des ressources de routage sont disponibles pour communiquer entre les éléments propres du circuit. La figure 3 schématise une architecture hiérarchique à quatre niveaux en utilisant un exemple d'architecture couramment rencontrée dans les FPGA ACTEL. Au niveau le plus haut de la hiérarchique, le circuit est constitué de tuiles agencées matricielles. Les tuiles sont constituées de clusters logiques et de bancs de mémoires. Enfin, les clusters logiques regroupent les éléments logiques (et/ou arithmétiques) configurables. Ce style d'architecture peut être très efficace énergétiquement car elle permet de localiser les communications intenses et limite l'utilisation de longues lignes de routage[1]. Cependant, il est nécessaire pour les algorithmes de placement et routage de prendre en compte les caractéristiques spécifiques de ces architectures.

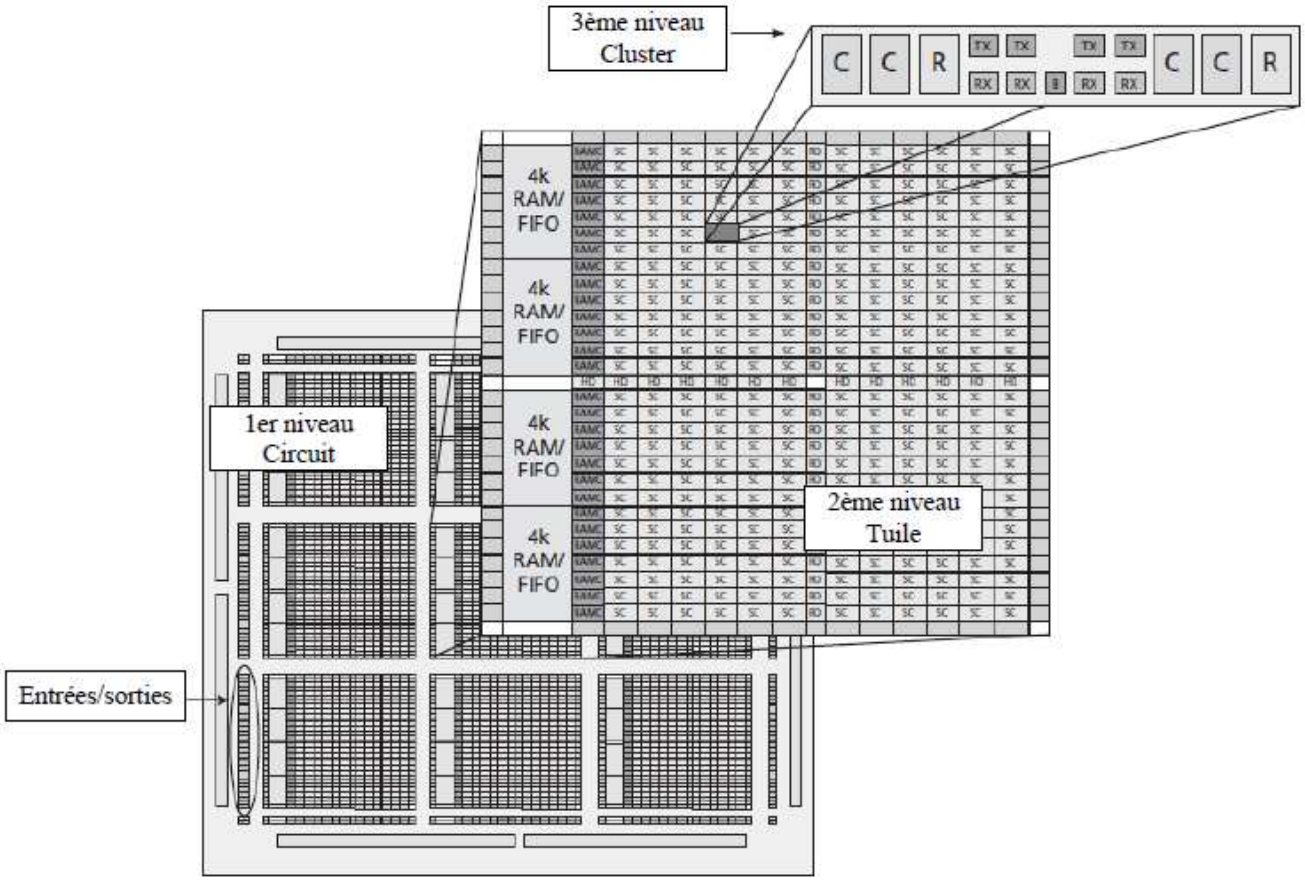

Figure 3 : Exemple d'Architecture hiérarchique à quatre niveaux (circuit, tuiles, clusters, éléments configurables)

2. DSP :

a. Généralités :

Un DSP est un processeur dont l'architecture est optimisée pour effectuer des calculs complexes en un coup d'horloge par la méthode VLIW, mais aussi pour accéder très facilement à un grand nombre d'entrées-sorties (numériques ou analogiques). La fonction principale utilisée dans le DSP est la fonction *MAC* (*Multiply and Accumulate*), c'est-à-dire une multiplication suivie d'une addition et d'un stockage du résultat (fonction très utilisée dans les calculs d'asservissement et de filtrage).

Les DSP sont utilisés dans la plupart des applications du traitement numérique du signal en temps réel. On les trouve dans les modems (modem RTC, modem ADSL), les téléphones mobiles, les appareils multimédia (lecteur MP3), les récepteurs GPS... Ils sont également

utilisés dans des systèmes vidéo, les chaînes de traitement de son, partout où l'on reçoit un signal complexe que l'on doit modifier à l'aide du filtrage.

Donc comme leur nom l'indique, ces circuits ont pour vocation première l'exécution la plus efficace possible des algorithmes de traitement du signal. Ils diffèrent en ce sens des microprocesseurs du monde PC dont l'usage est beaucoup plus général de part la diversité même des applications. Leur architecture est conçue de manière à pouvoir gérer efficacement les boucles de calculs intensives et à assurer une bonne précision numérique pour les données. Pour cela, ils disposent des caractéristiques suivantes :

- opérateurs arithmétiques spécialisés pour les calculs DSP.
- grande bande passante pour les données liée à l'utilisation de bus indépendants pour les données et les instructions (Architecture Harvard).
- unités de calculs d'adresses permettant la parallèlisation des calculs et des accès aux données, et proposant des modes d'adressage spécifiques au traitement du signal.
- matériel embarqué permettant l'exécution rapide des boucles logicielles.

A l'heure actuelle, on peut distinguer deux types de processeurs DSP. Les DSP « conventionnels » sont directement issus de la première génération, leur architecture n'a guère évoluée au fil du temps et l'amélioration de leurs performances tient avant tout aux progrès de la technologie (ex : l'*ADSP-218x d'AnalogDevices*). La deuxième génération est apparue il y'a quelques années avec l'apparition du *TMS320C6x* de *Texas Instruments*, le premier processeur basée sur une architecture VLIW. Ces processeurs, beaucoup plus puissants, intègrent des techniques matérielles évoluées identiques à celles que l'on trouve dans les microprocesseurs modernes. Ils visent avant tout le créneau des applications DSP hautes performances, tandis que les DSP conventionnels se placent sur le secteur du « faible coût / faible consommation ».

Donc on se focalisera sur la famille des DSP dite TMS320 de Texas Instruments, ce constructeur est en effet le leader sur le marché des processeurs de traitement du signal et de l'image. Après avoir rapidement survolé cette famille, nous nous intéresserons à l'un de ces DSP : le TMS320C67xx. Ce processeur est en effet dit « High Performance » et il est de plus en plus répandu.

Des DSP particuliers : la famille TMS320 :

Cette famille de DSP englobe des processeurs 16-32 bits, à virgule fixe ou flottante. Leur développement a commencé en 1982 avec le TMS32010, un DSP à virgule fixe.

Aujourd'hui, la famille TMS320 est divisée en trois plates-formes qui sont les TMS320C2000, TMS320C5000 et TMS320C6000. Cette dernière englobe deux types de processeurs, les TMS320C62xx, TMS320C64xx et TMS320C67xx. Afin de simplifier l'écriture, nous parlerons de 'C62, 'C64 et 'C67. On peut voir sur la figure 4 les différents types de processeurs de cette famille.

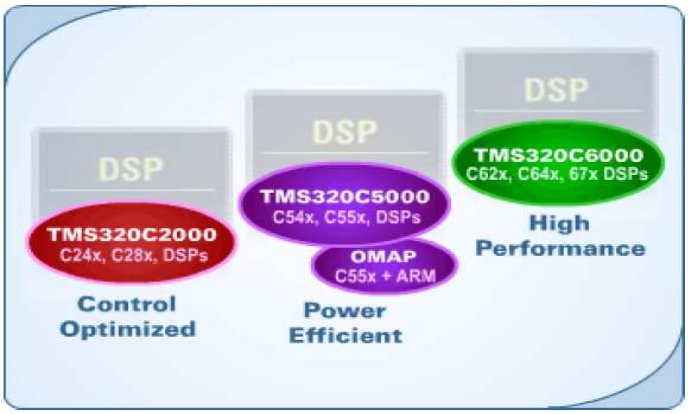

Figure 4 : Différent famille de Texas Instruments.

Les processeurs 'C6211, 'C6701, 'C6201 et 'C6202 travaillent respectivement à des fréquences de 150, 167, 200 et 250 MHz. Ce sont des architectures VLIW (Very Long Instruction Word) d'ordre 8, c'est-à-dire qu'ils sont capables d'exécuter jusqu'à 8 instructions 32-bits en parallèle. Le cœur de leur CPU (Central Process Unit) est constitué de 32 registres généraux de 32-bits, et de huit unités de traitement, deux multiplieurs et six ALU (Arithmetic and LogicUnits). Ces notions sont essentielles pour la compréhension des mécanismes internes et la programmation, notamment en assembleur.

b. Architecture interne du DSP :

Afin de bien comprendre le fonctionnement des DSP, il faut voir ce processeur comme un ensemble de blocs interconnectés.

On voit sur la figure 5 le détail de l'architecture interne du DSP basé sur le cœur C6000 (dont est dérivé le C67). Le cœur CPU possède :

Une unité de chargement de programme (program fetch), une unité de répartition des instructions (instruction dispatch), une unité de décodage des instructions, 32 regitres de 32-bits séparés en deux fois 16 registres (A et B), deux voies pour les données (data path) contenant chacune quatre unité de traitement, deux registres de contrôle et une logique de contrôle.

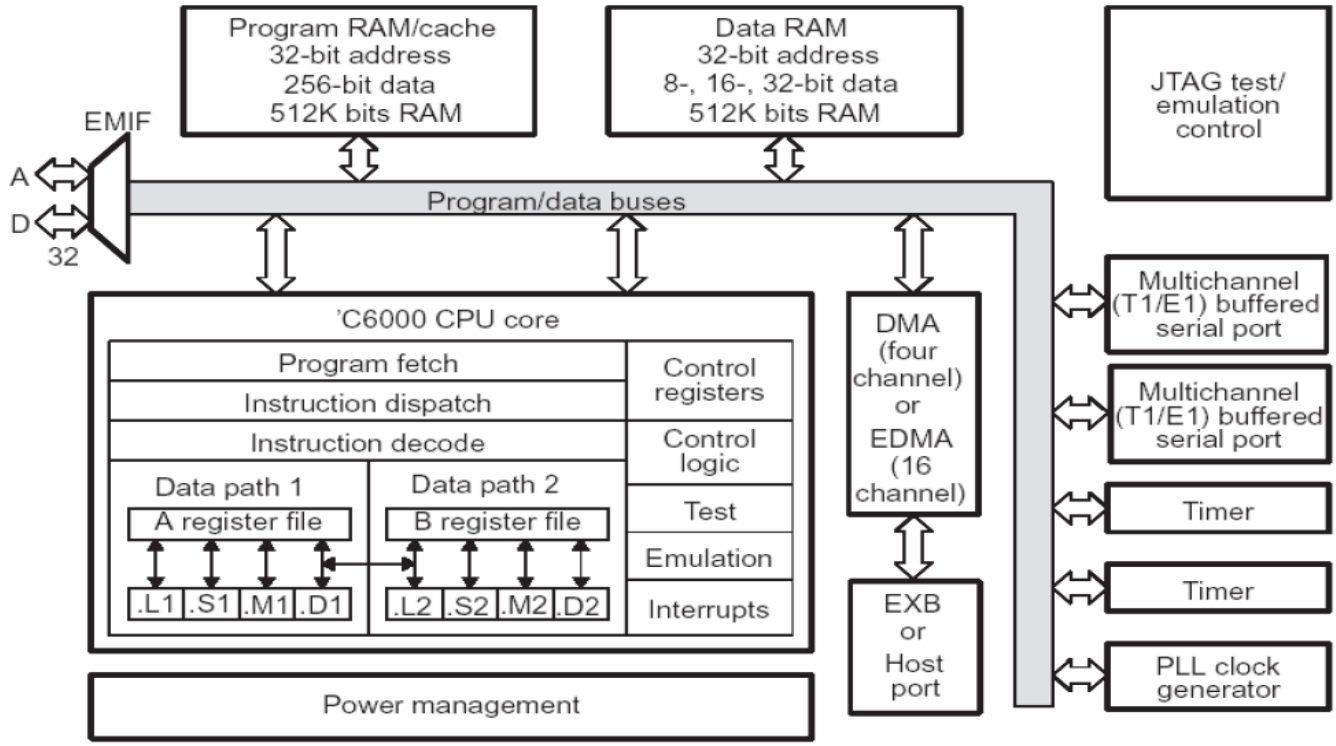

Figure 5 : Architecture interne du DSP.

Les unités exécutent des instructions logiques, de décalage (Shifting), des MultiplicationsHardwares et des opérations sur les adresses de Données. Tous les transferts de données entre la file de registre et la mémoire se font exclusivement par les unités « Data-adressing ».

Les quatre chemins sont croisés, ce qui permet l'échange de données entre les deux files de registre. La figure montre les deux « data paths » du 'C67 où apparaissent :

Les deux files de registre (A et B), les huit unités (.L1, .L2, .M1, .M2, .S1, .S2, .D1 et .D2), les deux voies de chargement depuis la mémoire LD1 et LD2 (LD=Load), les deux voies de stockage vers la mémoire ST1 et ST2 (ST=Store), deux croisements (registre file cross path) entre les files de registre 1X et 2X et deux chemins d'adresse de données (data addresspaths) DA1 et DA2.

Les registres généraux supportent des données sur 32 et 40-bits pour les calculs en virgule fixe. Les données sur 40-bits sont contenues dans deux registres : les 32-bits de poids faibles (LSB) sont stockés dans un registre pair et les 8-bits de poids fort restant (MSB) sont stockés dans les 8-bits de poids faible (LSB) du registre situé immédiatement au-dessus (qui est forcément impair). Le 'C67 peut également utiliser cette même paire de registres pour les calculs double-précision sur 64-bits.

Un processeur à virgule flottante est généralement plus coûteux, car il a plus d'une estimation réelle ou une plus grande puce en raison de circuit supplémentaire nécessaire pour gérer les nombres entiers ainsi que l'arithmétique flottante. Plusieurs facteursentrent en jeu lors du choix d'un DSP spécifique tels que le coût, la consommation d'énergie et la vitesse. Les processeurs C6x sont particulièrement utiles pour les applications nécessitant des calculs intensifs. La famille de la C6x comprend deux processeurs à virgule fixe (par exemple, C62x, C64x) et à virgule flottante (par exemple, C67x).

La figure ci-après montre l'organisation générale en blocs d'un DSP basé sur une architecture VLIW et Harvard :

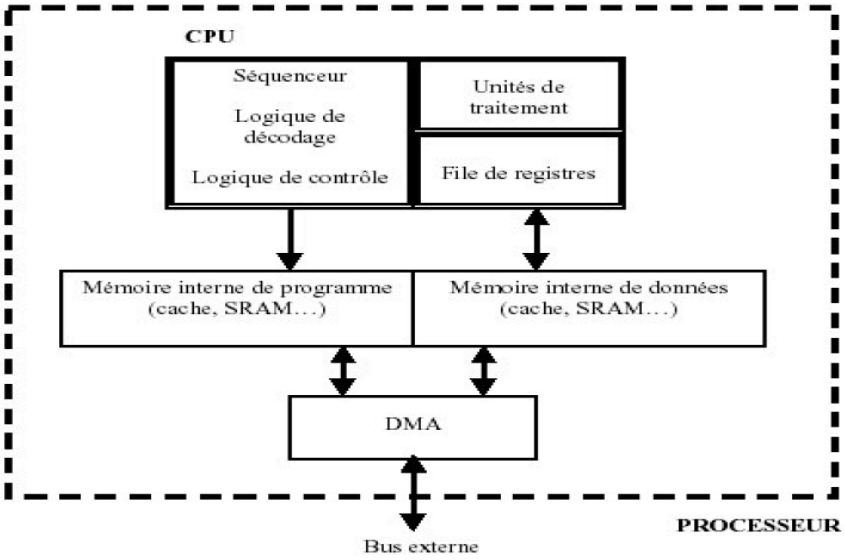

Figure 6 : l'organisation en blocs d'un DSP basé sur une architecture VLIW et Harvard.

La différence principale entre ce DSP et le TMS320C6201 est qu'il a la capacité de réaliser des calculs en virgule flottante en double précision, suivant une norme IEEE.

On peut également s'intéresser à ses principales caractéristiques :

La tension d'alimentation de 1.8V (pour le cœur) ou 2.5V ainsi la fréquence d'horloge nominale de 167MHz. Celle-ci est paramétrable, comme dans le cas du 'C6201, grâce à un multiplieur de fréquence par 1 ou 4, Architecture Harvard (mémoire de programme et mémoire de données séparées) et VLIW d'ordre 8, Pipeline d'une profondeur de 16 étages (11 dans le cas du 'C62), la mémoire de programme fonctionne suivant 4 modes (MAPPED, CACHE, FREEZE, BYPASS), Possède un DMA (Direct Memory Access) qui permet de faire des transferts de données de l'extérieur vers la mémoire interne sans passer par le CPU et

Possède une EMIF (External Memory InterFace) qui permet de faire l'interfaçage entre différents types de mémoire externe et le DSP.

Conclusion :

Ce chapitre nous a permis de comprendrela différence entre un DSP et un FPGA au niveau de l'architecture en précisant les éléments de bases et les différentes architectures dans chaque cible. Donc selon notre besoin et selon le type d'application en traitement de signal, on choisit la cible qui répond aux exigences en termes de vitesse, consommation et coût.

Chapitre 2 : Implémentation du filtre numérique FIR sur FPGA et DSP

Introduction :

Dans le cadre pratique, une implémentation du filtre FIR sur le FPGA et le DSP nécessite une bonne connaissance sur le fonctionnement d'un filtre FIR pour l'utilisation des nouveaux outils permettant de faciliter les tâches de la programmation software du filtre afin de l'implémenter, simuler et tester son fonctionnement en temps réel.

I. Filtre Numérique :

1. Définition :

En électronique, un filtre numérique est un élément qui effectue un filtrage à l'aide d'une succession d'opérations mathématiques sur un signal discret. C'est-à-dire qu'il modifie le contenu spectral du signal d'entrée en atténuant ou éliminant certaines composantes spectrales indésirées. Contrairement aux filtres analogiques, qui sont réalisés à l'aide d'un agencement de composantes physiques (résistance, condensateur, inductance,transistor, etc.), Les filtres numériques étant généralement réalisés par des processeurs, ils sont définis à l'aide de langages de programmation[9].

Les filtres numériques peuvent, en théorie, réaliser la totalité des effets de filtrage pouvant être définis par des fonctions mathématiques ou des algorithmes. Les deux principales limitations des filtres numériques sont la vitesse et le coût. La vitesse du filtre est limitée par la vitesse (l'horloge, le « *clock* » en anglais) du processeur. Pour ce qui est du coût, celui-ci dépend du type de processeur utilisé. Par contre, le prix des circuits intégrés ne cesse de diminuer, et les filtres numériques se retrouvent partout dans notre environnement, radio, téléphone cellulaire, télévision, lecteurs MP3, etc.

Un filtre numérique peut être défini par une équation aux différences, c'est-à-dire l'opération mathématique du filtre dans le domaine temporel (discret).

La forme générale du filtre d'ordre M est la suivante :

$$y[n] = \sum_{k=1}^{M} b_k * x[n-k] - \sum_{k=1}^{M} a_k * y[n-k]$$

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_N x[n-N] - a_1 y[n-1] - a_2 y[n-2] + \cdots + a_M y[n-M]$$

Exemple :

$$y[n] = x[n] + 1{,}5\, y[n-1] - 0{,}85\, y[n-2]$$

Une fonction de transfert, dans le domaine fréquentiel (Transformée en Z), permet également de définir un filtre numérique. Ainsi, la fonction de transfert générale d'ordre N d'un filtre numérique est la suivante :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k * z^{-k}}{\sum_{k=0}^{M} a_k * z^{-k}}$$

$$H(z) = \frac{b_0 + b_1 * z^{-1} + b_1 * z^{-2} + \cdots + b_N * z^{-N}}{1 + a_1 * z^{-1} + a_2 * z^{-2} + \cdots + a_M * z^{-M}}$$

Exemple :

$$H(z) = \frac{1}{1 - 1{,}5z^{-1} + 0{,}85z^{-2}}$$

La valeur des coefficients a et b fixera le type de filtre, passe-bas, passe-haut, etc[9].

2. Filtre FIR

En traitement numérique du signal, le filtre à réponse impulsionnelle finie ou filtre RIF (en anglais *Finite Impulse Responsefilter* ou *FIR filter*) est un filtre numérique qui est caractérisé par une réponse uniquement basée sur un nombre fini de valeurs du signal d'entrée. Par conséquent, quel que soit le filtre, sa réponse impulsionnelle sera stable et de durée finie dépendante du nombre de coefficients du filtre.

Parmi les filtres linéaires, les filtres à réponse impulsionnelle finie, sont opposés aux filtre à réponse impulsionnelle infinie (filtre RII) qui eux ne peuvent être réalisés qu'avec des implémentations récursives qui remplacent une convolution sur une plage infinie, par un nombre fini d'états internes qui dépendent de l'histoire passée du filtre.

De façon générale le filtre à réponse impulsionnelle finie est décrit par la combinaison linéaire suivante :

$$y[n] = b_0 * x[n] + b_1 * x[n-1] + b_2 * x[n-2] + \cdots + b_N * x[n-N]$$

où $x[i]_{1 \le i \le n}$ représente les valeurs du signal d'entrée et $y[i]_{1 \le i \le n}$ les valeurs du signal de sortie.

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

Les remarques générales suivantes peuvent être portées sur les filtres FIR :

- Les filtres FIR sont forcément stables, peu importe les coefficients utilisés
- La complexité d'un filtre IIR(Infinite Impulse Responsefilter)est moindre que celle d'un filtre RIF du même ordre. Cette propriété peut être utile sur les plateformes limitées en puissance de calcul
- Généralement, les filtres FIR sont moins sensibles aux erreurs de quantification que les filtres RII. L'absence de récursivité empêche les erreurs cumulatives.
- Un filtre FIR est moins sélectif qu'un filtre IIR du même ordre. C'est-à-dire que la transition entre la bande passante et la bande rejetée est moins rapide que dans le cas du filtre IIR.
- Contrairement à un IIR, un filtre FIR peut avoir une réponse impulsionnelle symétrique et introduire un retard sur le signal mais aucun déphasage.

Les filtres numériques peuvent être réalisés à l'aide de trois éléments ou opérations de base. Soit l'élément gain, l'élément de sommation et le retard unitaire. Ces éléments sont suffisants pour réaliser tous les filtres numériques linéaires possibles. La réalisation présentée dans la figure suivante est une réalisation directe de type 1 du filtre FIR[8].

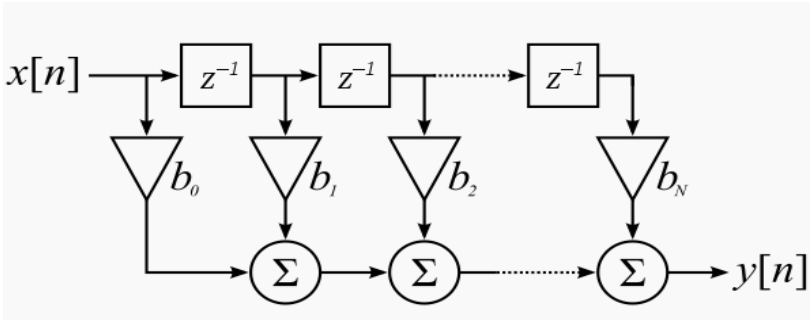

Figure 7 : Structure générale d'un filtre FIR.

3. Pourquoi un filtre FIR

Une des questions qu'on peut se poser c'est dans quelles situations il nous convient d'utiliser un filtre genre FIR ou un filtre genre IIR.

Les filtres IIR ont l'avantage qu'ils sont efficaces. Avec très peu de pôles et zéros on peut la plupart des réponses fréquentielles dont on peut avoir besoin dans les applications audio. Cependant, le filtre étant rétroactive, les erreurs de précision numérique deviennent une question d'importance, car ils peuvent s'amplifier et devenir dehors contrôle, d'abord dans la forme de bruit, mais éventuellement dans la formed'instabilité. La forme de la réponse impulsionnelle n'est pas facile àdéterminer, car elle est définie indirectement par les pôles et zéros de H(w).

Par contre, les filtres FIRn'ont jamais des problèmes d'instabilité, car la sortie n'est qu'une somme finie des échantillons de l'entrée. Un autre avantage des RIF est le retard de group constant, qui permet d'avoir une distorsion de phase minimale sur le signal traité. La réponse impulsionnelle dans le cas RIF est parfaitement contrôlable[10].

II. Implémentation sur FPGA et DSP :

1. La carte TSW6011 du FPGA:

a. Description de la carte TSW6011 du FPGA:

Présentation du TSW6011EVM de Texas Instruments :

La carte d'évaluation TSW6011EVM de Texas Instruments est la seule carte de canal RX (canal de réception) qui peut être utilisée pour démontrer à la fois les performances d'un dispositif TRF3711 et l'algorithme de correction IQ (I : In phase, Q :Quadrature de phase) qui est intégré dans le FPGA.

La carte TSW6011EVM contient les circuits de base suivants :

Un récepteur des signaux RF TRF371125 qui contient aussi un filtre passe bas ; un convertisseur Analogique-Numérique des données ADS5282 ; un FPGA Altera Cyclone III pour le traitement adaptatif du signal numérique pour la correction IQ ; un générateur d'horloge CDCE62005 pour générer les horloges du système et un convertisseur Numérique-

Analogique DAC5672 pour faciliter la mesure de la performance à travers un analyseur de spectre[3].

La carte TSW6011EVM fournit des options pour envoyer un signal RF d'entrée directement à l'entrée du TRF371125 ou par l'un des deux amplificateurs à faible bruit LNA en déplaçant deux résistances. En outre, il y a une option pour activer deux ADC (Analogue to Digital Conversion) que l'on a besoin à partir d'une source externe. Cette source peut être asymétrique (single-ended)ou un signal différentiel (differential signal). Par exemple, la sortie CMOS du TRF3711EVM peut être interfacée avec l'entrée CMOS d'une TSW6011 pour effectuer des tests de température de la TRF371125. Il y a également une option pour contourner l'oscillateur intégré avec une source externe.

Schéma du TSW6011EVM :

Le schéma de principe de la carte TSW6011 est illustré sur la figure 8. Les données de sortie peuvent également être analysées à l'aide de deux autres sorties. Une sortie d'un connecteur CMOS qui fournit des données numériques, cette interface dispose d'un réseau RC sur toutes les données et les signaux d'horloge pour permettre à l'utilisateur de brancher un 'connecteur CMOS analyseur logique directement sur le connecteur.

Une autre option pour la capture des données est d'utiliser un autre connecteur branché surune autre carte d'acquisition de données TSW1200EVM de Texas Instruments. Ce connecteur J21 sur la TSW6011 contient des données LVDS qui peuvent être reçus par un conseil TSW1200. Le connecteur d'entrée LVDS sur les TSW1200 se branche directement sur J21 de la TSW6011. Cette interface peut être utilisée pour capturer les données de l'ensemble des quatre ADC utilisés par cette conception. Le FPGA reformate la sortie de données binaires sans signe, qui est requise par le TSW1200[3].

Les principales composantes sur TSW6011 sont les suivants:

- TRF3711: intégré directementle démodulateur en quadrature.
- ADS5282: 12-bit avec 8canaux ADC (jusqu'à 65 MSPS (Million Symbole Per Seciond)).
- CDCE62005: 5/10 générateur d'horloge de sortie.
- DAC5672: 14 bits.

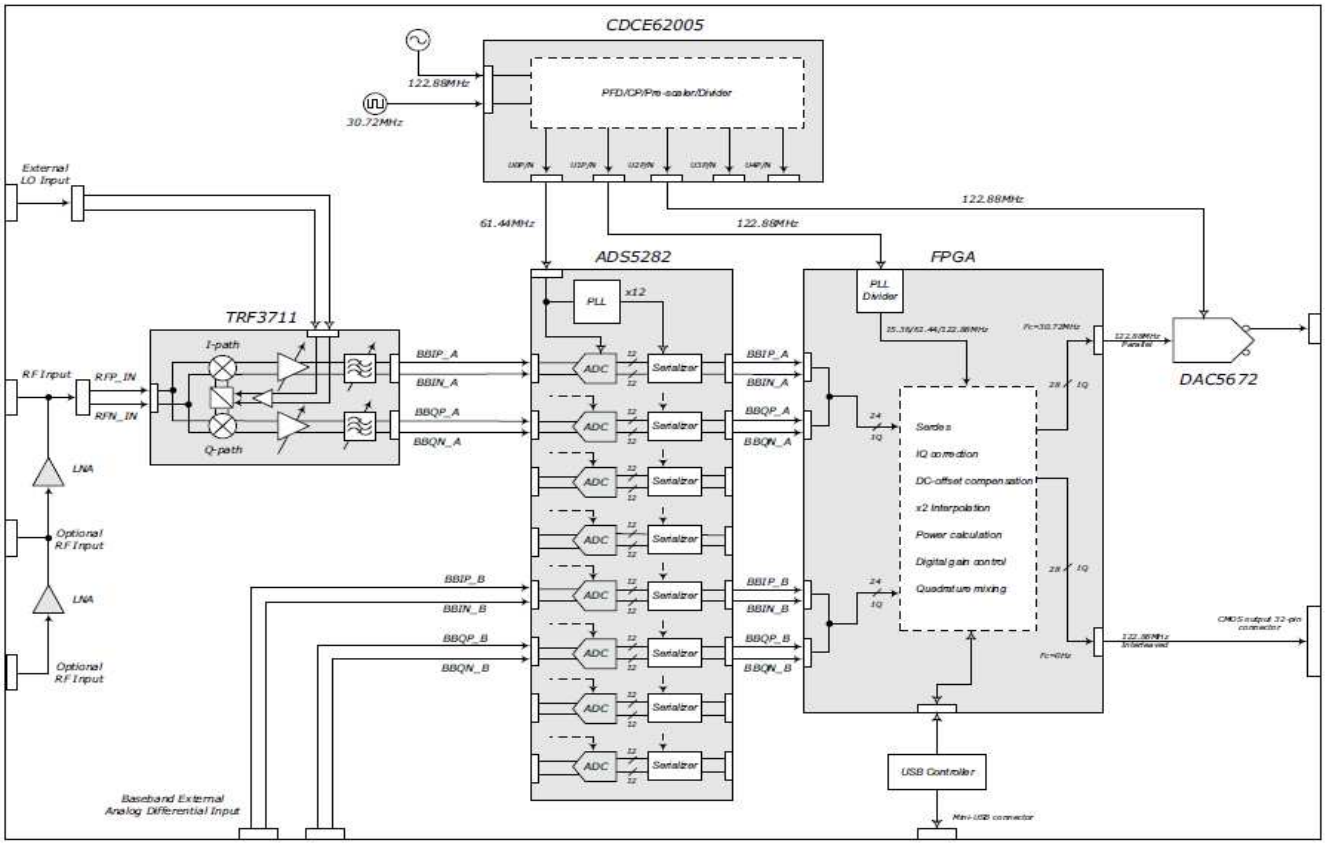

Figure 8 : Schéma synoptique de la carte TSW6011

b. Les difficultés de l'implémentation :

Parmi les problèmes rencontrés lors de l'implémentation du filtre FIR sur cette carte est que toutes les entrées sont de type analogiques RF sachant que celles du FPGA sont de type numériques, donc on a besoin de convertir les signaux analogiques en signaux numériques,et ceci en utilisant un convertisseur analogique numérique de la carte,et chaque sortie de l'ADC qui est codé sur 12 bits sont suivis par un bloc de sérialisation c'est-à-dire de 12 bits parallèle vers série de type différentiel. Alors les données reçuesà partir de l'ADS sont sous format séries et pour faire le traitement sur ces données il faut ajouter un bloc en VHDL sur le FPGA qui fait l'opération inverse c'est-à-dire de la série au parallèle et ensuite entrer ces données à l'entrée du filtre car elle est de type parallèle.

Le 2ème problème c'est que pour utiliser ce convertisseur il faut le configurer à travers une interface proposée par Texas Instruments, cette configuration nous permet de choisir quel ADC on va utiliser, l'annulation du bruit ou non et aussi la configuration du circuit TRF par l'ajustement du gain et l'activation du filtre passe bas et autre configurations.L'interfaceest

connectée à la carte via le port USBalors que les données reçues par ce port sont dirigées seulement vers le FPGA ce qui signifie que ce dernier est l'uniqueresponsable de la configuration du TRF et du ADC, donc l'achat de l'IP original d'Alteraet indispensable pour réaliser cette tâche. C'est-à-dire le programme qui fait la configuration des circuits TRF et ADC

c. Les solutions proposées :

Pour arriver à implémenter le filtre sur le FPGA et tester son fonctionnement on a évité l'utilisation duTRF et de l'ADC de la carte. Donc on a proposé comme solution la création des données numériques à l'intérieur du FPGA.

Ces données ont été prises à partir du testbench générer par le FDATOOL deMatlabet étaient stockés dans le FPGA afin de les utiliser comme des entrées pour le filtre,et pour y arriver on a ajouté une mémoire en VHDL sur le FPGA pour stocker ces données et ensuite faire une lecture de la mémoire, donc la sortie de cette dernière est l'entrée du filtre.

d. Design et simulation du filtre :

La 1er étape est la réalisation du filtre numérique FIR de type passe bande à l'aide de l'outil FDATOOL du Matlab, alors on choisit les caractéristiques du filtre selon nosbesoin.

Dans notre cas on a choisi un filtre FIR passe bande d'ordre 40 avec la 1er fréquence de coupure qui est égale à 7.6KHz et la 2ème est de l'ordre de 14KHz, comme le montre la figure suivante :

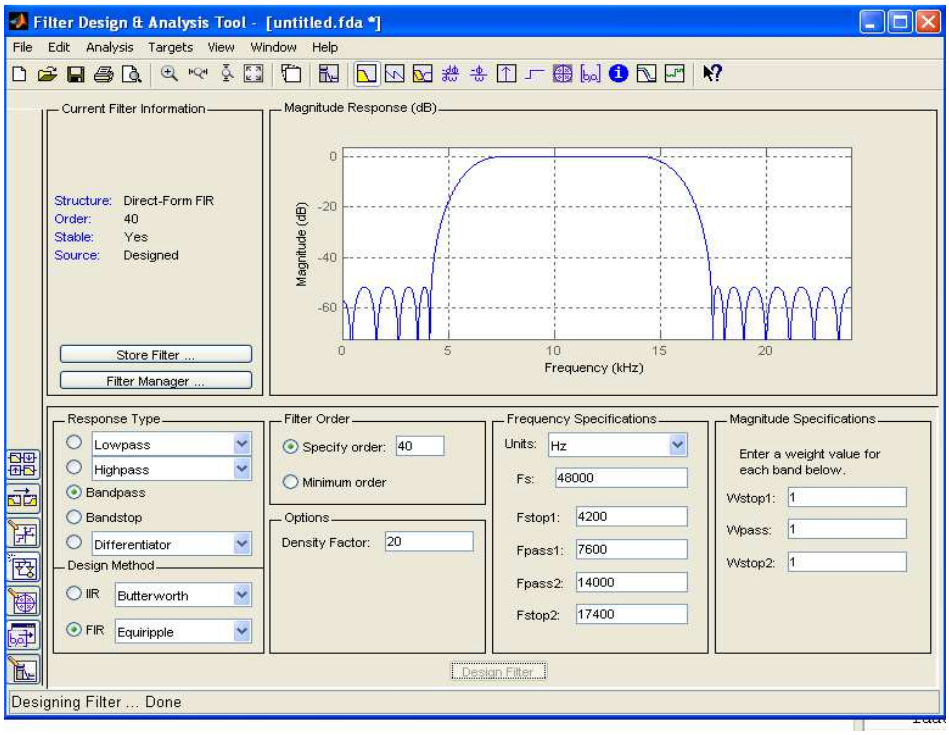

Figure 9 : design du filtre sur FDATOOL.

Après avoir définit le filtre et entrer toutes les caractéristiques, un simple clique sur la 3ème icône du haut qui apparait à droite de la fenêtre, et qui est nommée « Set quantizationparameters », permet de configurer le nombre de bit d'E/S et le type de l'implémentation entre 'Fixed-point' ou 'Floating-point'et autre paramètres comme le montre la figure suivante :

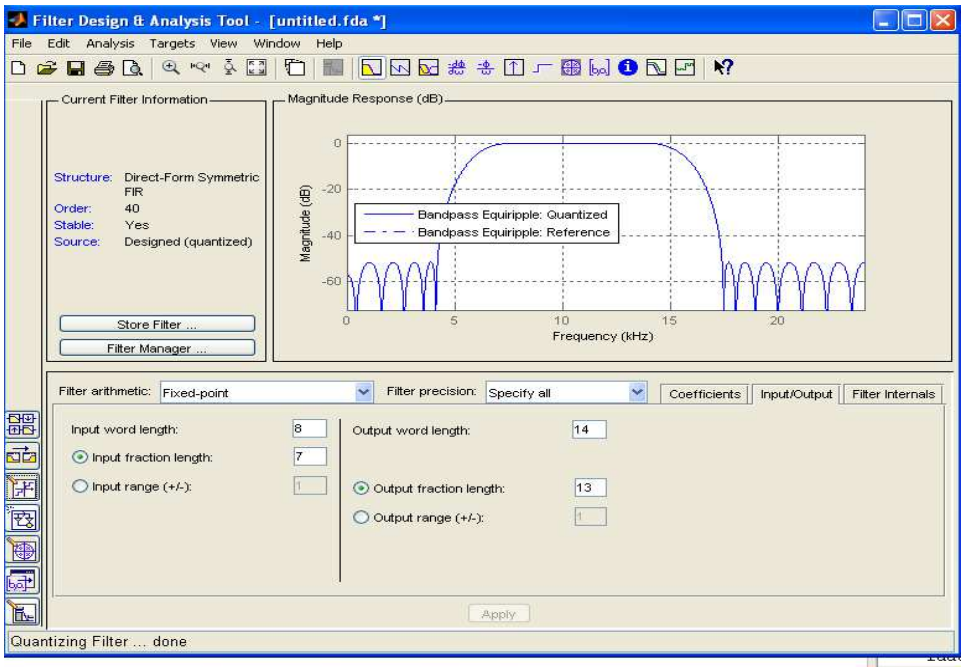

Figure 10 : Set quantizationparameters

Lorsque tous les paramètres sont bien déterminés, on peut ainsi générer le code VHDL spécifique à notre filtre qui est déjà configuré. Pour ce faire, dans le menu on sélectionne le bouton 'Target' puis 'Generate HDL …' pour configurer tous les paramètres associés au filtre concernant l'architecture, le Test Bench….

Après la génération du code VHDL et le Test Bench, et pour vérifier le code du filtre en VHDL et son fonctionnement, on a obtenu les résultats suivants sur le logiciel ModelSim.

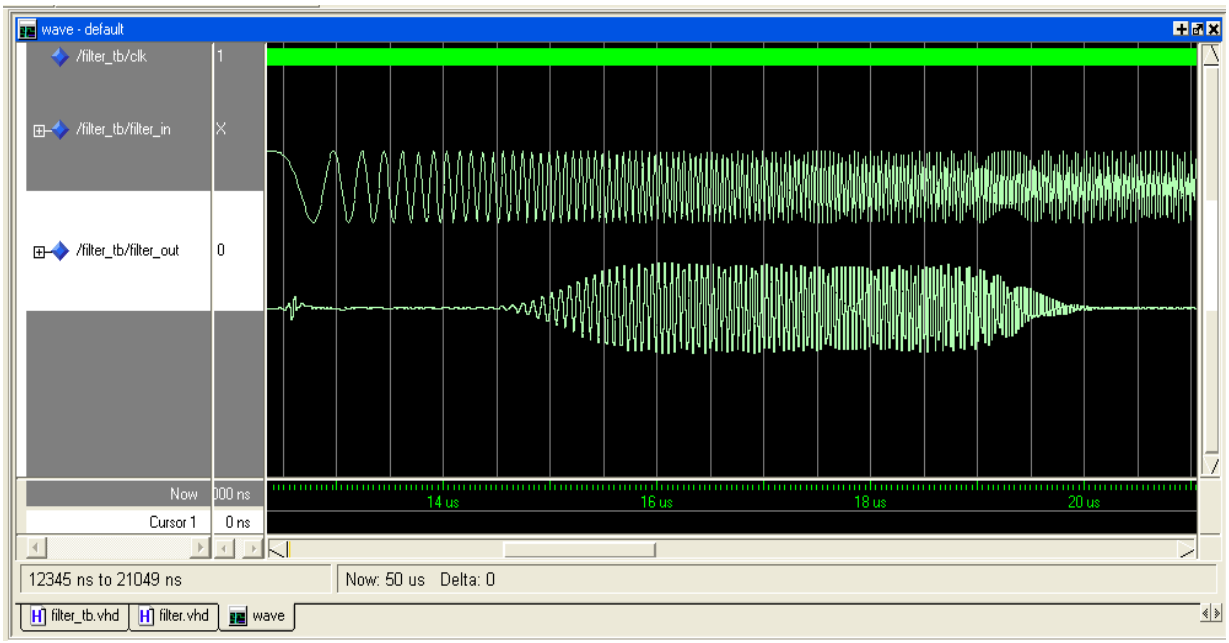

Figure 11 : Simulation du filtre sur ModelSim.

Donc on remarque que le signal d'entrée généré par le Test Benchet un signal sinusoïdal d'une fréquence variable comme le montre la figure précédente, et on remarque qu'on a obtenu à la sortie un signal filtré. Ce filtre joue le rôle d'un filtre passe bande, il laisse passer juste les fréquences inclues dans cette bande.Dans cette simulation la première fréquence de coupure est : 8 KHz et la deuxième fréquence de coupure est : 13 KHz

e. Implémentation sur FPGA :

Ensuite on passe au logiciel Quartus II d'Altera, qui représente un logiciel d'analyse et de synthèse pour créer le fichier .SOF permettant le transfert du Hard vers le circuit FPGA.

Donc comme on a déjà mentionné dans la partie solutions proposées, on a besoin de la mémoire pour stocker les données numériques et pour y arriver on a utilisé un outil très puissant intégré dans Quartus II c'est l'utilisation de l'outilMegaCorequi permet de générer

le code en VHDL ou en Verilog des différents blocs comme : la mémoire RAM, les filtres de traitement d'images et vidéos, bloc de la transformée de Fourrier et d'autre blocs et fonctions.

On a jusqu'à 4000 valeurs numériques du signal d'entrée qui est généré par le Test Bench, alors on a utilisé une mémoire RAM de 4096 mots de 8bits c'est-à-dire un bus d'adresse de 12bits ($2^{12} = 4096$).

L'entrée du filtre représente les données stockées sur la mémoire RAM de 8bits (l'outil MegaCore de Quartus II ne permet de générer que des mémoires de 8bits) pour cela on a choisi un vecteur de 8bits pour l'entrée du filtre.

Et pour la sortie du filtre, on remarque qu'elle estcodée sur 14 bits parce que l'entrée du convertisseur numérique analogique et aussi codée sur 14 bits.

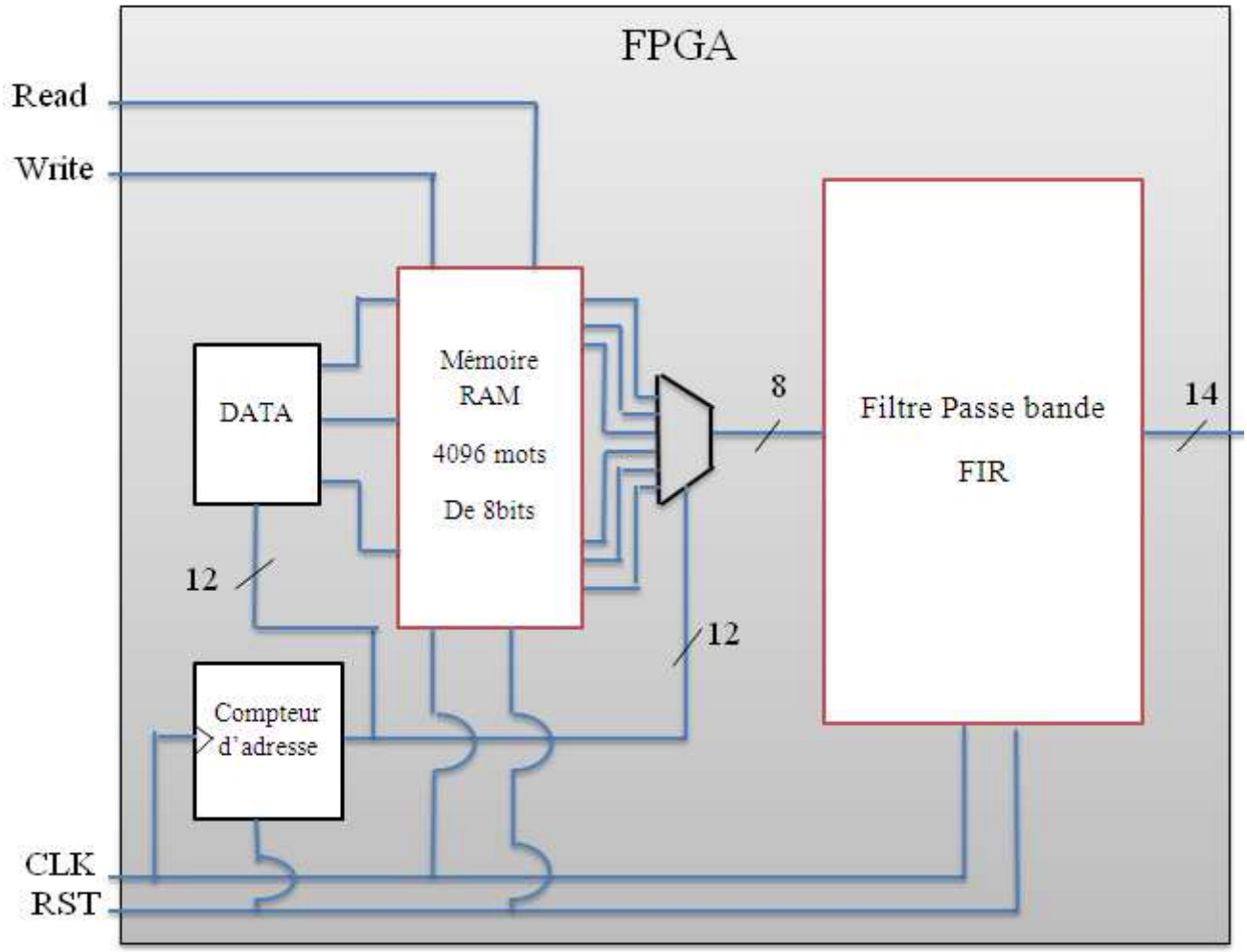

Figure 12 : Schéma synoptique du projet réalisé sur FPGA.

Donc on peut voir l'ensemble descomposants réalisés en VHDL dans un seul programme avec la création dans le programme principal des appels aux composants décrits dans le schéma synoptique du FPGA.

La vue générale de l'utilisation de la carte est montrée dans la figure suivante :

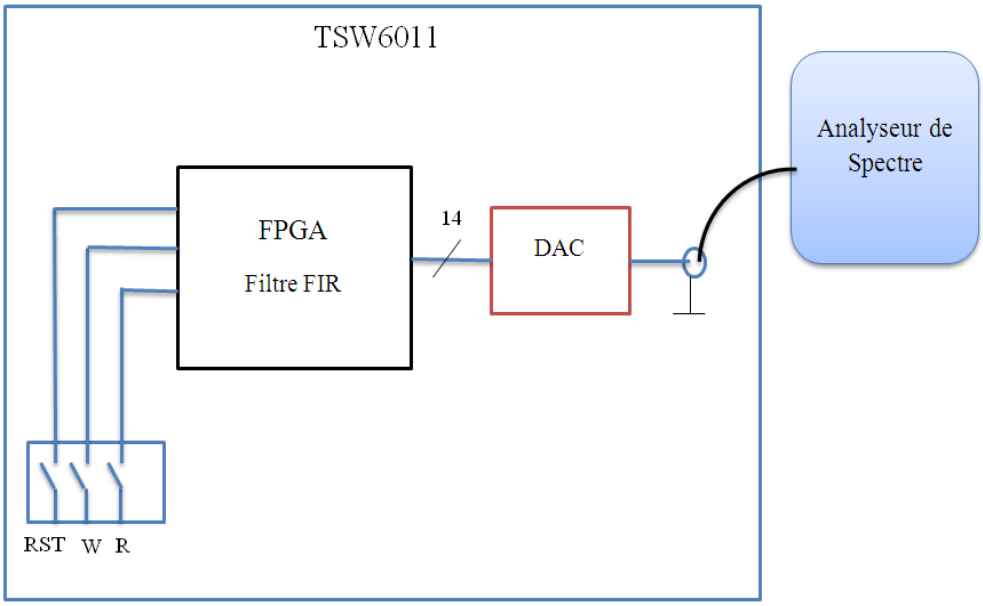

Figure 13 : Schéma synoptique du projet sur la carte.

Pour tester le bon fonctionnement du filtre on a utilisé l'outil 'Signal Tap' de Quartus II, qui permet de visualiser les états des signaux aux Pins du FPGA en temps réel, c'est-à-dire on n'a pas besoin de l'oscilloscope ou l'analyseur de spectre pour vérifier le fonctionnement d'un tel code en VHDL implémenter sur le FPGA.

Pour ce faire on a déjà simulé le même code sur le logiciel ModelSim avec l'utilisation du Test Bench et on a comparé les résultats de ModelSim avec celles obtenus en temps réels sur le Signal Tap, et on a trouvé que le filtre a bien fonctionné parce qu'on a trouvé les mêmes résultats comme illustre les figures suivantes, ces figures nous ont donnés un aperçu sur quelques états de la sortie du filtre.

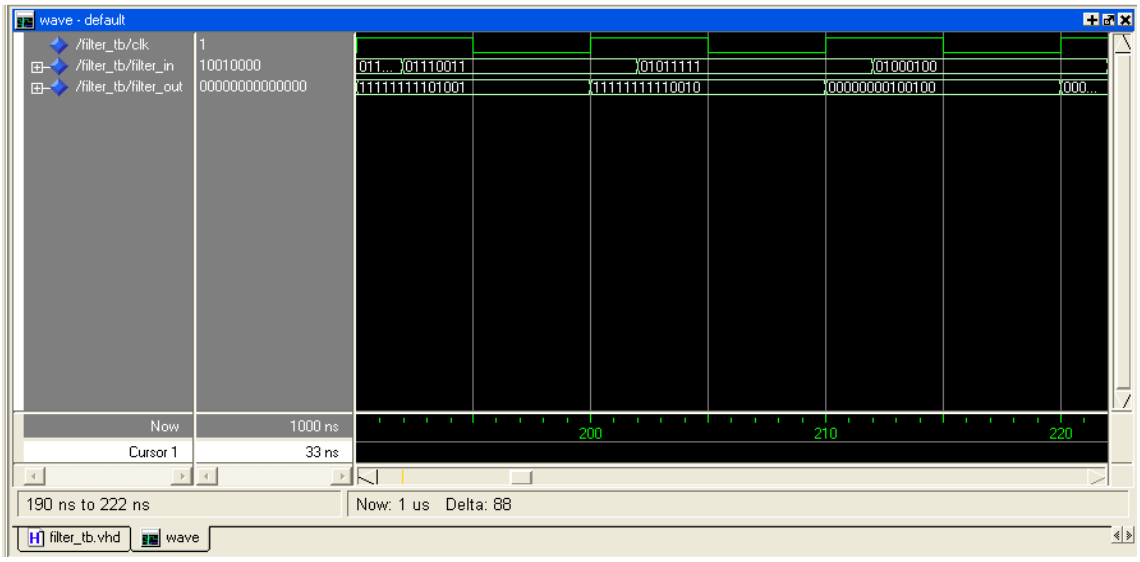

Figure 14: Simulation sur ModelSim.

Pour simuler la sortie du filtre on a visualisé en binaire pour qu'on puisse faire une comparaison entre les signaux sur ModelSim et la sortie du filtre en temps réel sur Signal Tap.

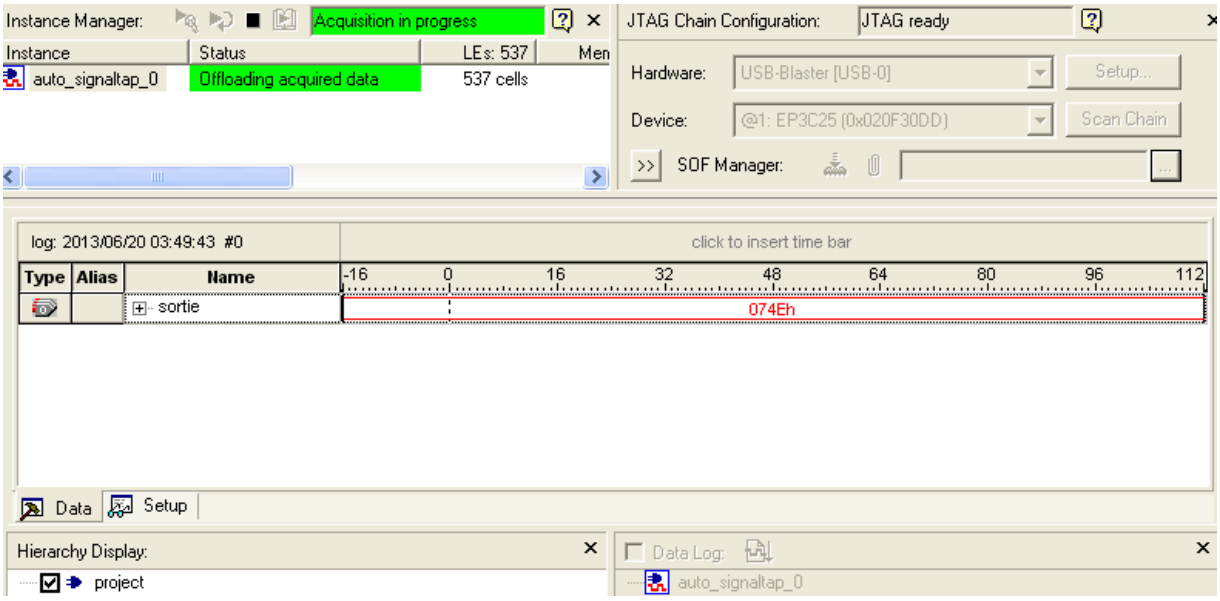

Figure 15: Visualisation en temps réel du signal de sortie sur Signal Tap.

2. La carte C6713 DSK du DSP :

a. Description de la carte C6713DSK :

Pour réaliser le projet et faire une conception d'un programme, les outils suivants sont nécessaires :

Le paquet DSK (DSP Starter Kit) comprend:

- Code Composer Studio (CCS), qui fournit les outils nécessaires de soutien. CCS fournit un environnement de développement intégré (IDE), réunissant le compilateur C, assembleur, éditeur de liens, un débogueur, et ainsi de suite.
- La carte qui est montrée sur la figure 16, contient le (C6713) processeur de signal numérique à virgule flottante TMS320C6713, ainsi qu'un codec stéréo 32 bits en entrée et en sortie.
- Un câble universel bus synchrone (USB) qui relie la carte DSK à un PC.
- Une alimentation 5V pour la carte DSK.

La carte DSK se connecte au port USB du PC via le câble USB fournit avec ce package.

L'oscilloscope, le générateur de signal et les haut-parleurs[2].

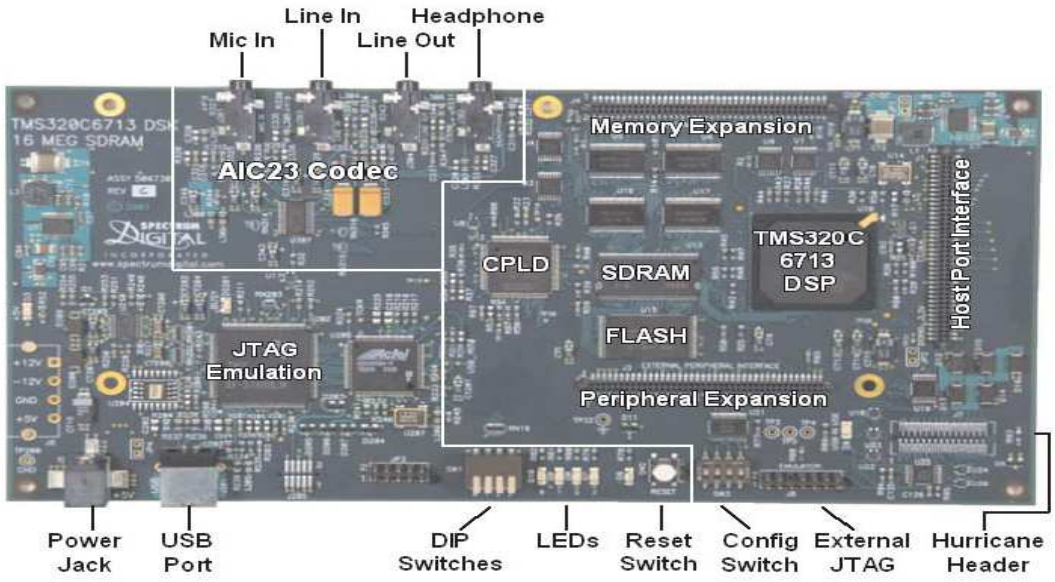

Figure 16: la carte C6713DSK.

Carte DSK :

Le paquet DSK est encore relativement peu coûteux (395 $) et il est puissantavec les outils de support de matériels et logiciels nécessaires pour le traitement du signal en temps réel. Il s'agit d'un système de DSP complet. La carte DSK a une taille approximative de 5*8 inches, et comprend un processeur de signal numérique C6713 à virgule flottante et un codec stéréo 32-bit TLV320AIC23 (AIC23) pour l'entrée et la sortie.

Le codec AIC23 utilise une technologie sigma-delta qui offre un ADC et DAC. Il se connecte à un système dont la fréquence d'échantillonnage est de 12 MHz, et qui peut être facilement réglable et peut varier de 8 à 96 kHz.

La carte DSK comprend 16 Mo de mémoire synchrone dynamique à accès aléatoire (DRAM) et 256 ko de mémoire flash. Quatre connecteurs de la carte fournissent des entrées et sorties : MIC IN pour l'entrée micro, entrée ligne pour l'entrée ligne, sortie ligne pour la sortie de ligne et casque pour une sortie casque (multiplexé avec la sortie ligne).

L'état des quatre commutateurs DIP de l'utilisateur sur la carte DSK peut être lu à partir d'un programme et fournit à l'utilisateur une interface de contrôle de la rétroaction. Cette carte DSK fonctionne à 225MHz et intègre des régulateurs de tension fournissent un voltage de 1,26 V pour le noyau C6713 et 3,3 V pour la mémoire et les périphériques[2].

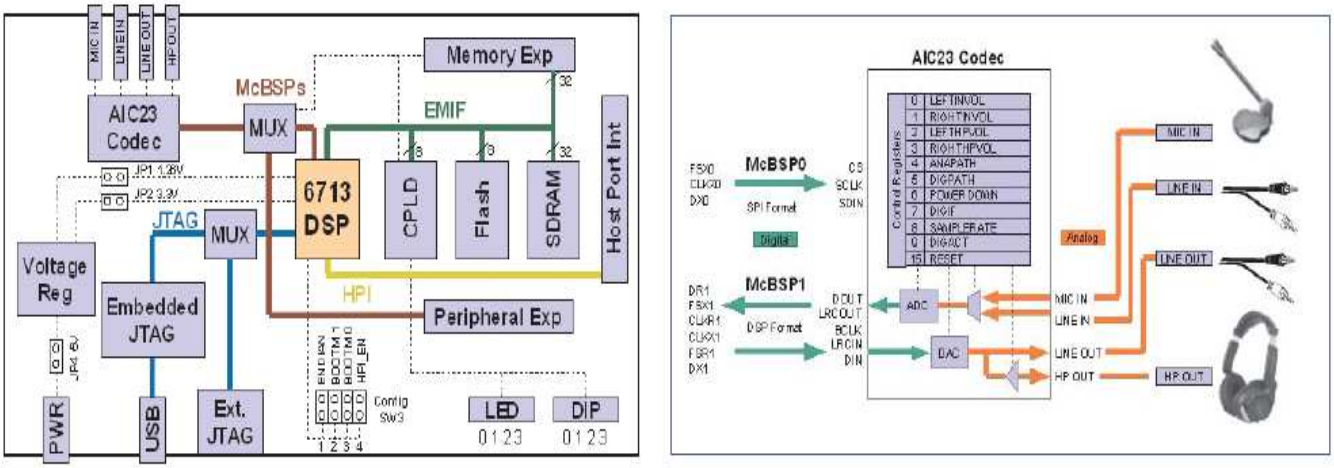

Figure 17: Schéma synoptique de la carte et du Codec.

b. Design et simulation du filtre :

Les coefficients du filtre sont les responsables du changement du type du filtre, et à partir de ces coefficients on peut jouer sur la nature du filtre et aussi sur les fréquences de coupures.

Toujours avec l'outil FDATOOL, on peut générer ces coefficients après avoir configuré tous les paramètres corresponds à notre besoin.

Sur le menu on clique sur le bouton 'Target' puis 'générer le C Haeder', ce fichier contient tous les coefficients de notre filtre qu'on peut utiliser dans le programme en C sur le logiciel Code Composer Studio.

Après avoir écrit le programme en C sur Code Composer Studio, ona entré toutes les bibliothèques nécessaires comme le montre la figure suivante :

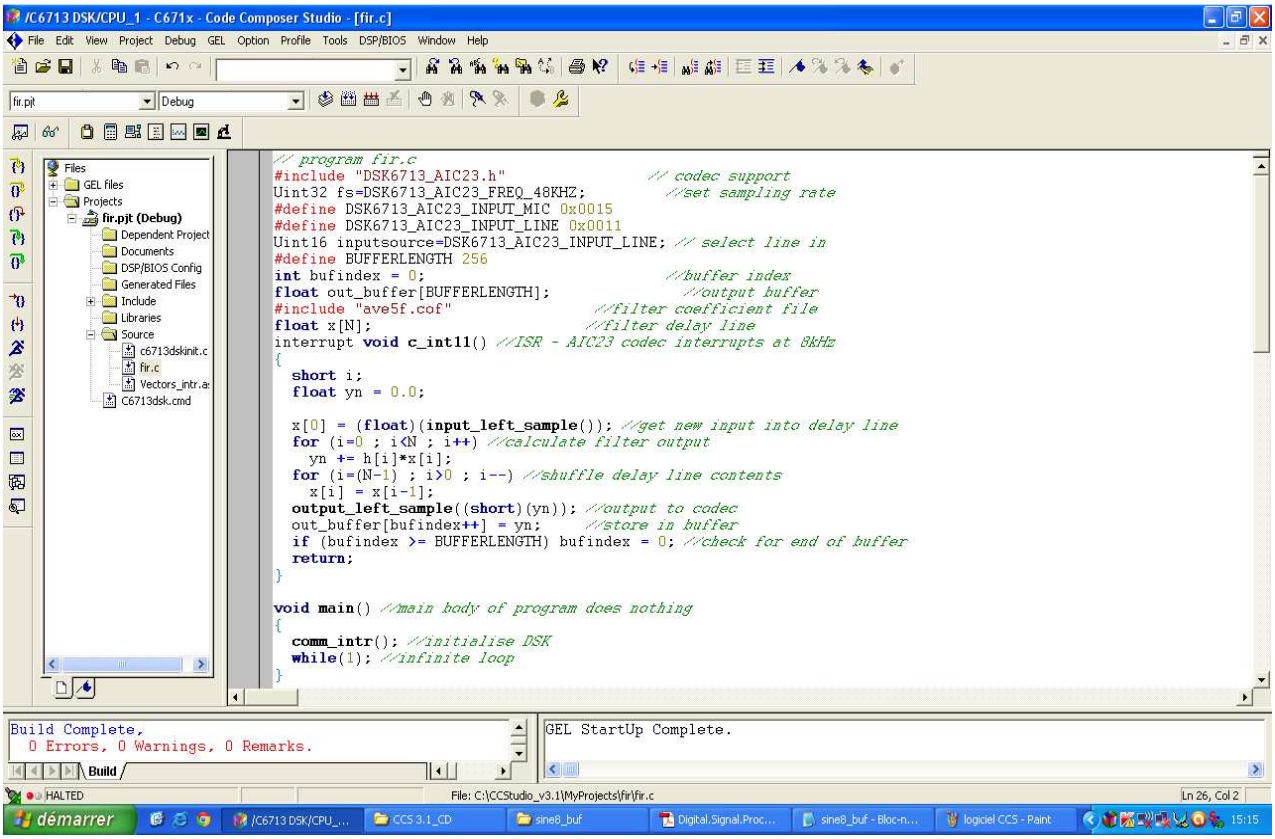

Figure 18: Programmation sur Code Composer Studio 3.1

On a simulé le programme sur le logiciel et par un outil d'affichage des graphes sur CSS on a trouvé les résultats suivants :

Dans le domaine temporel, on remarque qu'il représente le fonctionnement d'un filtre passe bande comme le montrer la figure suivante :

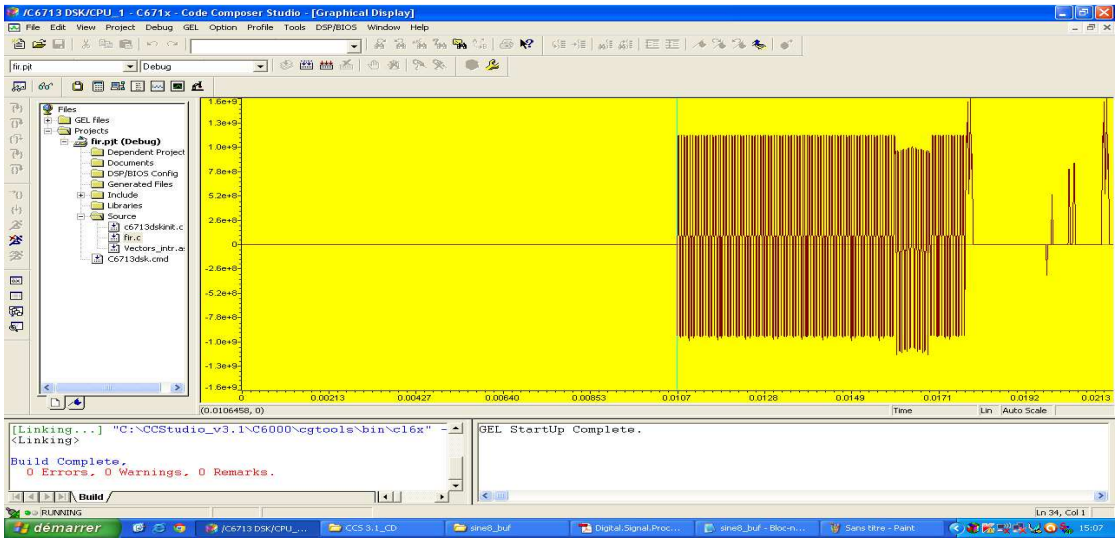

Figure 19 : Simulation dans le domaine temporel.

Et dans le domaine fréquentiel, on a obtenu le même graphe que l'on a déjà mentionné dans le FDATOOL, et on remarque que la 1ere fréquence de coupure est de l'ordre de 10KHz et la 2ème fréquence de coupure est de 12KHz, on remarque que la bande passante est un peu réduite par rapport à celle qu'on a déjà configurée dans le FDATOOL Matlab, ceci est dû à la valeur de sortie du filtre enregistrée dans un registre, donc il y a une perte d'information lors du stockage de cette valeurdans un nombre bien déterminé de bits.

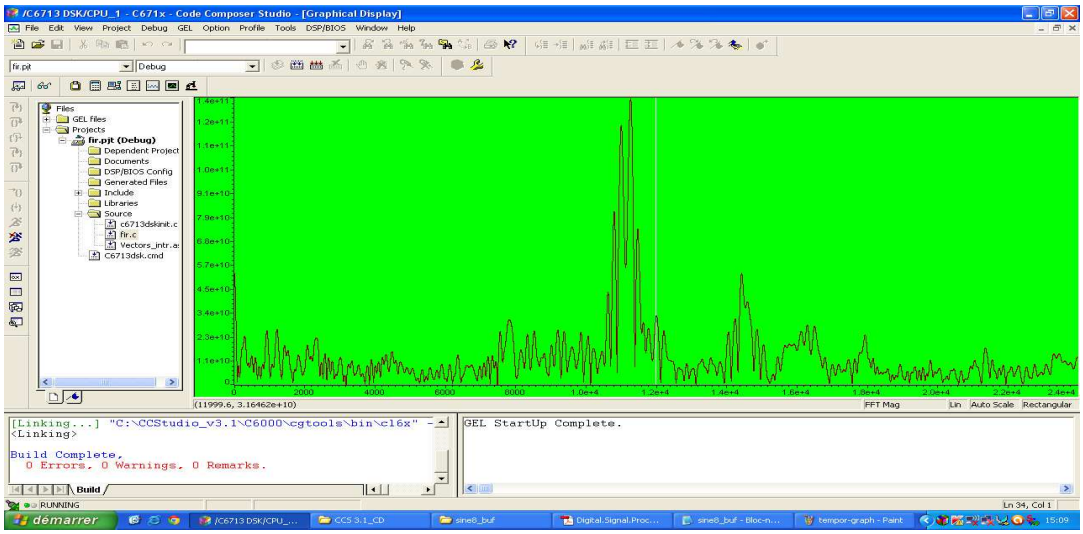

Figure 20: Simulation dans le domaine fréquentiel.

c. Implémentation et test :

Pour l'implémentation sur la carte C6713DSK, on a effectué le chargement du programme sur le DSP TMS320C6713 puis l'opération 'Run' pour exécuter le programme en temps réel.

Donc à la fin on a obtenu un filtre passe bande avec une bande variante de 8KHz à 14KHz, et on remarque que ce sont les mêmes fréquences que l'on a déjà mentionné sur le FDATOOL.

La simulation sur l'oscilloscope est la suivante :

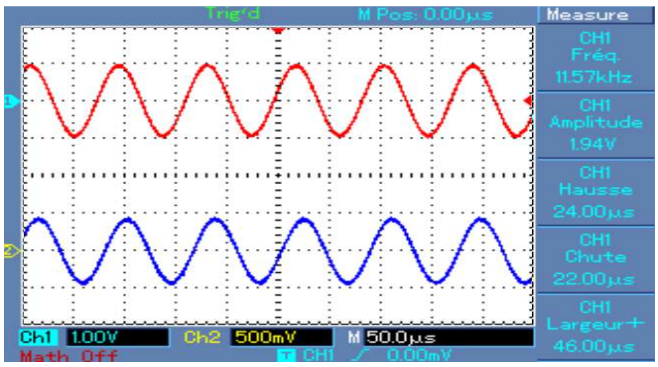

Figure 21: Simulation du filtre sur oscilloscope.

Conclusion :

Les difficultés rencontrées lors de l'implémentation sur le FPGA, nous ont permis de bien comprendre le côté pratique et aussi de nous familiariser avec la carte TSW6011.Malgrétous les obstacles rencontrés lors de l'implémentation, nous avons proposé une solution acceptable pour tester le fonctionnement du filtre.

Ainsi l'implémentation sur DSP nous a servis de vérifier le bon fonctionnement du filtre en temps réel sur l'oscilloscope.

Chapitre 3 : Comparaison de deux implémentations sur les différentes cibles

Introduction :

Dans ce chapitre nous détaillons la différence entre un DSP et un FPGA au niveau de la vitesse d'exécution des instructions, la consommation d'énergie et le coût, ces paramètres nous permettent de distinguer la meilleure cible pour implémenter des algorithmes de traitement de signal.

I. Différentes contraintes pour différents domaines

1. Performance :

La performance requise par une application est directement fonction de la contrainte tempsréel et de la complexité des algorithmes mis en œuvre. Plus la contrainte temps-réel est forte, moins le système dispose de temps pour exécuter les algorithmes requis. Si les algorithmes eux-mêmes sont complexes, c'est-à-dire que le nombre d'opérations nécessaires pour les réaliser est très élevé, alors la *charge de calcul* (le nombre d'opérations à réaliser par unité de temps) requise par l'application est importante et le système devant la réaliser doit être puissant. En règle générale, la contrainte temps-réel d'un système est fortement liée à la fréquence d'échantillonnage des données qu'il traite. Les systèmes Radar fonctionnent à des fréquences très élevées (Figure 22) et doivent donc traiter de grandes quantités de données en un temps très court, mais les algorithmes mis en œuvre sont relativement simples ; au contraire, les modèles numériques utilisés en météorologie ne sont pas soumis à unecontrainte de temps très forte (quelques prévisions dans la journée suffisent en général) mais la complexité des algorithmes est infiniment plus grande. Ces deux types d'application, bien que soumises à des contraintes différentes, requièrent donc toutes les deux des systèmes disposant d'une grande puissance de calcul.

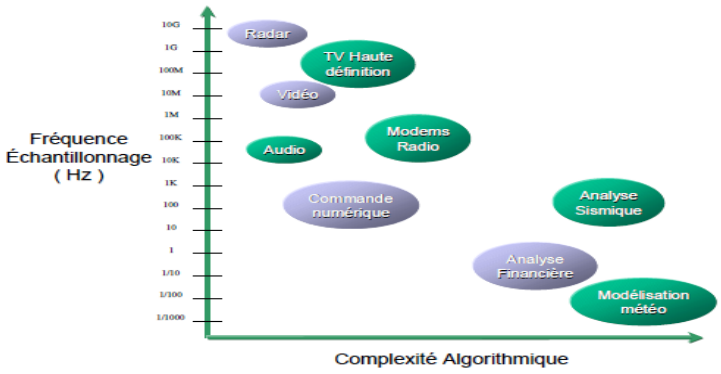

Figure 22 : Estimation Fréquence / Complexité pour différents types d'application

2. Consommation électrique :

La consommation est devenu le critère le plus important pour le domaine de l'embarqué. L'autonomie est devenue un argument de vente majeur et les industriels cherchent à réduire au minimum la consommation des systèmes dans leurs produits. Les perspectives d'augmentation de la capacité des batteries étant limitées, les efforts se tournent vers la conception de circuits assurant de grandes puissances de calcul pour une consommation la plus faible possible.

3. Le coût des circuits numériques :

Le coût financier d'un circuit intégré du point de vue du fabricant est une notion plus complexe qu'il n'y paraît, et difficile à évaluer *à priori*, c'est-à-dire au tout début du flot de conception. On peut l'estimer comme la somme de deux composantes principales : le coût de production « brut » du composant, et le coût de développement.

Le coût de production dépend de nombreux facteurs tels que la technologie utilisée, le coût du test, le rendement de fabrication, le type de packaging employé, etc[4]. Du point de vue d'un concepteur de circuits, la surface de silicium est un facteur très important puisque le coût de production d'un circuit en fonderie est directement lié à la surface employée. Le nombre de « pattes » externes de l'interface a aussi une grande influence pour les circuits destinés à l'intégration sous forme de composants discrets. Dans le cas qui nous intéresse, celui des cœurs de circuits intégrables dans des *System-On-Chip (Soc)*, ce dernier facteur n'intervient pas ; la recherche de la surface minimale devient donc le principal moyen pour réduire le coût d'un cœur de circuit.

Le coût de développement est quant à lui proportionnel au temps de conception et au coût humain. Son importance par rapport au coût final est inversement proportionnelle au volume de production. Les facteurs qui influent sur le coût de développement sont la complexité du circuit, le temps de test, la disponibilité et la qualité des outils de conception. A l'heure où l'industrie électronique utilise de plus en plus de standards (*MPEG* pour la vidéo, *GSM* et *UMTS* pour la téléphonie), la notion de flexibilité est devenue primordiale. Une solution flexible permet de prendre en compte les changements de standards au dernier moment, et évite ainsi de reconcevoir le circuit. Le gain est double : coût de conception réduit et temps d'accès au marché (*Time-to-Market*) accéléré, ce dernier point étant

particulièrement critique dans les marchés très concurrentiels comme celui de la téléphonie mobile.

II. Interprétation des résultats de l'implémentation :

1. Interprétation sur FPGA :

Après avoir écrit le programme en VHDL de notre filtre sur le logiciel Quartus II, il nous a permis de faire l'analyse et la synthèse du filtre par plusieurs paramètres qui entrent en jeu.

La synthèse du programme nous donne les résultats suivants :

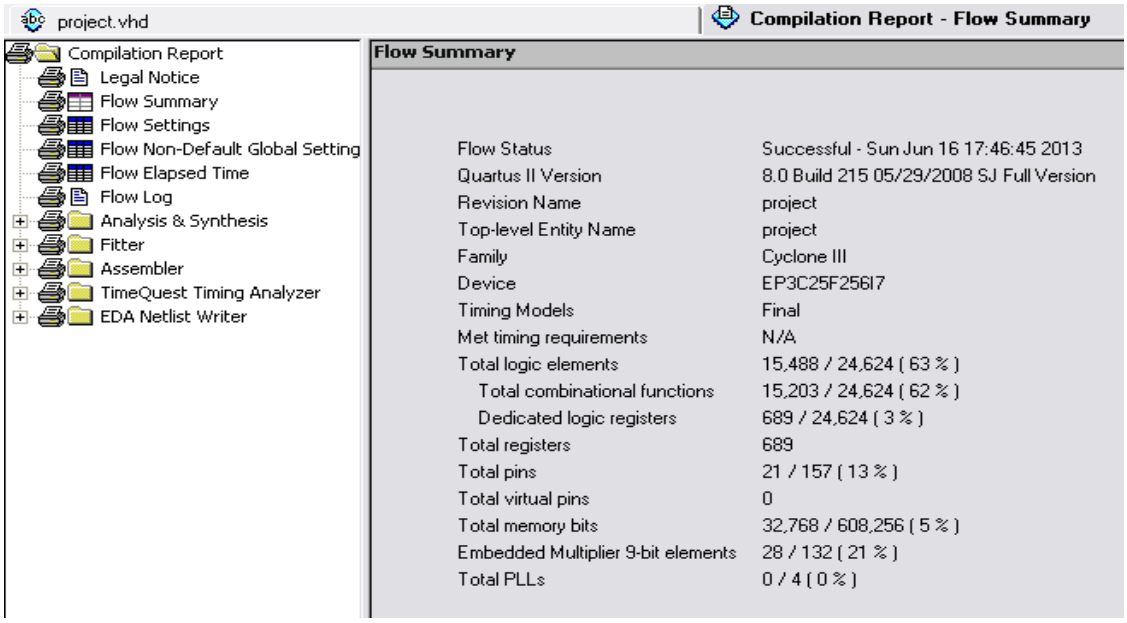

Figure 23: Rapport de compilation sur Quratus II.

On remarque qu'on a consommé jusqu'à 15488 parmi 24624 éléments logiques donc un pourcentage d'occupation de 63% dans le FPGA d'Altera Cyclone III.

Pour les registres on a une consommation de 689 parmi 24624 donc un pourcentage de 3%, concernant la mémoire interne de type RAM M9K du FPGA, on a consommé 32768 bits parmi 608256 c'est-à-dire que le FPGA contient 66 RAM de type M9K donc correspond au 594 Kbits = 76032 o = 74,25 Ko.

Et pour les multiplieurs embarqués de 9 bits, le filtre consomme 28 multiplieurs parmi 132 existants sur le FPGA.

a. Vitesse d'exécution :

La fréquence de fonctionnement du FPGA est de 122 MHz donc comme estimation de la vitesse on obtient :

- FPGA à 122 MHz avec 28 multiplieurs :

  122 000 000 * 28 = **3 416 000 000 opérations par seconde.**

Et une estimation maximale de la vitesse d'exécution est :

- FPGA à 122 MHz avec 132 multiplieurs :

  122 000 000 * 132 = 16 104 000 000 opérations par seconde.

b. Consommation électrique :

Quartus II nous a permis aussi de faire une estimation de la puissance consommée par le FPGA en utilisant un outil de compilation qui s'appelle 'PowerPlay', c'est un outil puissant permettant d'analyser le système et faire une estimation de puissance en donnant le résultat avec le Watt et aussi il permet de développer de nombreuses optimisations électriques automatiques qui sont transparentes pour le concepteur, mais offrent une utilisation optimale de l'architecture FPGA à minimiser la puissance.

Dans notre cas la puissance estimée est la suivante et comme le montre la figure suivante :

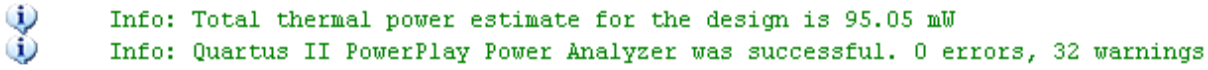

Figure 24: Estimation de consommation de la puissance sur FPGA.

Donc la puissance estimée pour le design est de l'ordre de **95.05** mW.

c. Coût :

Lors de la conception d'un FPGA à faible coût, nous devons régler le compromis entre puissance, performance, fonctionnalités et le coût global du dispositif. Nous sommes face à au défi d'aller au marché à la hâte avec la qualité, des produits de haute gamme avec un prix raisonnable (figure 25) :

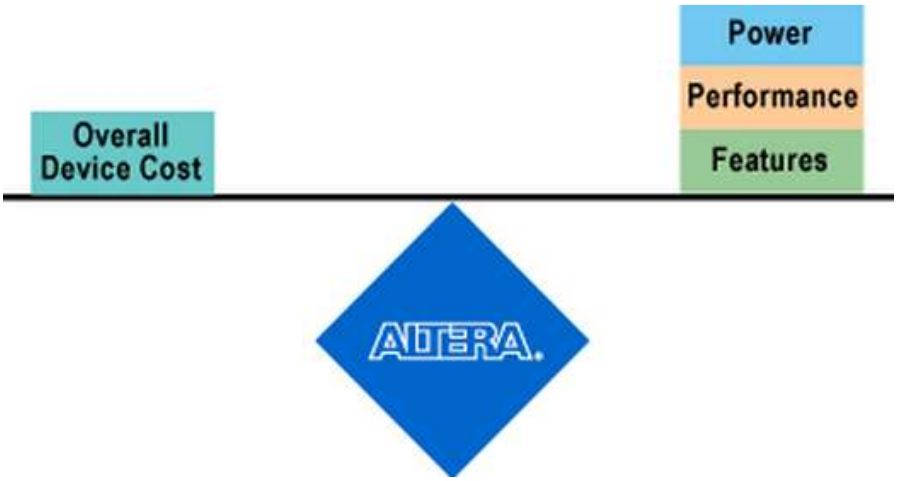

Figure 25 : Equilibre puissance, performance, fonctionnalité et coût.

Altera a adopté une nouvelle méthodologie de conception pour atteindre les objectifs de faible coût pour les applications à haut volume. L'approche traditionnelle de « l'optimisation par élimination » consiste à réduire le coût d'un produit existant à haute densité en éliminant les fonctions du logiciel. Bien que cette méthode soit peu efficace dans la réduction des coûts de FPGA, il n'atteint pas les niveaux de prix les plus bas possibles pour une taille donné de la matrice et l'emballage[11].

2. Interprétation sur DSP :

Pour le DSP et avec le logiciel Code Composer Studio on peut faire le profilingc.à.d connaître le nombre de cycle d'exécution d'une fonction ou une boucle ou une générale une rangée.

Pour y arriver à connaître le nombre de cycle de la boucle 'for' qui fait la multiplication des données entrant au DSP avec les coefficients du filtre, on a fait les deux tests suivants sur le logiciel Code Composer Studio :

a. Benchmarking (Profiling) sans optimisation :

Pour profiler le code du filtre il faut vérifier que le compilateur utilisé est `(-g)`et le linker utilisé est `(-c -o)` comme le montre la figure suivante :

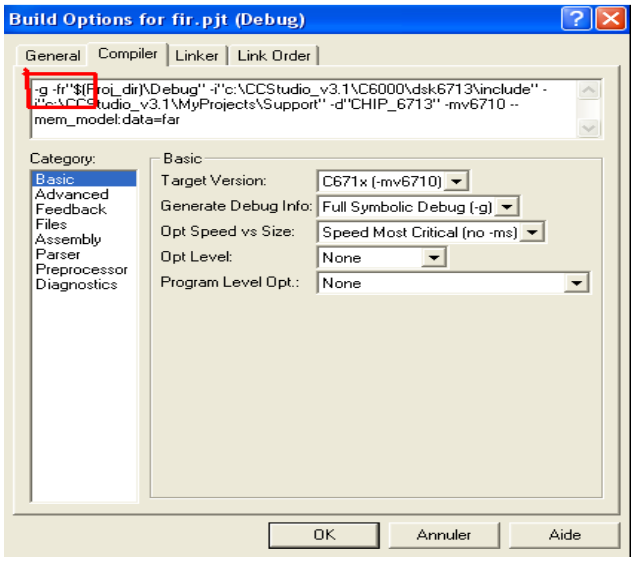

Figure 26: Configuration du compilateur sans optimisation.

Après avoir effectué duprofiling sans optimisation on obtient le résultat suivant :

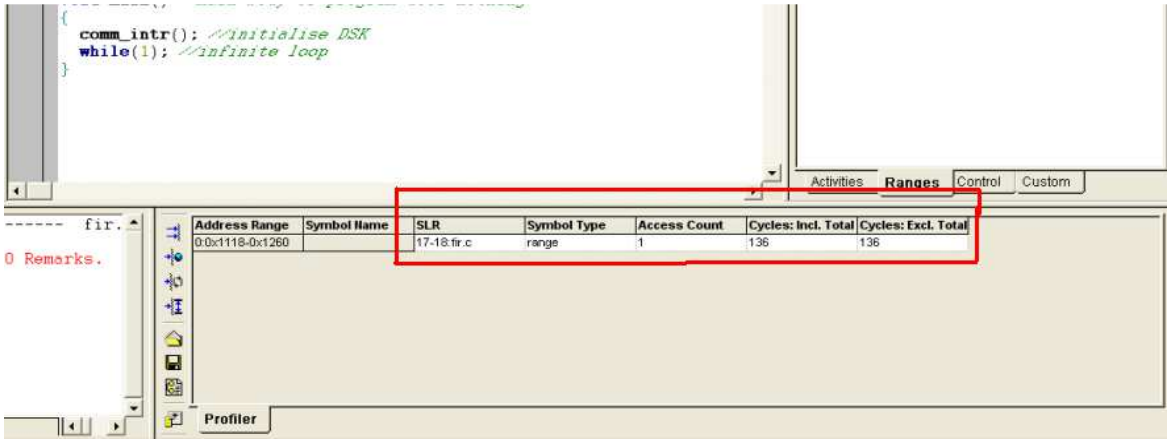

Figure 27 : Nombre de cycle 'exécution du boucle 'for' sans optimisation.

Donc on a pour un seul accès de la boucle, un nombre de **136 cycle d'horloge**, c'est-à-dire que cette boucle 'for' nécessite 136 cycle d'horloge pour exécuter à l'intérieur du DSP.
Cette analyse est faite sans optimisation.

b. Benchmarking (Profiling) avec optimisation :

Pour profiler le code du filtre avec optimisation il faut changer le compilateur par l'utilisation de `(-g -o3)`et pour le linker ne change rein `(-c -o)` comme le montre la figure suivante :

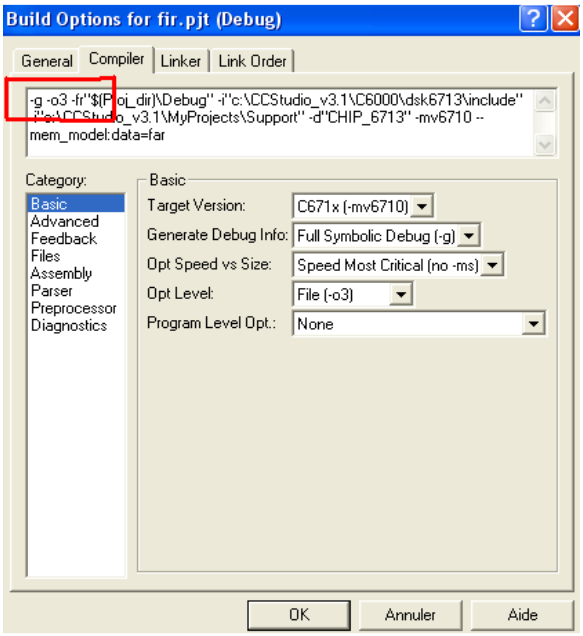

Figure 28: Configuration du compilateur avec optimisation.

Et après avoir effectué leprofiling avec optimisation on obtient le résultat suivant :

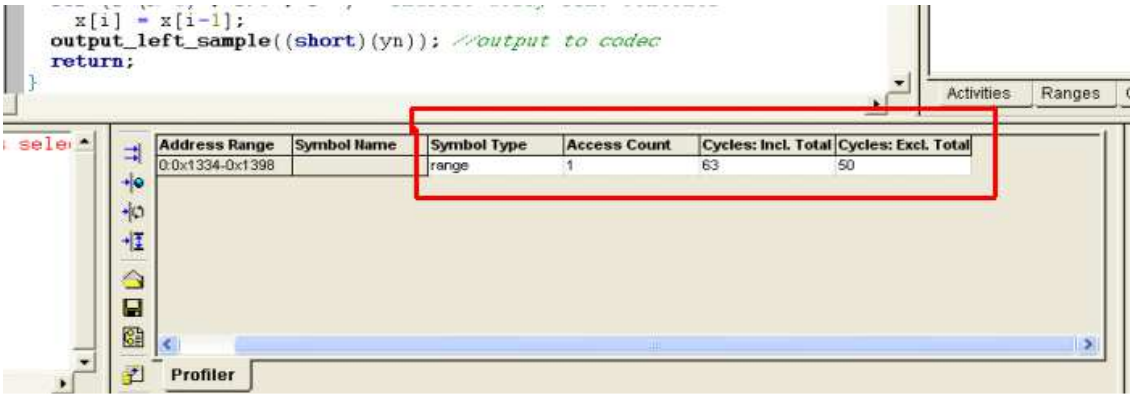

Figure 29: Nombre de cycle 'exécution du boucle 'for' avec optimisation.

Donc pour un seul accès de la boucle, un nombre de **63 cycle d'horloge**, c'est-à-dire que cette boucle 'for' nécessite 63 cycle d'horloge pour exécuter à l'intérieur du DSP.

Donc l'optimisation du code en C permet de réduire le nombre de cycle d'horloge par la moitie pour l'exécuter à l'intérieur du DSP.

c. Vitesse d'exécution :

La fréquence de fonctionnement du DSP est de 225 MHz donc une estimation maximale de la vitesse d'exécution est :

- DSP à 225 MHz avec 8 unités :

  225 000 000 * 8 = **1 800 000 000 opérations par seconde.**

Et selon le Datasheet du DSP TMSC320C6713 on trouve que si on a 1800 MIPS (1800 Million Instructions Per Second) ce résultat correspond au 1350 MFLOPS (1350 Million Floating Point Operations Per Second).

d. Consommation électrique :

Il est difficile d'estimer la consommation électrique du DSP, mais en général, le DSP consomme beaucoup plus pour des raisons de conception et de technologie.

e. Coût :

La plupart des processeurs DSP ne sont pas chèrset sont de l'ordre de 10$. Donc pour des applications de traitements de signalqui sont simples, un DSP est suffisant.

III. Comparatif :

La Figure 30 présente un comparatif qualitatif des solutions matérielles envisageables pour l'implémentation d'un système numérique de traitement du signal (filtre FIR). Chaque type de solution offre avantages et inconvénients.

Technologie	Performance	Surface	Consommation	Flexibilité	Temps de conception
FPGA	Très bonne	Grande	Moyenne	Bonne	Court
DSP	Bonne	Moyenne	Grande	Très Bonne	Moyenne

Figure 30 : Solutions matérielles pour systèmes DSP

1. Matériel ou Logiciel

Du point de vue de la performance de calcul brut, l'avantage va bien sûr aux solutions matérielles parallèles de type FPGA. On y voit que les solutions FPGA sont à la fois les plus rapides mais aussi celles qui présentent le plus faible coût *surface * temps*, due à l'excellente *densitéde calcul* de ces solutions. Par densité de calcul, on entend le nombre d'opérations réalisables ramené par unité de surface. De ce point de vue, la faiblesse des processeurs programmables est due au fait qu'ils utilisent des mémoires de grandes capacités pour mémoriser le code logiciel. Ces mémoires coûtent cher en terme de surface de silicium,

souvent plus cher que le processeur lui-même. A l'inverse, pour les FPGA, la reconfigurabilité est assurée par les bits de configuration des CLB et des interconnexions, qui forment une seule instruction très large pour tout le circuit. L'utilisation d'une seule instruction pour mémoriser le comportement du matériel fait que les FPGA représente de manière beaucoup plus dense l'état et la description d'un calcul donné.
L'autre avantage de ces solutions est qu'elles fonctionnent à une granularité bit, contrairement aux processeurs qui utilisent des mots (16, 32, 64 bits). Un processeur devant traiter des informations de largeur 1 bit utilisera des opérateurs de largeur mot, faisant chuter l'efficacité en terme de densité de calcul.

La meilleure utilisation du matériel dans les solutions câblées se traduit aussi au niveau de la consommation électrique. Ce paramètre, difficile à estimer pour un circuit complet, est cependant très fortement lié à la surface de silicium. Les FPGA plus denses en terme de surface et dont le taux d'utilisation des ressources est beaucoup plus élevé que pour les processeurs, ont donc des rendements performance / consommation bien meilleurs comme l'illustre la Figure 31 [5].

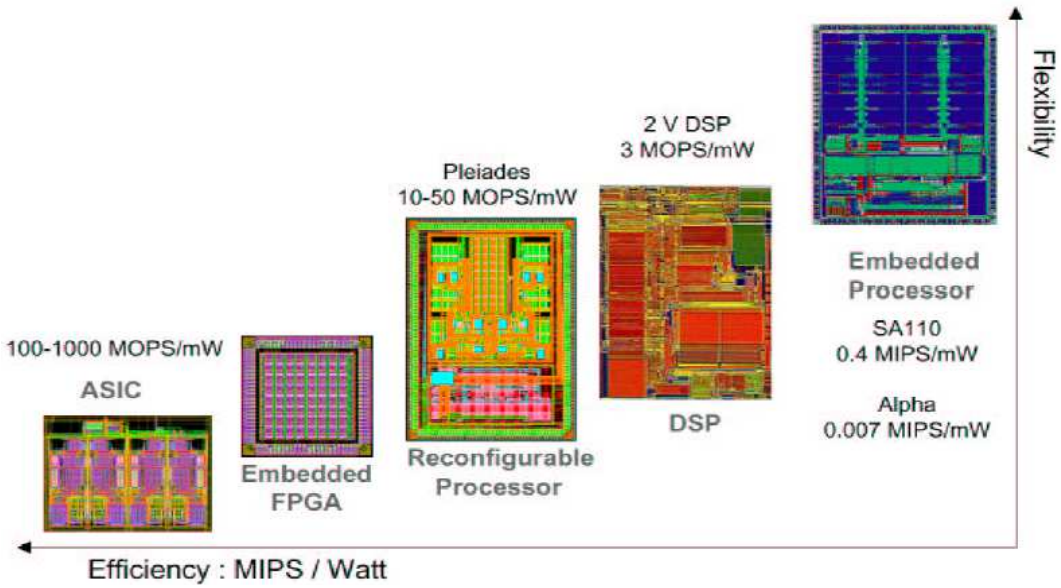

Figure 31 : Performance, consommation et flexibilité des différentes architectures matérielles

L'utilisation de processeurs programmables spécialisés en lieu et place des solutions purement câblés est pourtant de plus en plus fréquente, particulièrement dans le domaine des circuits dédiés aux applications DSP [6]. La complexité croissante des applications constitue une première explication. Dans le domaine de l'audio multimédia par exemple, les

simples systèmes stéréophoniques utilisés dans les années 80 ont laissé la place à des standards de codage plus performants : *Dolby AC-3*, *MPEG2*, etc. Ces applications sont trop complexes pour pouvoir faire l'objet d'implémentation 100% matérielles, et utilisent en général des systèmes mixtes matériel/logiciel.

2. FPGA ou DSP

Du strict point de vue de la performance de calcul, les microprocesseurs généraux n'ont rien à envier aux processeurs DSP, notamment grâce aux techniques matérielles très sophistiquées utilisés dans ces processeurs et absentes dans la plupart des processeurs DSP. A l'heure actuelle, seuls les DSP évolués de type VLIW peuvent rivaliser en termes de performance.

Pourtant, les DSP « classiques » intégrant des architectures mono-scalaires (comme l'*ADSP218x*) sont encore les plus nombreux et les plus utilisés en dépit de leurs performances moyennes. Les deux défauts rédhibitoires des microprocesseurs généraux du point de vue des systèmes embarqués sont leur coût et leur consommation électrique trop élevés.

La Figure 32 illustre ces considérations en comparant 2 types de processeurs DSP en termes de performance brut (MIPS), de tension d'alimentation, de puissance moyenne dissipée et de rapport Performance/Consommation : Un processeur DSP spécialement conçu pour la basse consommation (*TMS320C54x*), un processeur DSP VLIW principalement utilisé pour ses performances élevées (*TMS320C62x*).

Processeur	MIPS	Vdd	Pmoy	MIPS/Watt
TMSC54x	30-200	1.8V	460 mW	1500-3000
TMSC6x	1600	2.5V	2 W	800

Figure 32 : Consommation et performance par type de processeur

La première chose marquante est la consommation du *DSP TMSC6x* par rapport à TMSC54x. Même le *TMS320C62x*, pourtant l'un des DSP consommant le plus, dissipe vingt fois moins d'énergie pour une performance deux fois plus grande. Ces chiffres illustrent bien le fait que les processeurs DSP équipant les stations de travail sont conçus avant tout pour la performance au détriment du critère de consommation.

La puissance de ces processeurs les rend cependant capables d'exécuter la plupart des applications multimédia, évitant ainsi le surcoût lié à l'utilisation d'un circuit spécialisé. C'est d'ailleurs la raison d'être des extensions multimédia (*MMX* ou *AltiVec)* intégrées aux processeurs récents. Cependant, pour les applications les plus exigeantes comme les jeux ou les outils graphiques professionnels, ces extensions sont insuffisamment puissantes et les utilisateurs doivent avoir recours à des cartes d'extension intégrant la plupart du temps des processeurs DSP ou des FPGA.

La flexibilité, le faible coût, les architectures spécialisées en traitement du signal et une consommation électrique raisonnable sont les principaux arguments qui plaident en faveur de l'utilisation de processeurs DSP pour les applications de traitement du signal. Parmi les différentes solutions matérielles envisageables, les processeurs DSP sont ceux qui présentent le compromis Performance/Consommation/Temps de développement/Coût le plus équilibré [7]. Ils ne sont les meilleurs dans aucune des catégories, mais ne sont mauvais nulle part, et c'est précisément ce qui fait leur succès.

Conclusion

L'étude et l'analyse des différents paramètres permettent de bien comprendre quelle est l'architecture la mieux adaptée pour un algorithme de traitement de signal. Donc pour une application qui ne coûte pas chère et qui ne nécessite pas de gros calcul, un DSP est largement suffisant. Mais pour les applications gourmandes en traitement de signal qui permettent une longue durée des batteries et une grande vitesse de calcul, un FPGA est le meilleur choix parmi les autres cibles.

Conclusion Générale

Dans le passé, l'utilisation de processeurs de signaux numériques est omniprésente, mais avec les besoins de nombreuses applications dépassant les capacités de traitement des processeurs de signaux numériques (mesurée en millions d'instructions par seconde (MIPS)), l'utilisation des FPGA est en croissance rapide. Actuellement, la principale raison pour laquelle la plupart des ingénieurs choisissent d'utiliser un FPGA par rapport à l'un des processeurs de signaux numériques est pilotée par les exigences MIPS de l'application et la consommation d'énergie. Ainsi, la comparaison entre le processeur de signaux numériques et le FPGA met l'accent sur la comparaison MIPS, et la consommation qui ne sont pas les seuls avantages d'un FPGA.

Les FPGA sont bien adaptées au traitement numérique du signal. Dans cette analyse, cela a été montré pour le cas particulier des filtres. Les dispositifs ont suffisamment de puissance pour permettre des traitements complexes en parallèle. Les outils de développement rendent le temps de développement raisonnable et la simulation permet un bon dimensionnement et une bonne analyse des performances. Les différents outils de synthèse n'ont pas tous les mêmes performances, parce qu'ils n'effectuent pas le même type d'optimisations.

Ce Projet de Fin d'Etudes nous a permis de nous familiariser avec les deux KIT de développements de Texas Instruments, le TSW6011 EVM qui est basé sur un FPGA d'Altera, et le C6713 DSK qui intègre un processeur de traitements numériques du signal.

L'utilisation des outils MATLABnous a servis àfaciliter la programmation des filtres par la génération de son code en VHDL avant de l'implémenter sur le FPGA et aussi la génération de ses coefficients en C utilisés pour paramétrer le filtre au niveau du DSP.

Aprèsl'implémentationdu programme sur le FPGA et le DSP, nous avons fait une étude comparative entre eux en termes de complexité, de vitesse d'exécution, de consommation d'énergie et du coût afin de faire le choix de la plateforme à utiliser pour des applications de traitement de signal sur les processeurs des systèmes embarqués.

Annexe 1 : Description des différents blocs sur la carte TSW6011EVM :

TRF3711 :

Le TRF3711 comprend des filtres programmables en bande de base, un DC réglable pour la correction de décalage, et des amplificateurs tampons pour piloter directement l'ADC. Ce circuit est conçu pour un fonctionnement avec WCDMA, WiMAX et LTE modulation ainsi que d'autres schémas de modulation de signaux à haute bande passante. Les amplificateurs à gain programmable sur puce permettent le réglage du niveau de signal de sortie sans le besoin de dispositifs externes à gain variable (atténuation). Le TRF3711 intègre des filtres passe-bas en bande de base programmables qui -atténuent les interférences à proximité, ce qui élimine la nécessité d'un filtre à bande de base externe. Le filtre à l'intérieur du TRF3711 est un Butterworth huitième ordre avec 0,7 MHz à 15 MHz (résolution 8 bits) bande passante programmable qui couvre 1,4 MHz à 30 MHz de bande passante du signal complexe (BW).

Les autres caractéristiques du TRF371125 comprennent:

- Gamme de fréquences jusqu'à 4 GHz.
- Amplificateur intégré à gain programmable en bande de base.
- filtre à bande de base programmable sur puce.
- Haute out-of-bandIP3 : 24 dBm à 2400 MHz.
- Haute out-of-band IP2: 60 dBm à 2400 MHz.
- La technologie silicium-germanium.

ADS5282 :

L'ADS528x est une famille de haute performance, de faible puissance, octal canal ADC qui sont disponibles soit dans un boîtier de 9 mm x 9 mm, QFN ou un paquet HTQFP-80, avec signalisation différentielle à basse tension sérialisé (LVDS) pour les sorties et une grande variété de fonctions programmables[3].

L'ADS5282 comprend également:

- résolution 12 bits avec 65-MSPS vitesse d'échantillonnage.
- entrée analogique pleine échelle: 2 VPP
- gain numérique programmable: 0 à 12 dB
- Une sortie série DDR LVDS.

Les applications typiques pour les dispositifs ADS528x sont l'imagerie médicale, l'infrastructure des stations de base sans fil et de l'instrumentation de mesure.

FPGA :

Le FPGA exécute un traitement de signal en bande de base tel que la mesure numérique de la puissance, la correction IQ, commande de gain numérique, 2x interpolation, la compensation du décalage en courant continu, et de mélange en quadrature. 12-bits de la sortie série ADS5282 avec un débit de données de 61,44 MHz est livré à une interface Serial-à-DESERIAL pour convertir le format des données série à 12 bits de données parallèles dans le FPGA. Pour une utilisation DSP, 12 bits d'entrée sont simplement étendus à une version 16 bits des données de format de complément à deux, et puis livrés au module de correction d'IQ en FPGA. La figure suivante montre le schéma de principe de FPGA[3].

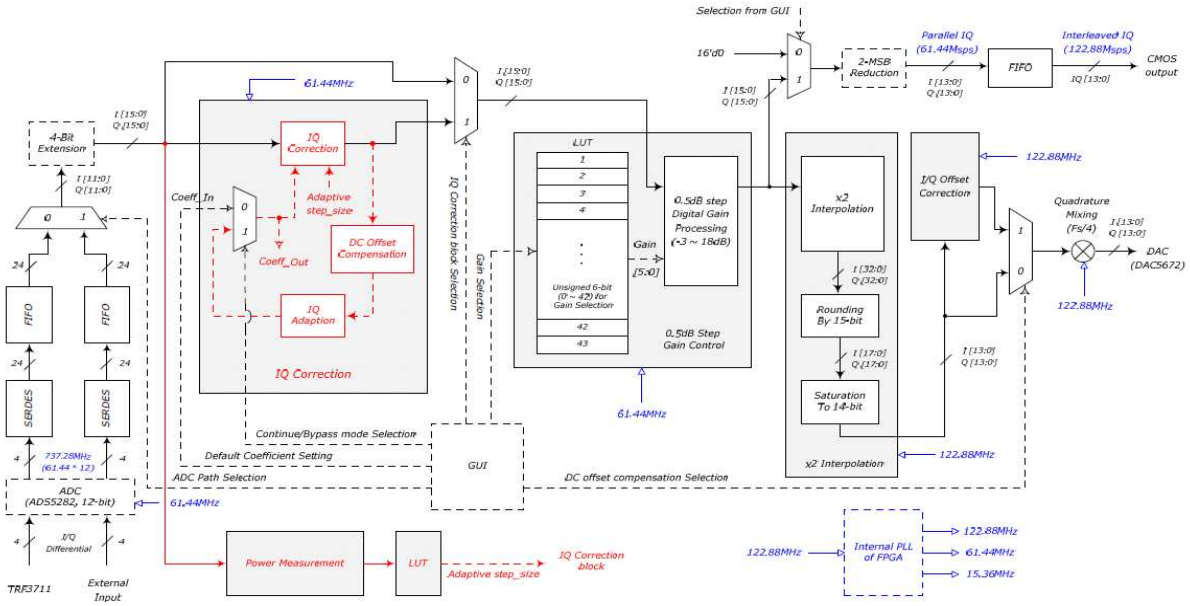

Figure 33: Schéma de principe du programme original.

Après un traitement de correction IQ, le gain des signaux I et Q peut être augmentée jusqu'à 18 dB sur la base de 0,5 dB. L'amélioration de la gamme dynamique du signal produit une

amélioration de la qualité du signal en bande de base pour des caractéristiques telles que le rapport signal-sur-bruit (SNR), le taux d'erreur sur les bits (BER), ou l'amplitude du vecteur d'erreur. Interpolation par un facteur de 2, DC offset compensation, et les blocs de mélange en quadrature sont suivis par le contrôleur de gain numérique pour transférer les données vers l'interface du DAC[3].

CDCE62005 :

Le CDCE62005 a 5/10 générateurs d'horloge de sortie et comprend deux VCO intégrés. Les autres caractéristiques de ce circuit sont les suivantes:

• Synthétiseur de fréquence PLL avec / VCO et le filtre de boucle partiellement intégré.

• Sorties entièrement configurables, y compris la fréquence, le format de sortie.

• EEPROM intégrée détermine la configuration du circuit au démarrage.

• blocs de sorties universelles en charge jusqu'à cinq différentiel, 10 asymétrique, ou des combinaisons de différentiel ou asymétrique.

• Faible bruit de phase de sortie: -130 dBc / Hz à 1 MHz offset, FC = 491.52 MHz.

Annexe 2 : Description du DSP TMS320C6713 :

Le TMS320C6713 de la carte DSK est un processeur à virgule flottante basée sur l'architecture VLIW, il contient une mémoire interne comprend une architecture de cache à deux niveaux avec 4 Ko de cache de niveau 1 du programme (L1P), 4 Ko de niveau 1 de cache de données (L1D), et 256 Ko de mémoire de niveau 2 partagé entre le programme et de l'espace de données. Il dispose d'une interface directe pour les deux mémoires synchrones (SDRAM et SBSRAM) et les mémoires asynchrones (SRAM et EPROM). Mémoire synchrone nécessite d'horloge, mais offre un compromis entre une mémoire statique SRAM et une mémoire dynamique DRAM, avec SRAM est plus rapide mais plus cher que la DRAM[2].

Périphériques sur puce comprennent deux McBSPs, deux timers, une interface de port d'hôte (HPI), et un EMIF 32-bit. Il nécessite 3,3 V pour I / O et 1,26 V pour le noyau (interne). Bus internes comprennent un bus d'adresse programme 32 bits, un bus de données programme 256-bit pour accueillir huit instructions 32 bits, deux bus de données d'adresses 32 bits, deux bus de données 64 bits et deux bus de données stockent 64 bits. Avec un bus d'adresse de 32 bits dont quatre espaces extérieurs de mémoire: ce0, CE1, CE2 et CE3 et l'espace total de la mémoire est de 2^{32} = 4 Go.

La figure 34 montre un schéma fonctionnel du processeur C6713 inclus avec CCS.

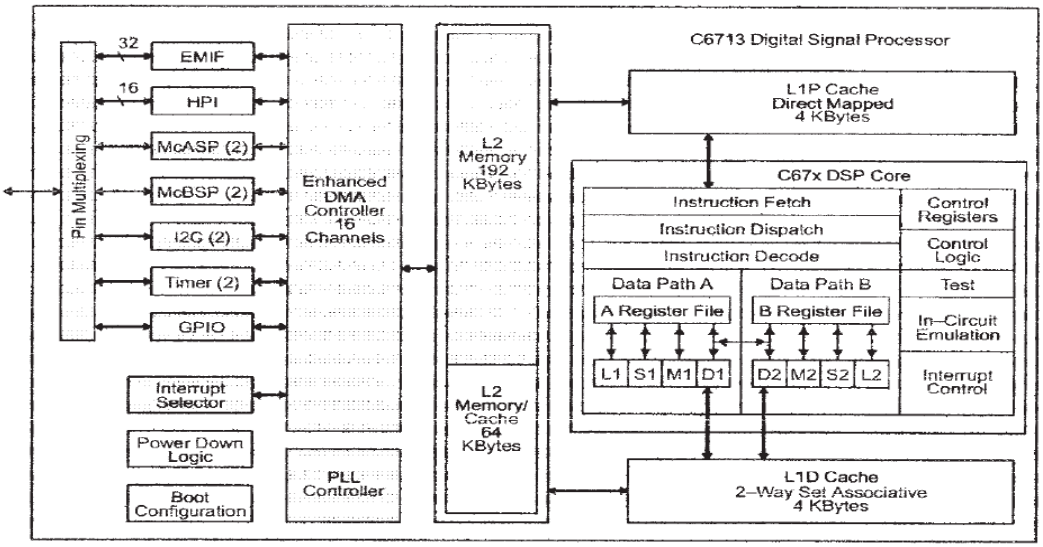

Figure 34 : Les blocs diagrammede TMS320C6713.

Les banques de mémoire indépendantes sur la C6x permettent de deux accès mémoire dans un cycle d'instruction. Deux banques de mémoire indépendantes sont accessibles via deux bus indépendants. La mémoire interne est organisée dans les banques de mémoire, deux charges ou deux stocks d'instructions peuvent être exécutées en parallèle. Aucun résultat de conflits si les données accédées se trouvent dans les différentes banques de mémoire. Les bus sont séparés pour le programme, les données, et l'accès direct à la mémoire (DMA) permet de récupérer les données dans un programme pour lire et écrire. Avec les données et les instructions résidant dans des espaces mémoire séparés donc on peut accéder simultanés à la mémoire.

Le C6x dispose d'un espace mémoire adressable par octet. La mémoire interne est organisé mémoire de programme distinct et un autre espace de mémoire de données, avec deux ports internes de 32 bits pour accéder à la mémoire interne.

Le C6713 sur la carte DSK comprend 264KB de mémoire interne, qui commence à 0x00000000, et 16 Mo de SDRAM externe, repérés à travers CE0 à partir de 0x80000000. La carte DSK comprend également 512 Ko de mémoire flash (256 kB facilement disponibles pour l'utilisateur), tracée à travers CE1 à partir de 0x90000000.

La figure 35 présente la configuration de la mémoire interne L2.

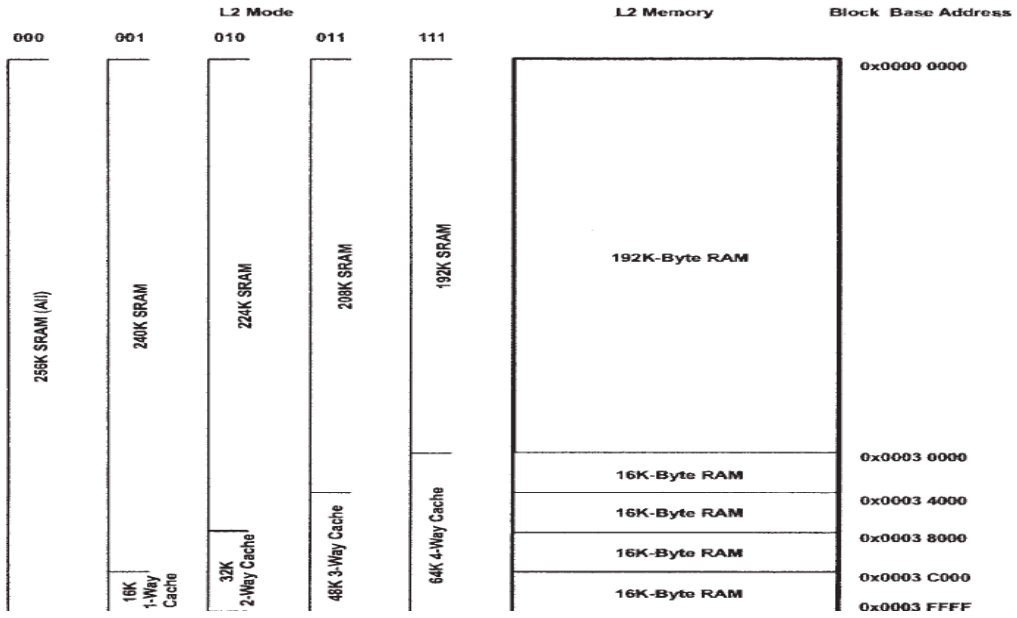

Figure 35: Répartition de la mémoire interne L2.

La carte DSK fonctionne à 225MHz, on peut idéalement atteindre deux se multiplie et s'accumule par cycle, pour un total de 450 millions multiplie et accumule (MAC) par seconde. Avec six parmi huit unités fonctionnelles dans la figure 3.1 (pas les unités .D décrites ci-dessus) capable de gérer des opérations en virgule flottante, il est possible d'effectuer 1350000000 opérations en virgule flottante par seconde (MFLOPS). Fonctionnant à 225MHz, cela se traduit par 1,8 milliards d'instructions par seconde (MIPS) avec un temps de cycle de l'enseignement 4,44 ns.

Bibliographie

[1] **L. Bossuet**, *Exploration de l'espace de conception des architectures reconfigurables*, Thèse de doctorat, Université de Bretagne Sud, Lorient, septembre 2004.

[2] **RULPH CHASSAING.**, Digital Signal Processing and Applications whit the C6713 and C6416 DSK, WILEY-INTERSCIENCE A JOHN WILEY & SONS, INC., PUBLICATION.

[3] **KYUNG-WAN NAM**., TSW6011: A Direct Downconversion System with IQ Correction and Impact on EVM, Wireless Infrastructure Radio Products, Texas Instruments, SLWA065– July 2011, http://www.ti.com/lit/an/slwa065/slwa065.pdf

[4] **HENNESSY J., PATTERSON D.**, *Architecture des Ordinateurs, une approche quantitative*, Deuxième édition, traduction de Daniel Etiemble, Thomson Publishing

[5] **SENTIEYS O**., Processeurs de traitement du signal : Etat de l'art, Critères de choix, Perspectives, ENSSAT – Université de Rennes 1, http://archi.enssat.fr

[6] **PAULIN P., LIEM C., CORNERO M., NACABAL F., GOOSSENS G.**, *Embedded Software in Real-Time Signal Processing Systems: Application and Architecture Trends*, Proc. of the IEEE, Vol 85, March 1997

[7] **RESTLE R**., Choosing between DSPs, FPGAs, mPs and ASICs to implement digital signal processing, ICSPAT2000, Dallas, USA, 2000

[8]http://fr.wikipedia.org/wiki/Filtre_%C3%A0_r%C3%A9ponse_impulsionnelle_finie

[9]https://fr.wikipedia.org/wiki/Filtre_num%C3%A9rique

[10]http://archives.limsi.fr/Individu/tuttitom/mpANFiltrage.pdf

[11] http://www.altera.com/devices/fpga/cyclone3/overview/architecture/cy3-cost-optimized.html