Synthèse architecturale analogique/numérique appliquée aux systèmes sur puce dans un contexte radio logicielle

Ludovic Barrandon

N° d'ordre : 3329

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE RENNES 1

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Électronique

par

Ludovic BARRANDON

Équipe d'accueil : Institut d'Électronique et de Télécommunications de Rennes

École doctorale : MATISSE

Composante universitaire : Structure et Propriétés de la Matière

Synthèse architecturale analogique / numérique appliquée aux systèmes sur puce dans un contexte radio logicielle

Soutenue le 8 Décembre 2005 devant la commission d'examen

COMPOSITION DU JURY

Président	Pascal FOUILLAT	Professeur, ENSEIRB, Université de Bordeaux 1
Rapporteurs	Jean-François DIOURIS	Professeur, Ecole Polytechnique de l'Université de Nantes
	Patrick GARDA	Professeur, Université Pierre et Marie Curie, Paris VI
Examinateurs	Ali AHAITOUF	Professeur, Université Sidi Mohammed Ben Abdallah, Fès, Maroc
	Samuel CRAND	Maître de Conférences, Université de Rennes 1
Directeur de thèse	Dominique HOUZET	Maître de Conférences HDR, INSA de Rennes

Le commencement de toutes les sciences,
c'est l'étonnement de ce que les choses sont ce qu'elles sont.

Aristote, Métaphysique.

Remerciements

Je remercie Monsieur le Professeur Daniel THOUROUDE, Directeur de l'Institut d'Electronique et de Télécommunications de Rennes de m'avoir accueilli au sein de son laboratoire.

Je tiens à exprimer ma reconnaissance à Monsieur Dominique HOUZET, Maître de Conférences à l'INSA de Rennes, directeur de cette thèse, et à Monsieur Samuel CRAND, Maître de Conférences à l'Université de Rennes 1, co-encadrant de ces travaux, tout d'abord pour m'avoir proposé ce sujet de recherche, mais surtout pour leur encadrement indéfectible et toute la confiance qu'ils ont su m'accorder tout au long de ces trois années.

Je remercie vivement Monsieur Jean-François DIOURIS, Professeur à l'Ecole Polytechnique de l'Université de Nantes, et Monsieur Patrick GARDA, Professeur à l'Université Pierre et Marie Curie (Paris VI), qui m'ont fait l'honneur d'accepter d'être les rapporteurs de ce mémoire.

Mes remerciements vont également à Monsieur Ali AHAITOUF, Professeur à l'Université Sidi Mohammed Ben Abdallah de Fès, et Monsieur Pascal FOUILLAT, Professeur à l'Université Bordeaux I, pour leur participation au jury de cette thèse.

Je ne peux malheureusement me permettre de nommer tous ceux qui ont compté pour moi et qui m'ont aidé durant ces dernières années. Je n'oublierai jamais le soutien que m'ont apporté tous mes collègues permanents, doctorants, post-doctorants, stagiaires, toutes équipes confondues, actuels et de passage, tous mes amis de Rennes et d'ailleurs et, bien sûr, toute ma famille. Merci aussi à tous les artistes qui ont composé et enregistré les milliers d'heures de musique qui ont constitué la bande-son de mes journées (et de mes nuits !) passées à l'IETR.

Sommaire

Introduction

Les progrès constants des technologies de l'électronique ouvrent continuellement des opportunités de développement de nouveaux produits toujours plus performants et fonctionnels. Les lecteurs de DVD, les écrans plats, la photographie numérique, les systèmes de télécommunications sans fil sont des exemples récents représentatifs de cette évolution.
Afin de satisfaire la demande, il est nécessaire de proposer des systèmes offrant des fonctionnalités de plus en plus variées. Les architectures mises en œuvre croissent donc en complexité et peuvent intégrer des modules hétérogènes (numériques, analogiques, mixtes, logiciels). Ce constat mène à reconsidérer en permanence les méthodes de conception avec pour objectif principal la réduction des délais de mise sur le marché. On assiste alors à l'émergence de techniques de conception modulaires basées sur le principe de propriété intellectuelle (IP pour *Intellectual Property*). Son rôle est de favoriser la réutilisabilité d'éléments matériels et/ou logiciels. Cette méthodologie, couplée aux progrès des technologies de la microélectronique, conduit à la possibilité de concevoir des systèmes complets sur une seule puce. Les systèmes sur puce (ou SoC pour *System-on-Chip*) bénéficient d'une approche de conception descendante inhérente au domaine de l'électronique numérique. Dans le cadre plus général des SoC mixtes, intégrant à la fois des fonctions numériques et analogiques (SoC-AMS), apparaît la nécessité d'unifier les méthodes de conception numériques et mixtes afin de converger vers une démarche descendante.

Dans le domaine des télécommunications sans fil, le passage de la deuxième à la troisième génération a été encouragé par la forte croissance du marché et la demande d'applications multimédias. Il existe donc de nombreux enjeux techniques guidés par des motivations économiques.
Le concept de radio logicielle (RL) s'inscrit dans la perspective de proposer des objets communicants nomades, adaptatifs et flexibles. Ils sont destinés à s'insérer dans ce contexte évolutif, alliant les différentes technologies des réseaux locaux et cellulaires. La radio logicielle vise essentiellement à introduire la reconfigurabilité dans les terminaux de télécommunication mobiles pour répondre à une diversité de standards et de fonctionnalités.

Dans ce cadre, le développement de mobiles multistandards repose d'une part sur les possibilités offertes par les composants de traitement du signal numériques programmables et d'autre part sur les avancées méthodologiques évoquées au début de cette introduction.

L'objet de cette thèse se situe à la confluence de ces deux thématiques : les notions de SoC et de radio logicielle posent les bases de l'étude développée dans ce manuscrit et sont abordées dans le premier chapitre. Celui-ci oriente l'exploration architecturale des solutions candidates pouvant répondre aux contraintes de la radio logicielle. Il se focalise plus particulièrement sur un espace de conception incluant une partie de la chaîne de réception radio à savoir la conversion analogique–numérique (CAN) et la tête numérique associée. L'ensemble constitue la tête de réception mixte (TRM). Les solutions architecturales, discutées en fonction du partitionnement analogique–numérique, sont essentiellement dépendantes des performances des CAN. Ce partitionnement conduit à la notion de radio logicielle restreinte (RLR), solution alternative à la radio logicielle dite idéale.
Un exemple illustratif de TRM non reconfigurable est proposé dans le but d'identifier les besoins méthodologiques. Il permet de présenter les outils employés et donne une vue d'ensemble de la problématique de ce travail.

Les différents éléments de la TRM doivent être sélectionnés puis dimensionnés afin d'optimiser l'application. Les principales variables de cette optimisation sont la quantité de ressources utilisées, la fréquence de fonctionnement et les performances spectrales.
Dans un premier temps, cette exploration architecturale est scindée en deux parties. L'une concerne la partie mixte de la tête de réception, c'est-à-dire la conversion A/N. La seconde traite des modules de traitement numérique du signal. Ces deux voies de recherche représentent le cœur des travaux présentés dans ce mémoire et donnent lieu aux deuxième et troisième chapitres. Elles reposent sur l'idée commune d'un découpage modulaire des fonctions.
L'objectif est de proposer des outils d'aide au dimensionnement dans la perspective d'une mise à disposition d'un flot de conception systématique et automatisé. Cette démarche a pour but de synthétiser une architecture de TRM à partir de ses spécifications.

Le chapitre 2 propose un flot d'aide à la sélection et au dimensionnement de CAN. Leurs performances sont déterminantes puisqu'elles limitent le partitionnement analogique–numérique de la TRM.
Les composants numériques programmables (DSP, FPGA) offrent naturellement plus de flexibilité que les circuits analogiques. Partant de ce constat, il est essentiel d'effectuer la plus grande partie de traitement possible en numérique afin de satisfaire au principe d'adaptabilité de la RL. Les convertisseurs A/N et N/A doivent donc être situés au plus près de l'antenne.
Dans ce contexte, le chapitre 2 propose un flot d'aide à la sélection et au dimensionnement de CAN, basé sur une méthode automatisée permettant de comparer rapidement les solutions candidates. Elle repose sur un découpage modulaire du CAN et utilise une

fonction de mérite (FDM). Cette métrique unique reflète la prise en compte de plusieurs paramètres définissant ainsi un compromis. La FDM est compatible avec la définition d'un système à un niveau d'abstraction élevé, c'est-à-dire sans relation directe avec une éventuelle technologie de fabrication.

Concernant les traitements numériques associés à la conversion A/N dans la TRM, le chapitre 3 développe une méthodologie de synthèse systématique. Elle s'appuie sur le même principe de découpage modulaire des blocs de traitement. Cette partie présente les différents systèmes, essentiellement des filtres multifréquences, permettant de prendre en charge les traitements de la tête de réception numérique (TRN). Un outil de dimensionnement de TRN, associé à une génération automatique de code VHDL permettant la synthèse complète de l'application, est présenté.

Les deux branches du flot de conception sont mutualisées dans le quatrième chapitre. Cette partie, conjuguant les aspects conversion et traitements numériques, a pour rôle de spécifier les différentes fonctions constitutives de la TRM. Ce travail se base sur une étude de cas axée sur les standards GSM et UMTS. Une réalisation autour d'une plateforme de prototypage utilisant un protocole d'interface PCI regroupe les principes méthodologiques et applicatifs des précédents chapitres.

Une conclusion générale résume les principales contributions de ce travail portant principalement sur l'optimisation, l'implantation de systèmes radio logicielle et sur la mise en œuvre des méthodologies de conception associées. Différentes perspectives de développement sont également présentées.

Chapitre 1

Contexte de l'étude

Ce chapitre dresse un bilan des concepts associés à l'évolution actuelle des méthodologies de conception électronique. Ces notions sont appliquées au domaine des télécommunications et en particulier à la radio logicielle restreinte. Les outils employés ainsi qu'une vue d'ensemble des développements nécessaires sont présentés à travers un exemple.

1.1 Contexte général

La croissance du marché du semi-conducteur, la diversification de ses besoins et l'évolution des technologies nécessitent d'améliorer sans cesse l'efficacité des méthodes de conception. On assiste à l'émergence des concepts de propriété intellectuelle et de systèmes sur puce qui proposent de favoriser la réutilisabilité de fonctions organisées en bibliothèques. Ces points sont les sujets de la première partie de ce chapitre.

1.1.1 Evolution du marché de l'électronique et de ses besoins

1.1.1.1 Croissance du marché

Des statistiques récentes indiquent un accroissement important des besoins en matériels électroniques au niveau mondial dans l'ensemble des secteurs d'activité [Oll04, Sit]. Malgré des fluctuations rapides (forte croissance en 2000, crise en 2001/2002), le marché a plus que doublé durant ces dix dernières années (Figure 1.1).

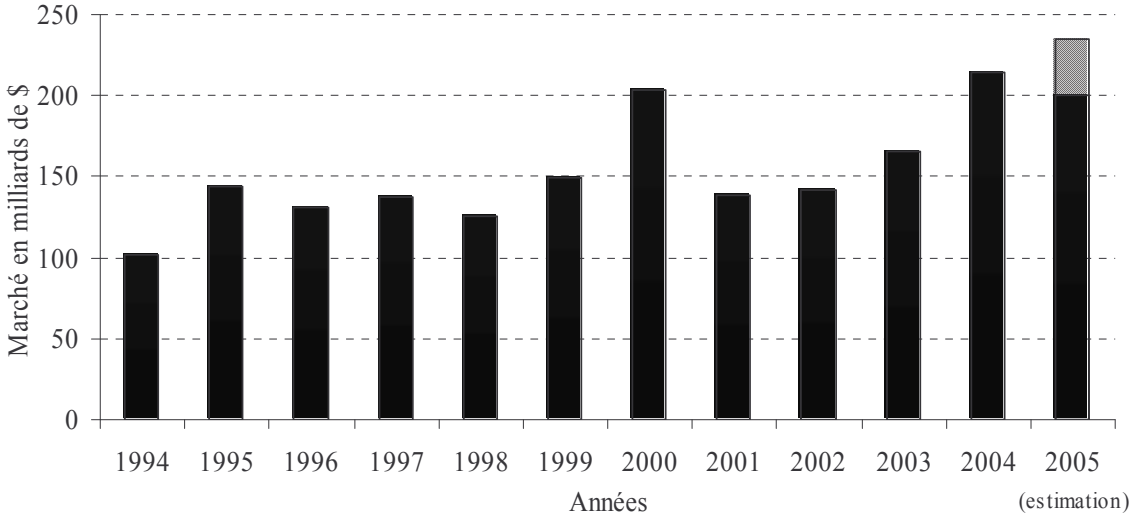


Figure 1.1 Evolution du marché mondial de l'électronique par année en milliards de dollars.

Chacun des grands secteurs apporte sa contribution à cette croissance globale (Tableau 1.1). Ainsi, l'informatique bénéficie d'un renouvellement avec l'arrivée de technologies nouvelles (wifi, DVD-RW…) tout comme les applications grand public qui profitent de l'attrait en matière de multimédia, de DVD et d'écrans plats. L'automobile quant à elle propose de nouveaux systèmes d'aide à la conduite, de sécurité et de multimédia. L'industrie voit un accroissement des demandes en matière d'applications émergentes : analyse et recherche médicales (*lab-on-chip*), sans contact, biométrie, nouveaux passeports… Le domaine des télécommunications représente le secteur contribuant le plus à

la croissance du marché grâce à l'arrivée de la génération 3G et de nouveaux services entraînant un taux de remplacement important (40% par an).

Domaine d'application	Informatique	Automobile	Grand public	Industrie	Télécoms
Croissance	25%	18%	25%	14%	45%

Tableau 1.1. Taux de croissance par domaine d'application entre 2003 et 2004.

A cette demande, dont le volume et la diversité augmentent constamment, s'associent des enjeux économiques et une concurrence imposant une réduction des délais de mise sur le marché (*time-to-market*). Cette pression s'exerce non seulement sur la production mais également en amont sur la conception des systèmes électroniques.

1.1.1.2 Enjeux pour les concepteurs

Pour la majorité des applications, la tendance consiste à développer des systèmes consommant moins et compacts (systèmes embarqués et nomades) tout en assurant plus de fonctionnalités pour un même objet (multimédia, Internet, photo, vidéo…).
Cette évolution est très nette dans le domaine de l'informatique dont l'histoire s'étend sur les quatre à cinq dernières décennies [Lil03]. Les premières machines étaient des calculateurs encombrants offrant peu de possibilités. Au cours du temps, les progrès de l'électronique ont permis de mettre au point des ordinateurs parfaitement adaptés aux besoins de chaque utilisateur. Du point de vue matériel, l'architecture se compose de modules communicants, dédiés et interchangeables (cartes, mémoires, processeur…). Du point de vue applicatif, l'emploi d'un même ordinateur peut être complètement modifié en fonction du programme utilisé.
Dès à présent, de nombreux domaines de l'électronique font appel à ces deux concepts de modularité (matériel) et de programmation (logiciel). Combinés à l'augmentation constante des possibilités d'intégration, ils permettent de concrétiser l'idée de systèmes sur puce (ou de systèmes monopuces) plus souvent nommés SoC pour *System on Chip*. Les SoC sont des systèmes intégrant sur un même substrat l'ensemble des modules nécessaires à une application donnée. Ils sont donc constitués de blocs hétérogènes associés à un réseau d'interconnexion comme l'illustre la Figure 1.2.

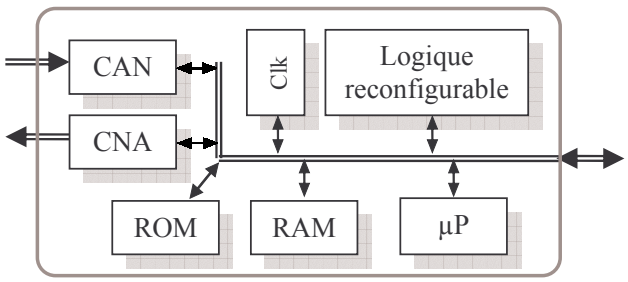


Figure 1.2. Le SoC : intégration de fonctions hétérogènes.

1.1.2 Evolution des technologies de l'électronique et des méthodologies de conception : le défi des SoC

1.1.2.1 Progrès technologiques

Gordon Moore, cofondateur de la société Intel, avait affirmé en 1965 que le nombre de transistors par circuit de même taille allait doubler tous les 18 mois. 1972 est l'année de la mise au point du 1er microprocesseur, le 4004, avec 2300 transistors. Basée sur ces données, la loi de Moore (équation (1.1)) permet d'évaluer les capacités d'intégration en nombre *n* de transistors pour une année *a* :

$$n = 2300.2^{\left[(a-1972).\frac{2}{3}\right]} \tag{1.1}$$

Cette capacité d'intégration peut être illustrée en observant le nombre de transistors utilisés par les processeurs Intel depuis le 4004 (Figure 1.3).

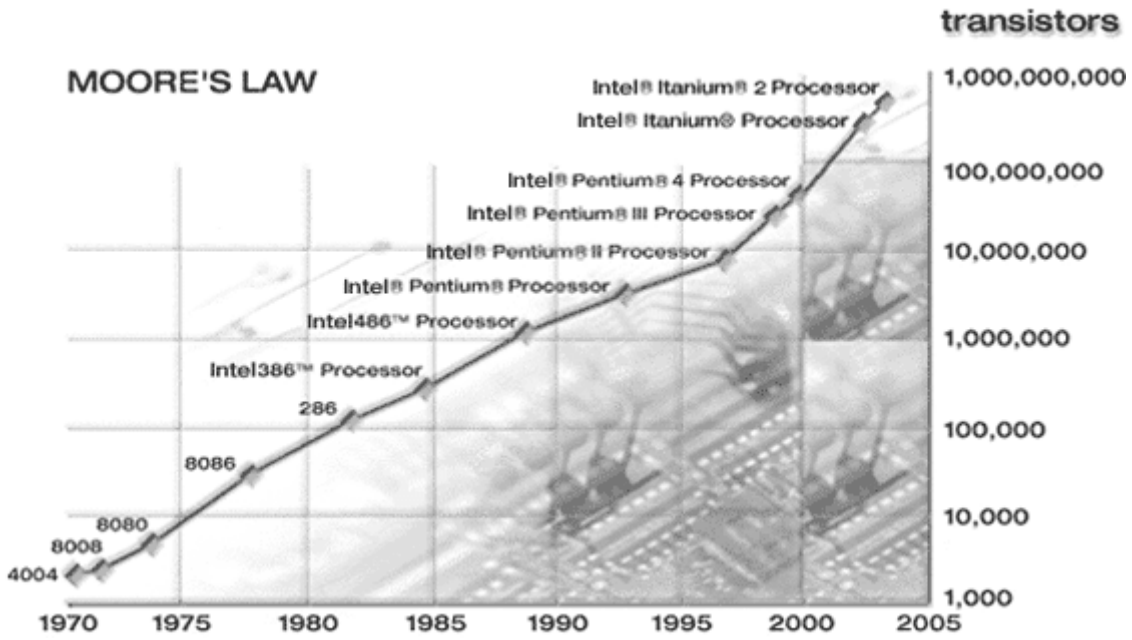


Figure 1.3. Exemple illustrant l'évolution des capacités d'intégration (source : [Int]).

Ce constat se base sur les progrès technologiques dont différents paramètres sont présentés sur la Figure 1.4. Alors que la taille minimale du canal des transistors diminue, la surface des plaquettes de semi-conducteur (*wafer*) et des puces s'accroît. Les procédés permettent également de concevoir des systèmes de plus en plus complexes en augmentant le nombre de couches d'interconnexion.

L'industrie du semi-conducteur utilise aujourd'hui la génération de transistors 90 nm sur des wafers mesurant jusqu'à 300 mm de diamètre. Nous assistons donc à une diminution des coûts par fonction et une augmentation de la productivité.

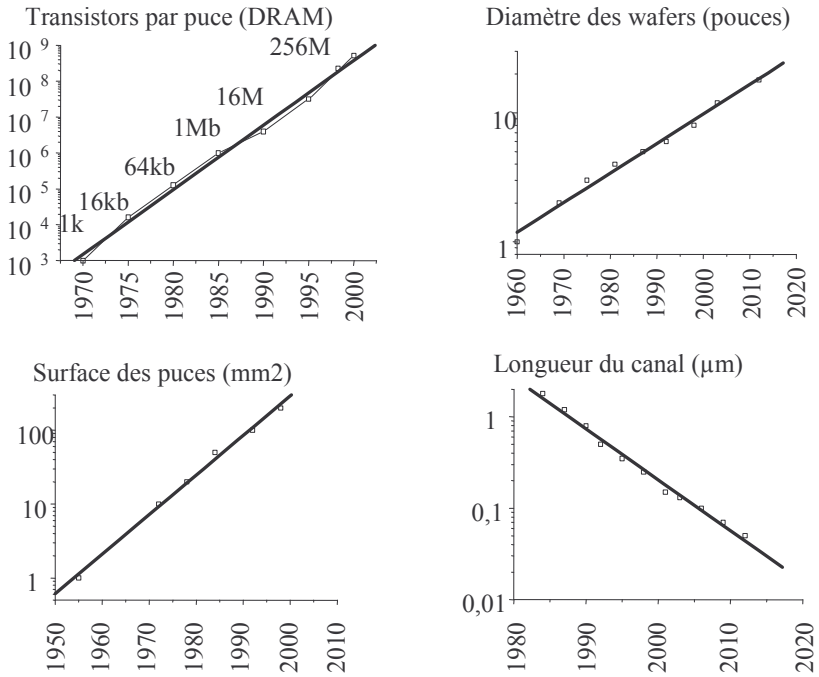


Figure 1.4. Evolution des technologies de l'électronique : quelques paramètres.

Afin de bénéficier de ces progrès technologiques, il paraît indispensable de renouveler en permanence la façon de concevoir les systèmes. La Figure 1.5 (source [Ben04]) représente l'écart croissant entre les progrès de conception et les possibilités technologiques. Elle met également en évidence les besoins algorithmiques des systèmes de communications : la loi de Shannon prévoit un doublement de leurs performances tous les 8,5 mois [Kar03]. Il existe donc un besoin important concernant le développement de méthodes et d'outils permettant de profiter des progrès des technologies et de bénéficier ainsi au maximum du potentiel d'intégration.

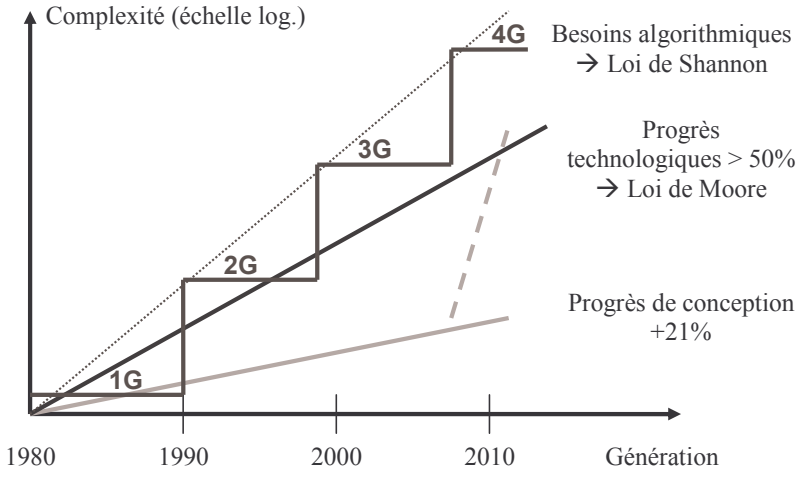


Figure 1.5. Comparaison relative entre les progrès de conception et les progrès technologiques.

1.1.2.2 Systèmes sur puce

Dans ce contexte, les SoC sont destinés à diminuer les coûts de production des systèmes tout en augmentant leurs performances (consommation, encombrement…). Le principe est d'intégrer sur un même substrat un ensemble de sous-systèmes logiciels, numériques, analogiques et mixtes. Techniquement réalisables, le développement de tels produits nécessite d'adapter les flots de conception et de mettre au point des outils adaptés. Afin de tirer parti du potentiel des SoC, il est nécessaire de mettre en œuvre des méthodologies innovantes [Cha99]. Cette évolution peut être comparée à l'émergence des circuits intégrés spécifiques (ASIC pour *Application-Specific Integrated Circuit*) à la fin des années 1980, reposant principalement sur la mise en place de bibliothèques de cellules de base standards afin de favoriser leur réutilisation. Les ASIC représentent le passage d'une conception au niveau transistor vers une conception au niveau porte logique. Cette remise en question du mode de conception a donné naissance à une nouvelle organisation des flots de développement en introduisant un niveau d'abstraction nouveau [Cha99].

Comme le montre le Tableau 1.2. [Rob04], la simulation d'opérations complexes à bas niveau devient rapidement prohibitive. Lors du développement et de la vérification des circuits, l'augmentation de l'abstraction permet d'accroître la complexité pour un même effort de conception.

Système	Objet de la simulation	Durée de simulation
Décodage vidéo MPEG2	Une trame	16 jours
Echo-Canceling	1 minute d'exécution	1 an
Démarrage de Windows	Pentium 100MHz	19 ans

Tableau 1.2. Exemples de durées de simulation au niveau électrique. [Men]

La notion de SoC est intimement liée à l'apparition de l'idée de propriété intellectuelle (ou IP pour *Intellectual Property*) appliquée aux circuits numériques [Des]. La conception s'effectue à base de blocs (BBD pour *Block-Based Design*), et conduit à la manipulation de composants virtuels (VC – *Virtual Component*).

Deux types d'IP peuvent être définis.
Les ***Hard IP*** contiennent les informations relatives à un circuit réalisé et caractérisé dans une technologie donnée. Leur diffusion implique le transfert de l'ensemble des indications relatives à leur fabrication et au test.
Les ***Soft IP*** sont portables et indépendantes de la technologie. Leur mise en œuvre exige l'utilisation d'un outil de synthèse. Dans la suite de ce manuscrit, le terme IP fera référence à cette catégorie.

Le développement d'IP consiste donc à favoriser la portabilité et la réutilisabilité d'objets matériels. Historiquement, les IP ont été destinées en premier lieu au monde numérique. En effet, la conception dans ce domaine fait principalement appel à une démarche descendante, méthode pour laquelle le système à concevoir est décrit à un haut niveau d'abstraction, ou

niveau fonctionnel. Par la suite, cette description est raffinée par étapes successives pour aboutir à une description physique. Cette technique permet donc assez naturellement de faire appel aux IP.
La conception de circuits analogiques et mixtes quant à elle suit généralement une démarche ascendante. Elle consiste à débuter la conception d'un système par une description physique dépendante de la technologie utilisée. La génération d'IP-AMS, c'est-à-dire d'IP appliquées aux circuits analogiques et mixtes (*Analog and Mixed Signals*), est un domaine qui n'a commencé à se développer qu'assez récemment [Bar, Des01, Mad01, Pol02].

Ce travail s'inscrit dans une problématique consistant à faciliter la conception de systèmes monopuces interfaçant des blocs analogiques, mixtes et numériques. Il se crée donc une convergence des méthodes de conception analogique vers les méthodes descendantes. Cela implique le développement de bibliothèques « AMS » et d'outils permettant de sélectionner et de dimensionner un circuit en fonction de ses besoins de façon systématique [Van02, Ruo05] voire automatique comme c'est déjà le cas en numérique depuis plus d'une dizaine d'années. Ce sera en partie le sujet du chapitre 2 avec une application concernant les convertisseurs analogique-numérique (CAN).

La mise au point et la vérification d'IP dans un contexte SoC peut mettre à contribution des ressources matérielles. Pour le domaine numérique, des composants reprogrammables tels que les DSP [Ti] et les FPGA [Xil, Alt] sont très versatiles et permettent de tester des applications variées. Ils sont le plus souvent disponibles sur des cartes de prototypage [Bar04a, Hou02, Hun, Sun] facilitant l'échange de données avec les composants. Il se pose le problème du partitionnement logiciel–matériel lorsque l'architecture d'un SoC est hétérogène [Len03, Tho99]. Nous ne nous intéresserons pas particulièrement à cette problématique par la suite en retenant le choix d'une architecture matérielle. En effet, l'application visée est constituée d'un CAN associé à des traitements numériques simples mais travaillant à des fréquences élevées (cette problématique sera développée dans le chapitre 3). De telles opérations se prêtent donc assez naturellement à une implémentation sur FPGA.

Les avancées technologiques de l'électronique conduisent à repenser la conception de systèmes et voient la généralisation de méthodes de conception modulaire (BBD) et d'une nouvelle génération de circuits intégrés, les SoC. Ces notions sont appliquées ici au domaine des télécommunications en se concentrant particulièrement sur le thème émergent de la radio logicielle.

1.2 Radio logicielle

1.2.1 Concept

Le secteur des télécommunications sans fil implique de façon importante les progrès en terme de consommation, de compacité et de fréquence. A chaque nouveau standard est attribuée une bande de fréquences propre. Chaque nouvelle bande ne devant pas recouvrir les précédentes, les fréquences utilisées augmentent. Alors que le GSM (*Global System for Mobile communication*), démocratisé vers le milieu des années 1990 utilise les bandes 890-915 MHz et 935-960 MHz, l'UMTS (*Universal Mobile Telecommunication System*) standardisé en décembre 1999 occupe les bandes 1920-1980 MHz et 2110-2170 MHz. Afin d'assurer des débits de données de plus en plus élevés et une prise en charge d'un nombre croissant d'utilisateurs, les bandes attribuées aux standards sont de plus en plus larges.

Dans le paragraphe §1.1.1.2, nous avons vu l'intérêt d'emprunter à l'informatique ses caractères modulaire et programmable pour les appliquer à des domaines tels que l'électronique et plus spécifiquement les télécommunications sans fil. Nous pouvons d'ailleurs constater une convergence mutuelle de ces deux domaines puisque les ordinateurs portables se miniaturisent avec l'émergence des PDA tandis que les téléphones portables incluent de plus en plus d'applications multimédia (photographie numérique, vidéo, MP3, accès à Internet). Dans un avenir proche il sera donc de plus en plus difficile de distinguer un ordinateur d'un téléphone mobile. C'est ainsi que peut naître l'idée de systèmes de télécommunication portables programmables répondant au terme général de radio logicielle (RL) ou *software radio* (SWR).

Il est donc prévu d'intégrer aux futurs systèmes mobiles la capacité d'être reconfigurés afin de pouvoir répondre avec un matériel unique à une diversité de fonctionnalités et de standards de télécommunication existants et éventuellement en développement. En effet, à la deuxième génération (GSM, IS-95, DECT, PHS…) s'ajoutent les standards dits 3G (W-CDMA, cdma2000, TD-CDMA) et les réseaux locaux comme bluetooth, IEEE 802.11b. Les objectifs commerciaux consistent donc à développer différentes applications autour d'un matériel standard évitant ainsi un « empilement » de chaînes de traitements dédiées chacune à un standard particulier (*stack radios*) [Pal01, TuT02]. L'intérêt se trouve essentiellement pour des matériels appartenant au moins à la génération 2.5G, c'est-à-dire incluant des fonctions multimédia. Concernant les terminaux, le principal enjeu est de trouver le meilleur compromis entre la puissance consommée, les performances et le prix. Cette recherche est moins prioritaire dans le cas des stations de bases pour lesquelles les contraintes sont moins draconiennes en termes d'intégration et de consommation [Mit02, Tut02].

L'une des perspectives majeures de la radio logicielle est l'AI-SR (*Adaptative Intelligent Software Radio*) pour laquelle le terminal doit s'adapter automatiquement à son environnement pour augmenter ses performances et son efficacité spectrale [Mit02].

1.2.2 Partitionnement analogique–numérique

La radio logicielle conduit au schéma de principe de la Figure 1.6. Afin de favoriser les capacités de reprogrammation, l'idée principale consiste à amener les conversions analogique-numérique (CAN) et numérique-analogique (CNA) au plus près de l'antenne pour effectuer un maximum d'opérations en numérique. Les objectifs visés sont la flexibilité, la reconfigurabilité et la portabilité de l'application.

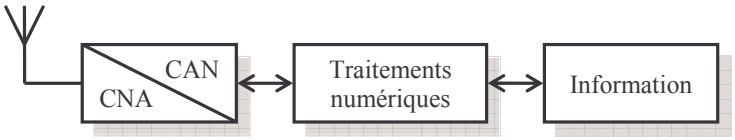


Figure 1.6. Schéma de principe de la radio logicielle idéale.

La suite de cet exposé s'intéresse essentiellement à la tête de la chaîne de réception (le terme « tête » fait référence à celui de *front-end*, fréquemment employé dans la littérature). Du fait des meilleures performances des CNA par rapport aux CAN, elle présente a priori plus de difficultés de réalisation que la partie émission. Le schéma bloc (Figure 1.7) d'un récepteur de type radio logicielle idéale se présente donc de la manière suivante :

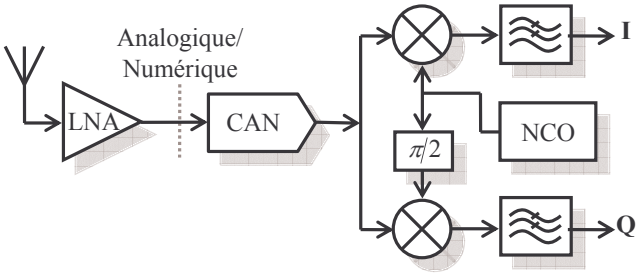


Figure 1.7. Schéma bloc d'un récepteur radio logicielle idéale.

Les inconvénients majeurs des circuits analogiques résident essentiellement dans la difficulté de reconfiguration, la sensibilité aux perturbations, le vieillissement, le « shift » de performances… Cependant, il est important de signaler que les circuits numériques souffrent du fait qu'ils ne peuvent assurer à ce jour des traitements rapides aux fréquences RF. Un compromis apparaît donc entre les deux domaines et le besoin d'établir une frontière, fluctuant et évoluant en fonction des progrès de l'électronique.
Dans l'idée de transposer la conversion vers les hautes fréquences, ce partitionnement dépend essentiellement de deux facteurs limitants : le débit maximum et la résolution des CAN disponibles. Les performances des convertisseurs sont principalement limitées par trois phénomènes physiques (Figure 1.8) : le bruit thermique, le jitter, c'est-à-dire

l'incertitude sur l'instant d'échantillonnage (de l'ordre de quelques dixièmes de ps), et la résolution (pas de quantification minimum) des comparateurs constitutifs du CAN [Lou02]. Dans le cadre de la radio logicielle, les CAN doivent pouvoir répondre à différents standards sachant que les fréquences porteuses sont de l'ordre de quelques GHz. Dans ces conditions, la linéarité souhaitée est forte ce qui conduit à des résolutions d'environ 18 bits. Actuellement, les convertisseurs commercialisés présentent des caractéristiques variant entre 1 MHz pour 24 bits, 200 MHz pour 12 bits et 2 GSPS pour 8 bits (les architectures basées sur la mise en parallèle de CAN élémentaires sont exclues, certaines consommant plusieurs Watts sont inutilisables pour des systèmes mobiles).

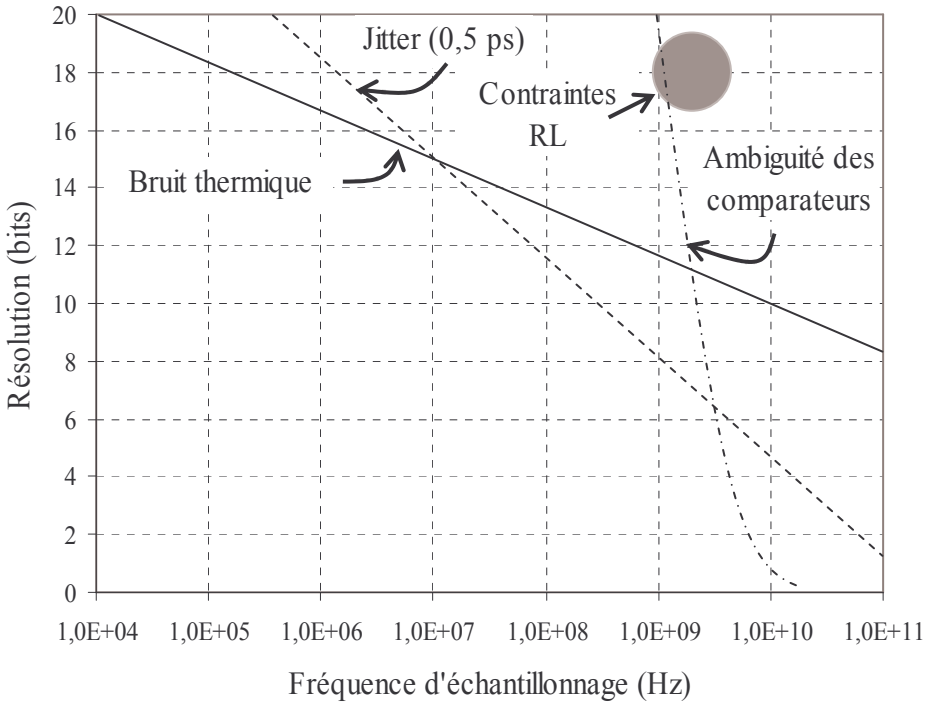


Figure 1.8. Besoins de la radio logicielle en matière de CAN et limitations technologiques (source : [Lou02]).

La radio logicielle dite idéale est donc irréaliste pour un avenir proche. Les CAN imposent un partitionnement analogique numérique [Don97] laissant encore la place à une tête analogique (cf. Figure 1.9). On parle alors de radio logicielle restreinte (RLR ou SDR pour *Software Defined Radio*) [Mit02, Tut02].

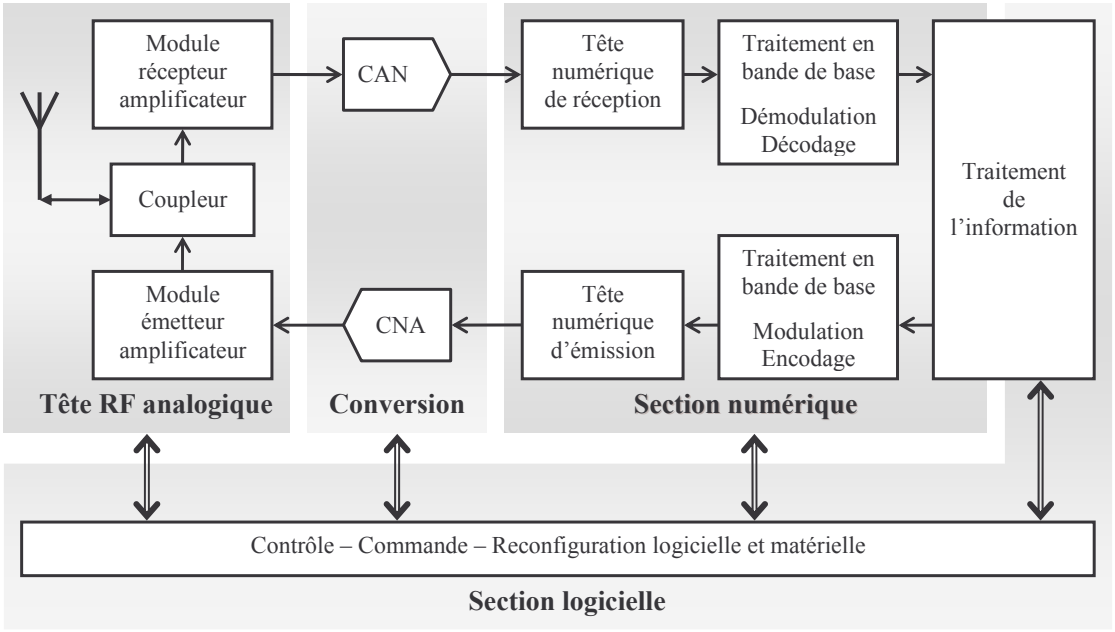


Figure 1.9. Synoptique de l'architecture d'un système radio logicielle restreinte.

1.2.3 Radio logicielle restreinte : architectures candidates

L'étude du partitionnement analogique numérique conduit à considérer différentes solutions [Kou01a, Kou01b, Tut02, Wep95] : l'architecture est étroitement liée à la sélection et à l'utilisation du CAN.

1.2.3.1 Sous échantillonnage

La solution la plus proche de la radio logicielle idéale consiste à utiliser la technique du sous échantillonnage (appelée également échantillonnage de bande pour *bandpass sampling*). Elle conduit à l'architecture de la Figure 1.7 puisque le signal RF est directement converti en numérique.
Le principe de cette architecture est d'utiliser le repliement des spectres induit par l'échantillonnage pour transposer la bande de fréquences à traiter en bande de base [Vau91].

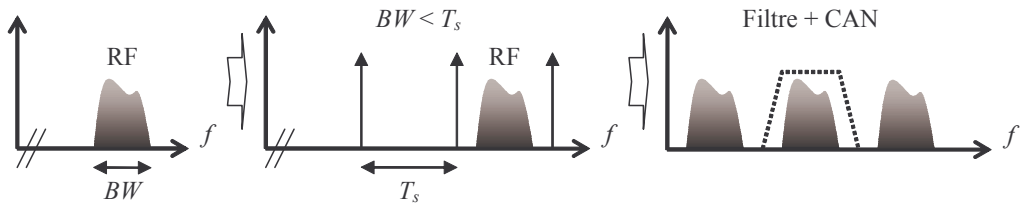


Figure 1.10. Utilisation du sous échantillonnage pour la réception.

Cette solution comporte trois inconvénients majeurs.
Le premier est une perte de dynamique du système global correspondant à une augmentation du bruit de fond du signal numérisé. En effet, le rapport signal à bruit (SNR

exprimé en dB (1.2)) est dégradé comme l'indique le dernier terme de l'égalité suivante [Wep95] :

$$SNR = 6{,}02N + 1{,}76 + 10.\log\left(\frac{f_s}{2.f_{\max}}\right) \tag{1.2}$$

N représente le nombre de bits du CAN considéré, f_s la fréquence d'échantillonnage utilisée et f_{max} la fréquence maximale contenue dans le signal à numériser. Le SNR ne dépend pas uniquement du nombre de bits du CAN, sauf dans le cas particulier d'un échantillonnage à la fréquence de Nyquist définie par $f_s = 2.f_{max}$. Dans le cas du sous échantillonnage, ($f_s < 2.f_{max}$), le SNR est dégradé et le nombre effectif de bits (ENOB pour *Effective Number Of Bits*) répondant à la relation (1.3) diminue à raison de (m-2).0,5 bits pour $f_s = 2^{(m-1)}.f_{max}$.

$$ENOB = \frac{SNR - 1{,}76}{6{,}02} \tag{1.3}$$

Le deuxième inconvénient réside dans la nécessité d'utiliser un filtre analogique très sélectif pour isoler la bande système sans provoquer de chevauchement avec les canaux adjacents.
Enfin, la bande passante analogique du CAN doit au moins être égale à la fréquence la plus élevée de la bande système (cf. f_{max} dans (1.2)). Actuellement, les bandes passantes en entrée des CAN commercialisés sont de l'ordre de 0,5 GHz [An2, All03].

1.2.3.2 Conversion directe

A l'autre extrémité du spectre des possibilités architecturales de TRM, une transposition en bande de base dans le domaine analogique avant conversion peut être envisagée [Col03] (Figure 1.11).

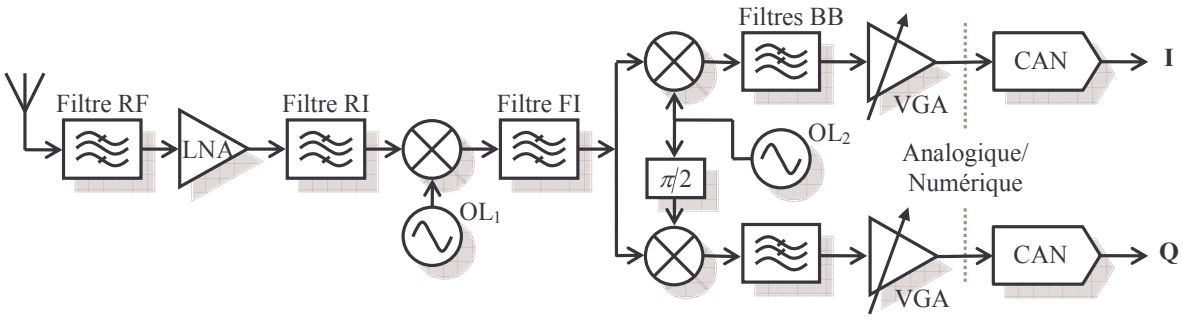


Figure 1.11. Schéma bloc d'un récepteur à conversion directe.

Dans ce cas, la transposition du signal RF en bande de base est analogique. Les contraintes sur la numérisation sont réduites puisque celle-ci s'opère à des fréquences relativement basses (Figure 1.12).

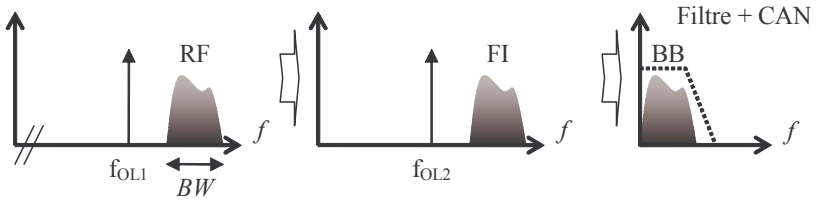


Figure 1.12. Utilisation de la conversion directe pour la réception.

Dans ce cas, l'architecture est relativement éloignée du postulat de base de la radio logicielle qui consiste à exécuter en numérique le plus d'opérations effectuées initialement en analogique. Sans intégrer la capacité de reprogrammation inhérente à la RLR, cette architecture est actuellement utilisée dans les systèmes mobiles.

1.2.3.3 Numérisation en fréquence intermédiaire

La troisième solution (Figure 1.13) consiste à utiliser une tête analogique effectuant la transposition de la bande système en fréquence intermédiaire (FI) et un seul CAN.

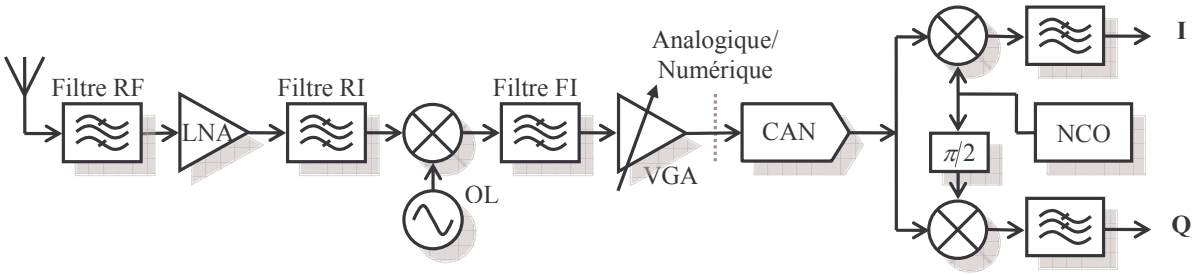


Figure 1.13. Schéma bloc d'un récepteur superhétérodyne avec conversion AN en FI.

Dans ce cas de figure, la transposition en bande de base est réalisée en numérique (Figure 1.14). Seul un mélangeur associé à un oscillateur libre (OL) doit être mis en œuvre à l'aide de circuits analogiques. Ce dernier doit fournir des capacités de reconfiguration pour avoir un caractère multistandard [Rou04].

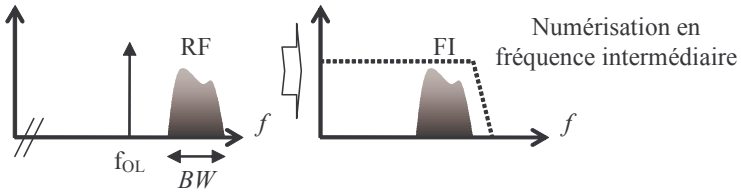


Figure 1.14. Utilisation de l'architecture superhétérodyne.

1.2.3.4 Architecture retenue

La vue d'ensemble des différentes solutions conduit à s'intéresser particulièrement à l'architecture utilisant une numérisation du signal en fréquence intermédiaire pour deux raisons : elle reste proche du concept de radio logicielle idéale et les contraintes qu'elle impose peuvent être respectées avec l'état de l'art actuel.

Différents travaux traitent de la reconfigurabilité des blocs constitutifs de la tête analogique. Celle-ci est principalement constituée d'un système de transposition des fréquences radio vers une FI et d'une antenne (Figure 1.13).

Dans [Boe03, Rou04] des fonctions analogiques reconfigurables sont présentées. Les solutions proposées dans la littérature concernent seulement la mise en œuvre de blocs multifonctions et non la tête analogique dans son ensemble.

Les composants analogiques reprogrammables de type FPAA (*Field Programmable Analog Array*) sont les équivalents analogiques des FPGA. Leurs performances actuelles ne permettent pas des fréquences d'utilisation supérieures à quelques dizaines de kHz [An1]. Il est cependant utile de prévoir des progrès dans ce domaine et de rester en veille sur ces technologies.

Dans ce contexte, les antennes doivent être capables de capter et d'émettre dans les bandes de fréquences des standards à prendre en charge. Les deux familles d'antennes candidates pour la RLR sont les antennes multibandes [Dec02, Vil02] et les antennes reconfigurables [Leg05, Pan04].
Les premières présentent plusieurs modes de résonances adaptés en impédance, elles sont donc conçues pour fonctionner sur plusieurs bandes de fréquence. Il est difficile d'adapter plus de deux bandes avec ces techniques.
Les antennes reconfigurables ont vu ces dernières années l'émergence des MEMS (*Micro Electro Mechanical Systems*) combinant l'électronique et la mécanique pour modifier l'architecture matérielle des antennes et donc leur comportement (fréquences adaptées, dépointage).

Les aspects purement analogiques évoqués dans cette partie ne font pas l'objet de développements ultérieurs. Une consultation de la littérature citée peut donner une vue d'ensemble de l'état de l'art. *A contrario*, les CAN seront traités plus largement dans le chapitre 2. De nombreuses solutions de reconfiguration seront proposées pour ces composants [Sum02a, Sum02b, Liu02, Gul01, Set99].

Nous venons de définir l'environnement radio logicielle restreinte auquel nous nous intéressons. La suite de l'étude concerne tout particulièrement les méthodes de dimensionnement de l'opération de conversion analogique numérique et des traitements numériques rapides inhérents à la réception.

1.3 Mise en place d'une méthodologie de conception

1.3.1 Objectifs

Ce travail est guidé par le concept SoC-AMS et l'idée de concevoir des systèmes analogiques/mixtes en suivant une démarche descendante. Cela implique de constituer une bibliothèque d'IP mixtes, analogiques et numériques ce qui permet d'optimiser des sous fonctions plutôt qu'un système dans son ensemble. Dans le cadre de la RLR, l'aspect reconfigurable du système doit être pris en compte. Cette caractéristique conduit à la nécessité d'isoler des opérateurs communs entre fonctions différentes. Cette étape permet d'identifier les blocs de base à concevoir et de définir leur granularité.

Le système considéré représente l'ensemble CAN/tête de réception numérique (TRN) d'un récepteur RLR correspondant à l'architecture définie dans le paragraphe 1.2.3.3. Cet ensemble est appelé tête de réception mixte (TRM). Les différentes étapes conduisant à leur conception et à leur dimensionnement sont présentées sur la Figure 1.15. L'objectif final est la validation conjointe des parties CAN et tête numérique à l'aide d'une simulation de la TRM dans sa globalité. L'implantation sur une carte de prototypage sera évoquée dans le but de réaliser et de valider la TRM conçue.

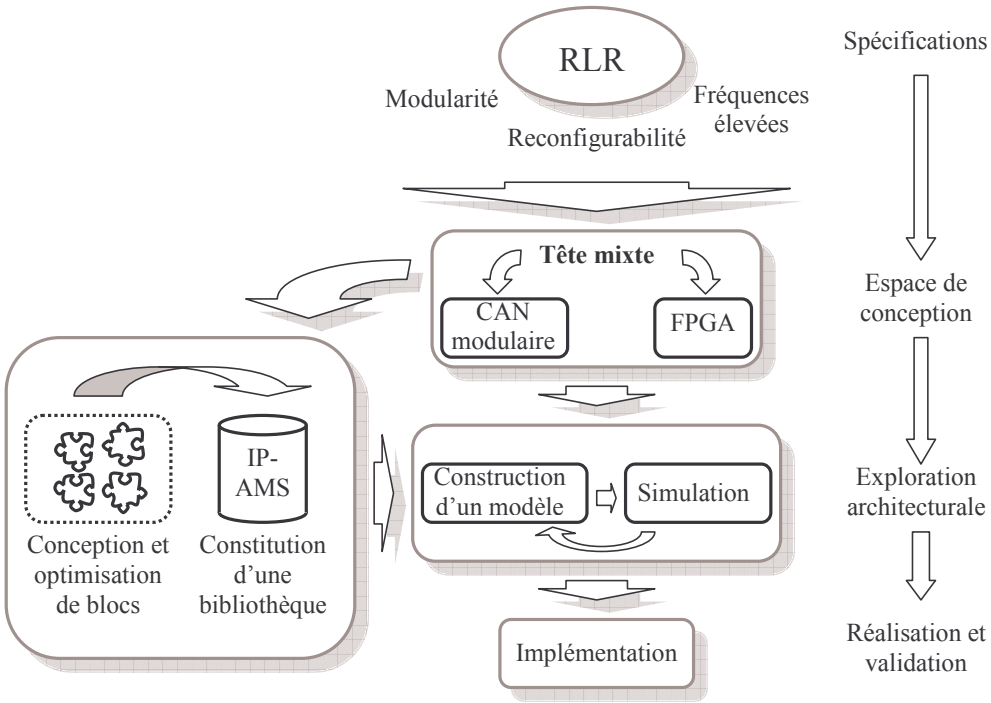


Figure 1.15. Principe général du flot de conception appliqué à la RLR.

1.3.2 Exploration architecturale

1.3.2.1 Langages de description

Pour mener l'exploration architecturale sur la TRM, la méthode la plus fiable consiste à simuler le système dans sa globalité. Une simulation de bas niveau (ou niveau transistor) pourrait être utilisée, mais cette méthode, trop consommatrice en terme de temps de calcul, ne permet pas une exploration efficace. De plus, elle nécessite une durée de développement importante et reste très dépendante de la technologie visée. La conséquence est un manque de portabilité des modèles.
L'environnement Matlab/Simulink couramment utilisé et les outils de simulation électronique sont distincts ; cette rupture du flot alourdit le passage d'une étape à la suivante. De plus, la cosimulation est relativement complexe et les modèles obtenus sont peu évolutifs [Ruo05].

Une méthode basée sur un langage de haut niveau est donc préférée. Les temps de simulation étant réduits, il est alors possible de simuler et de comparer efficacement différentes architectures.
Pour cette étude, le VHDL (*Very high speed integrated circuit Hardware Description Language*) et son extension aux systèmes mixtes VHDL-AMS (VHDL for *Analog and Mixed Signals*, [Std99]) sont utilisés pour simuler des modèles comportementaux dans le domaine des signaux mixtes [Mcc98, Nun98]. Ces langages, non propriétaires, permettent d'assurer la portabilité des modèles.
La méthode développée doit opérer la convergence de la conception mixte vers les flots numériques existants [Ruo05] (Figure 1.16).

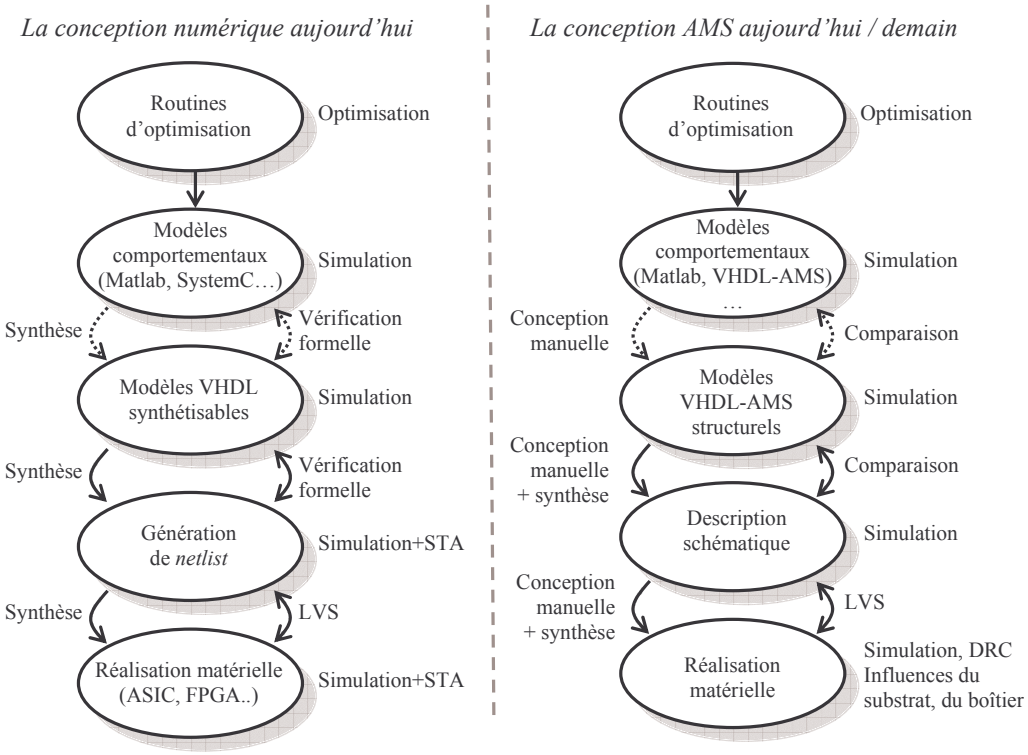


Figure 1.16. Convergence des flots de conception AMS et numérique (source [Ruo05]).

1.3.2.1.1 Utilisation du VHDL

Les méthodologies de conception numérique se sont adaptées pour remplacer une démarche ascendante encore utilisée en analogique par une démarche descendante. Cette mutation a facilité le développement de systèmes de plus en plus complexes tout en diminuant les délais de mise sur le marché. C'est ainsi qu'est apparue la nécessité de créer des langages de description de haut niveau d'abstraction comme le VHDL. Ces langages permettent d'appréhender les systèmes à concevoir dans leur globalité et d'en assurer le test et la maintenance de manière efficace. Toutes les étapes du cycle de vie des systèmes électroniques, de la spécification à la réalisation, sont ainsi couvertes.

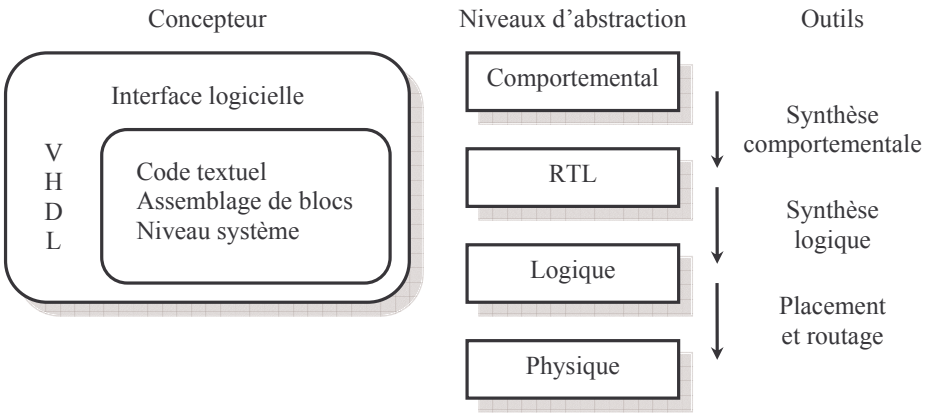


Figure 1.17. Flot de conception VHDL.

A l'aide d'outils de conception [Xil] et de simulation [Men] spécifiques, un flot VHDL complet sera utilisé, c'est-à-dire de la description haut niveau jusqu'à l'implémentation sur FPGA.

1.3.2.1.2 Utilisation du VHDL-AMS

Si l'on franchit les frontières du numérique, il apparaît un vaste champ de réflexion concernant, outre la continuité du temps et des amplitudes, la prise en compte du bruit, l'utilisation de grandeurs physiques (la mécanique, la thermique…), et leurs interactions avec les circuits numériques. Les concepteurs de systèmes numériques voulant développer un système mixte ou un système numérique en prenant en compte son environnement sont confrontés à la problématique suivante : la nécessité de définir, de modéliser, d'optimiser, de tester et de réaliser des structures hétérogènes.

Jusqu'à présent, il était possible de simuler de tels systèmes au niveau circuit mais ce type de simulation nécessite des modèles dépendant d'une technologie. Cette exigence va à l'encontre de l'utilisation d'une démarche descendante, liée au développement de systèmes complexes. De plus, par ce moyen, il n'est pas possible de profiter de la performance des outils de simulation numérique de haut niveau.
Le défi de la conception de systèmes mixtes est de pouvoir bénéficier des avancées techniques et méthodologiques de l'électronique numérique tout en intégrant des éléments analogiques (circuits analogiques et mixtes, bruit). L'objectif est donc de se situer à un niveau d'abstraction élevé pour optimiser les coûts de simulation tout en donnant la liberté de combiner des éléments appartenant à des disciplines physiques différentes. C'est l'ensemble de ces besoins qui a conduit à la définition du standard IEEE 1076.1 [Std99] plus connu sous le nom de VHDL-AMS.

Il est possible de définir le VHDL-AMS comme étant une extension du VHDL [Herv02]. Il présente une structure identique au VHDL (typage fort, organisation en bibliothèques, déclaration d'objets, unités de conception, hiérarchisation, mots-clé de base, gestion de la variable temps, description concurrentielle/séquentielle…) tout en élargissant ses possibilités. Le VHDL-AMS permet de décrire des modèles paramétrables à différents niveaux (système, macro modèle, transistor), et ce pour différents domaines d'application comme l'automobile, les télécommunications, la biométrie, l'optique intégrée… En effet, une de ses particularités est de ne pas se limiter à la discipline de l'électronique au sens large, mais il permet d'utiliser un environnement unique pour modéliser des phénomènes appartenant à d'autres domaines de la physique (thermique, mécanique…). Cependant, dans la suite de ce mémoire, nous allons limiter le champ d'application à l'électronique sachant qu'il est possible de prendre en compte des phénomènes physiques (thermiques par exemple) dans le fonctionnement d'un système électronique. Tout comme en numérique, ces modèles sont portables, c'est-à-dire intégrés dans des bibliothèques et réutilisables. Des précisions sur l'utilisation de ce langage sont disponibles dans les références suivantes : [Gar00, Herv02, Hou05, Std99]. Les principaux ajouts du VHDL-AMS par rapport au

VHDL sont rapidement développés dans les sections suivantes : formalismes de connexions, critère de solvabilité et synchronisation des noyaux analogique et numérique.

Formalismes de connexions, transport de l'information analogique.
Les grandeurs analogiques sont regroupées sous le mot réservé *quantity* et correspondent à trois usages distincts :
Les *quantités de branches* sont utilisées pour des connexions entre modèles utilisant le formalisme des lois de Kirchhoff généralisées pour lesquelles l'énergie d'un système est conservative.
Les connexions de type flot de données requièrent la définition de *quantités libres*. Dans ce cas, les lois de conservation de l'énergie ne sont pas prises en compte.
Les *quantités implicites* servent à la manipulation de données intermédiaires au sein d'un système.

Le **critère de solvabilité**, défini par la norme, impose au concepteur de construire un modèle de telle sorte que le simulateur dispose d'autant d'équations que d'inconnues. Un modèle ne vérifiant pas cette contrainte ne peut en aucun cas être simulé.
Les équations en question regroupent les équations différentielles algébriques définies explicitement par l'utilisateur, les équations issues des connexions correspondant aux lois de Kirchhoff, et les équations calculant les quantités implicites. Il y a autant de relations de Kirchhoff généralisées qu'il y a de terminaux de référence. Les inconnues sont quant à elles les quantités de branche, les quantités libres et les quantités implicites. Il faut donc bien prendre soin de faire correspondre une équation (explicite ou non) à chaque quantité déclarée.

Synchronisation des noyaux analogique et numérique :
Un cycle de simulation temporelle est constitué de plusieurs étapes (Figure 1.18). Dans un premier temps, le simulateur opère une initialisation afin d'obtenir l'état dit *quiescent* qui représente le point de stabilité, début de la simulation temporelle. Cet état représente les conditions initiales du système à ne pas confondre avec les valeurs initiales choisies par l'utilisateur et desquelles elles peuvent différer. Le temps est alors positionné en $T_0 = 0$. Partant de cette origine, le simulateur analogique effectue les calculs qui lui incombent avec son propre pas de simulation variable jusqu'à ce qu'un événement numérique se produise à l'instant T_n. S'il se produit un franchissement de seuil analogique avant T_n, c'est-à-dire un changement sur le booléen *Q'above(E)*, le simulateur analogique est arrêté pour permettre au simulateur numérique de prendre en compte l'événement. Si cet événement intervient avant T_n, le point de synchronisation des noyaux est donc avancé à l'instant T_n'. Une fois toutes les mises à jours effectuées à l'instant T_n, le cycle de simulation recommence entre T_c (instant courant remplaçant T_n) et T_n (occurrence suivante d'un événement numérique).

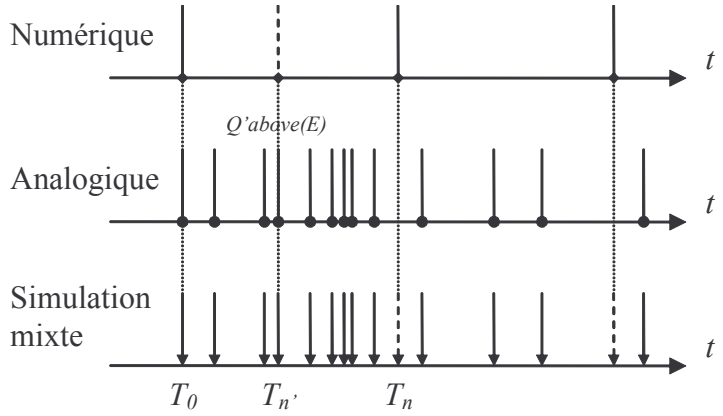


Figure 1.18. Synchronisation des noyaux analogique et numérique.

Le VHDL-AMS n'est pas, à l'heure actuelle, associé à des outils de conception permettant d'effectuer de la synthèse matérielle. Certains groupes de recherche développent des flots intégrant des bibliothèques d'IP analogiques et mixtes décrites à l'aide de ce langage (ce qui sera en partie l'objet du deuxième chapitre), mais aucune démarche n'est aussi aboutie que celle de la conception de systèmes purement numériques à l'aide du VHDL [Bar, Neo, Hers02].

1.3.2.2 Flot de conception

Une étude préliminaire semble nécessaire pour fixer un point de départ à l'exploration architecturale proposée [Bar04b, Bar04c]. Un exemple simple est présenté dans ce paragraphe afin de poser les bases d'un flot de conception et de rendre compte des développements attendus. Ce travail se concentre sur la conversion et les traitements numériques associés c'est-à-dire la tête de réception mixte. La méthode utilisée se base sur la modélisation et la simulation des blocs numériques et mixtes dans un environnement unique. Cette technique doit faciliter l'exploration architecturale en considérant la conception d'un système mixte dans son ensemble. Cela permet de valider le fonctionnement d'un système en prenant en compte l'interaction entre les blocs analogiques, mixtes et numériques. De plus, l'optimisation et le dimensionnement de ces blocs peuvent être menés en tirant parti d'un espace de conception plus large.

La validation de cette méthode est résumée sur la Figure 1.19. La plateforme de prototypage utilisée (Hunt engineering, module IO2V2 [Hun]) est composée de deux CAN 12 bits/105 MSPS et de deux CNA 14 bits/125 MSPS et d'un FPGA de 1 million de portes. Trois outils du marché ont été utilisés : Xilinx ISE [Xil] pour la spécification de l'architecture à partir d'une bibliothèque d'IP et pour l'implémentation sur FPGA, et ADVanceMS et ModelSim [Men] pour la simulation des blocs mixtes et numériques.

La plateforme permet d'implanter l'application numérique sur FPGA et de valider son fonctionnement à partir de données provenant des CAN. De plus, elle est utilisée pour

extraire à partir de mesures un modèle simple et fiable des CAN disponibles. Celui-ci est ensuite utilisé comme bloc constitutif du système mixte global.

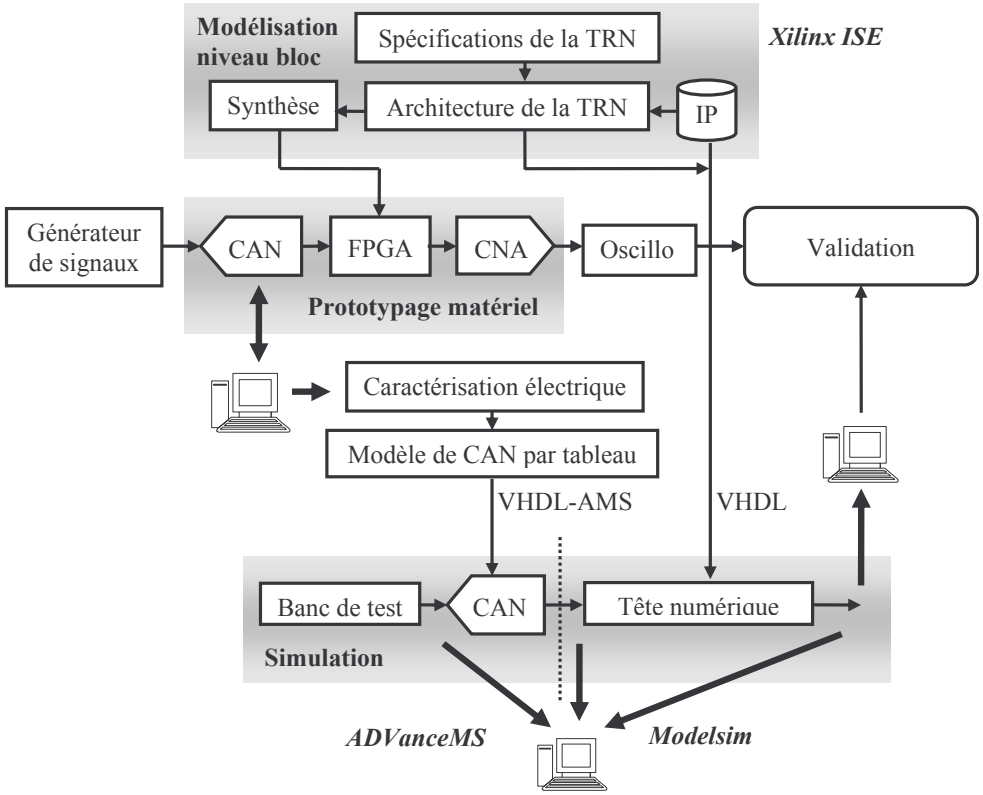


Figure 1.19. Flot de validation de la méthodologie.

L'intérêt d'une approche globale concernant la TRM peut être illustrée par deux démarches différentes.

Dans un premier cas, la conception de la TRN peut être menée en considérant l'utilisation d'un CAN spécifique. Ainsi, la TRN est simulée en incluant un modèle de CAN de haut niveau ce qui permet de dimensionner les blocs numériques et de valider le fonctionnement global. L'exploration architecturale est donc effectuée au niveau bloc ou au niveau système. Cela permet également au concepteur d'utiliser directement des signaux analogiques en entrée du banc de test et éventuellement d'inclure des sources de bruit.

Une autre approche peut consister à la sélection d'un CAN pour une TRN particulière. Si des CAN réels doivent être comparés et testés, des modèles de haut niveau doivent être développés et validés.Une simulation mixte permet la comparaison entre l'utilisation de différents CAN réels et de vérifier qu'un CAN correspond bien aux contraintes de la TRN. Si un CAN doit être conçu pour une application particulière, l'exploration architecturale sera orientée vers ce circuit.

Le niveau de granularité du modèle doit être choisi en fonction des besoins de l'application. Par exemple, pour une architecture de CAN de type pipeline, le bloc de base peut être un étage de conversion ou bien les blocs le constituant (CNA, amplificateur…). Le langage VHDL-AMS permet de choisir le niveau de description et de développer des modèles multi

niveaux. De plus, la définition des imperfections peut être étudiée avec intérêt, y compris à niveau d'abstraction élevé.
Dans un contexte de RLR, la reconfiguration de la tête de réception mixte (TRM) est un point essentiel. Par exemple, le remplacement de coefficients de filtre numérique ainsi que la modification de l'architecture du CAN peuvent être effectués en cours de fonctionnement. L'impact de cette reconfiguration sur le système global peut donc être observé avec précision et cette approche permet également d'identifier des interactions entre les fonctions analogiques et numériques.

1.3.2.3 Etude de faisabilité

Pour illustrer l'intérêt d'une telle approche, un exemple simple et didactique de TRM a été développé. La première étape consiste à définir un modèle du CAN de la carte de prototypage à partir de mesures. Dans la deuxième, les blocs constitutifs de la TRN sont dimensionnés. A partir des ces deux fonctions (CAN et TRN) la TRM peut être simulée. Au final, la TRN est implémentée dans le FPGA de la plateforme de prototypage et le système global est validé de façon fonctionnelle.

La Figure 1.20 présente une vue fonctionnelle de la TRM. Dans cette architecture, le signal analogique d'entrée est considéré comme étant en fréquence intermédiaire (FI) basse puis converti par le CAN.

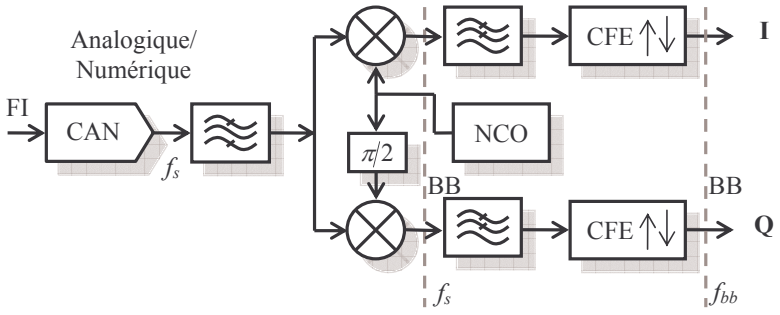


Figure 1.20. Vue fonctionnelle de la tête mixte.

La TRN est composée de trois modules [Hen99b] : transposition en bande de base (BB), sélection de canal et conversion de fréquence d'échantillonnage (CFE) qui permet de convertir le débit f_s imposé par le CAN en un débit f_{bb} imposé par chaque standard de communication [Hen99a].

1.3.2.3.1 Extraction d'un modèle de CAN

Pour extraire un modèle des CAN de la carte, la solution utilisée consiste à mesurer les paliers de conversion et de remplir un tableau répertoriant les niveaux de transition correspondant à chaque code [Per95]. Ce tableau (ou *Look-Up-Table* – LUT), modélise une partie des imperfections du composant. Ce modèle sera d'autant plus proche de la réalité que les signaux utilisés en simulation seront proches de ceux utilisés pour la caractérisation

(amplitude, forme du signal, fréquences). Avec cette solution, le modèle du CAN est indépendant de son architecture et seules la dynamique d'entrée et la résolution suffisent à déterminer le modèle.
Son utilisation est simple : le signal d'entrée est échantillonné et maintenu (*sampled and hold*) et chaque échantillon est comparé aux valeurs reportées dans le tableau jusqu'à ce que l'intervalle correspondant soit trouvé. La latence du convertisseur est modélisée par un registre à décalage. La bande passante d'entrée du convertisseur est spécifiée à l'aide d'un filtrage passe-bas du premier ordre ayant une fréquence de coupure déterminée par la mesure.

Un CAN idéal et linéaire a des paliers de conversion espacés régulièrement de LSB = $V_{ref}/2^N$ où V_{ref} représente la moitié de la plage d'entrée et N le nombre de bits. Pour un CAN réel, cette valeur n'est pas la même d'un niveau de conversion à l'autre, cette erreur est appelée non linéarité différentielle (NLD ou *Differential Non-Linearity*, NLD(i) pour le $i^{ème}$ palier). La méthode de l'histogramme (voir §2.2.2.1.3) est utilisée pour mesurer ce paramètre.

Le banc de test (Figure 1.21) consiste à enregistrer le nombre d'occurrences de chaque code durant l'opération de mesure avant de les analyser à l'aide d'un programme spécifique.

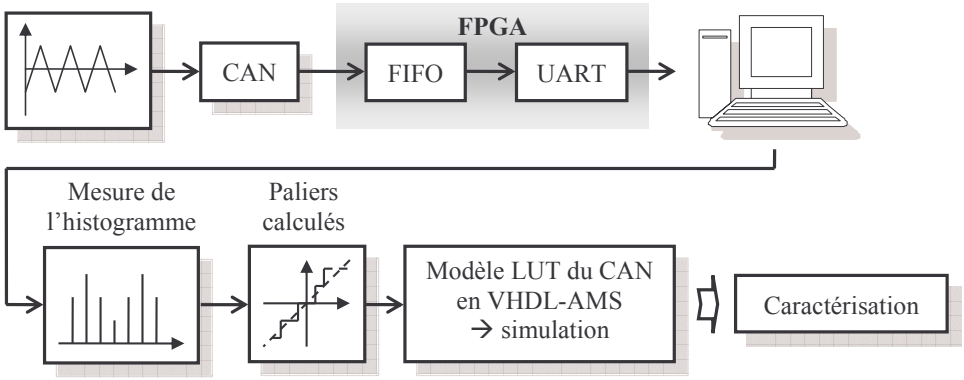


Figure 1.21. Banc de mesures permettant de définir le modèle LUT d'un CAN.

La Figure 1.22 représente la NLD de l'un des deux CAN 12 bit de la plateforme. La mesure compte 655000 échantillons, ce qui correspond à une moyenne de 160 occurrences par palier de conversion. La NLD est comprise entre -0,39 et 0,36 bit de poids faible (*Least Significant Bit* ou LSB).

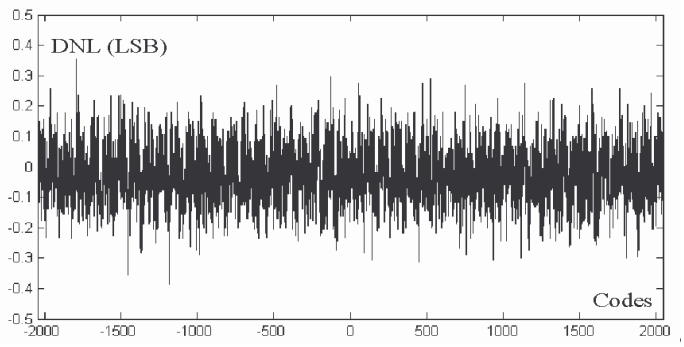


Figure 1.22. NLD mesurée.

Une fois construit, ce modèle est testé par calcul FFT, ce qui consiste en une analyse spectrale du signal de sortie du CAN excité par une sinusoïde (cette méthode ainsi que les méthodes d'ajustement seront développées dans le chapitre 2). L'évaluation et l'analyse du signal d'erreur indiquent les caractéristiques spectrales de la conversion. Pour l'utilisation de cette méthode, une sinusoïde est appliquée en entrée du CAN (Figure 1.23).

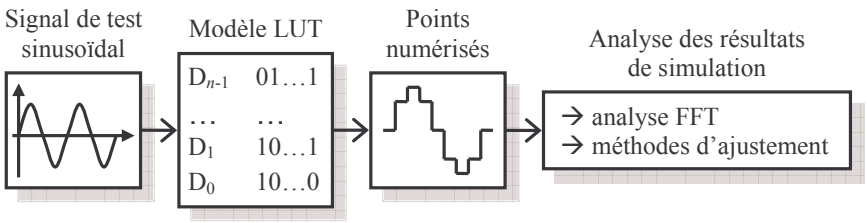


Figure 1.23. Simulation et test du modèle LUT de CAN en VHDL-AMS.

Cette méthode permet d'évaluer différentes caractéristiques spectrales du CAN : le rapport signal à bruit (SNR), la dynamique sans fréquence parasite (S*purious Free Dynamic Range* ou SFDR), le nombre effectif de bits (ENOB) et la position des harmoniques.
Sur la Figure 1.24, le signal d'entrée est une sinusoïde à 10,3 MHz.

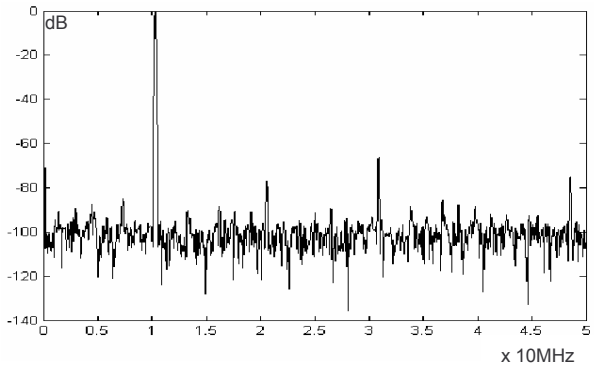


Figure 1.24. Analyse spectrale du modèle VHDL-AMS du CAN.

Les caractéristiques du CAN dans son environnement extraites par simulation sont présentées dans le Tableau 1.3.

	Données du constructeur [An1]		Résultats de simulation
	Min.	Typ.	
SNR (dB)	65,0	67,2	64,1
ENOB (Bits)	10,5	10,9	10,4
SFDR (dB)	75,0	85,0	65,9

Tableau 1.3. Comparaison entre la mesure du CAN et les données du constructeur.

Les mesures sont légèrement en deçà des valeurs minimales fournies par le constructeur. Plusieurs explications justifient ces résultats : les connexions du banc de test sont susceptibles d'introduire du bruit et le CAN est mesuré sur la carte de prototypage, c'est-à-dire dans son environnement en présence d'un filtre RC passe-bas à son entrée. Les valeurs sont également tributaires du nombre de points utilisés pour la FFT et du choix de la fenêtre de pondération.
Ce modèle de CAN *in situ* est considéré comme étant satisfaisant au regard des performances annoncées. Il peut être utilisé et intégré en tant que brique constitutive d'un système plus complexe.

1.3.2.3.2 Spécification d'une tête numérique

Les modèles utilisés pour les blocs numériques sont extraits de la bibliothèque d'IP XilinxCoreLib [Xil].
Une application simple a été choisie pour illustrer la méthodologie (Figure 1.25). Partant de l'horloge disponible sur la carte de prototypage, la TRM est conçue pour fonctionner à 100 MHz. Pour le bloc de transposition en bande de base, la liste des valeurs du NCO (*Numerically Controlled Oscillator*) est réduite à la séquence {0, 1, 0, -1} si la fréquence d'échantillonnage est exactement le quadruple de la fréquence intermédiaire.
Le signal d'entrée correspond à une modulation QPSK ayant une porteuse à 25 MHz et une période de 5 µs par mot binaire, ce qui conduit à une bande de 200 kHz.

Le premier étage est un filtre dont le rôle consiste à rejeter les signaux parasites comme les fréquences issues du repliement. C'est un filtre à réponse impulsionnelle finie (RIF) de 12 étages présentant une bande passante de 500 kHz autour de la porteuse.
Après l'opération de transposition en bande de base, les deux voies I et Q sont filtrées et une opération de conversion de fréquence d'échantillonnage est appliquée. Les filtres CIC (*Cascaded-Integrator-Comb*) remplissent simultanément ces deux fonctions. De plus, ils utilisent une architecture efficace ne contenant aucune opération de multiplication. Le composant sélectionné est une architecture à deux étages dont le coefficient de décimation est de 100.

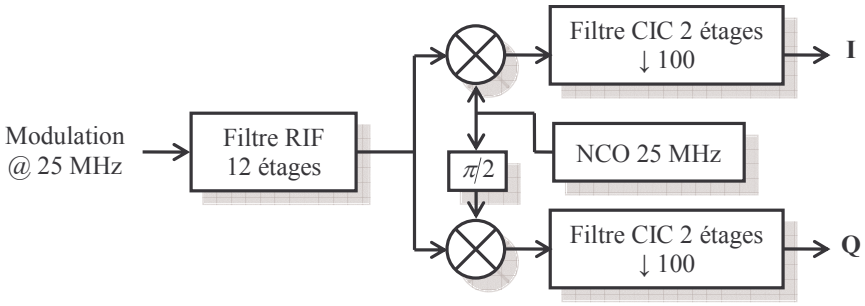


Figure 1.25. Exemple de tête de réception développé.

1.3.2.3.3 Simulation et validation

La Figure 1.26 présente les signaux analogiques et numériques dans les domaines temporel et fréquentiel. Le signal d'entrée du banc de test est la somme de cinq porteuses de 23, 24, 25, 26 et 27 MHz, avec une modulation QPSK idéale. Après filtrage et CFE, nous retrouvons les mots binaires générés en entrée dans les composantes I et Q.

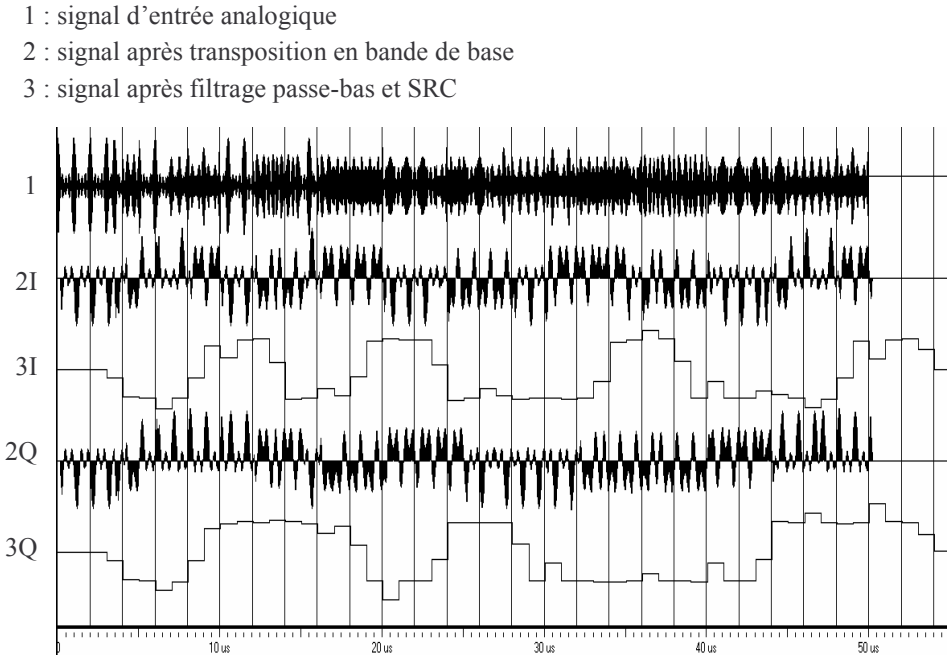


Figure 1.26. Observation du signal au sein de la TRM.

La tête numérique est implémentée sur la plateforme et les signaux analogiques sont générés à l'aide du FPGA associé à l'un des deux CNA de la carte. Une vérification qualitative permet de retrouver les mêmes composantes I et Q avec le système implémenté et en simulation.

Signalons que la simulation mixte offre la possibilité de valider les spécifications d'un bloc numérique à l'aide de signaux analogiques. Par exemple, la Figure 1.27 présente le spectre des 5 porteuses et ce même signal filtré par le filtre RIF.

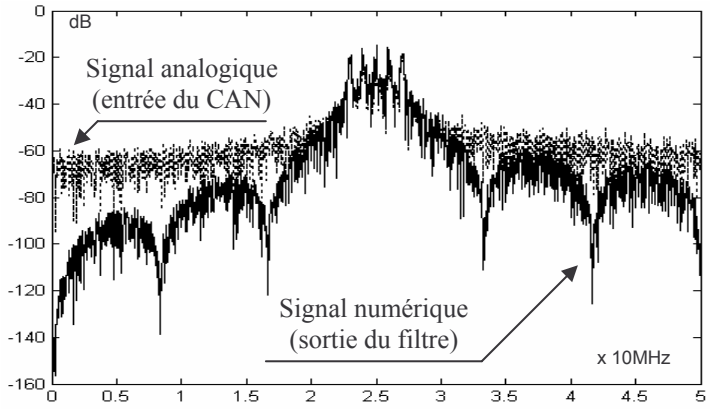


Figure 1.27. Validation du modèle comportemental du filtre RIF.

1.3.2.3.4 Bilan

Nous venons de voir le développement complet et le test d'une tête mixte. Cette méthode représente la base de l'exploration architecturale développée au fil des chapitres suivants. Les valeurs utilisées ici ne font référence à aucune application concrète, le rôle de l'exemple proposé étant de situer l'espace de conception par rapport à l'application RLR visée.

1.4 Conclusion

La croissance et les besoins du marché du semi-conducteur exercent une pression constante sur l'efficacité des méthodes de conception. Dans ce cadre, les concepts de propriété intellectuelle et de systèmes sur puce favorisent la réutilisabilité de fonctions organisées en bibliothèques généralisant ainsi les méthodes de conception modulaire. Dans le même temps, ces techniques de développement associées aux progrès technologiques permettent l'émergence des SoC. Plus généralement, les SoC-AMS impliquent l'unification des méthodes de conception numériques et mixtes afin de converger vers une démarche descendante.

Ces notions trouvent leur application dans le développement de méthodologies appliquées à la radio logicielle restreinte. La RLR vise à introduire la reconfigurabilité dans les terminaux de télécommunication mobiles pour répondre à une diversité de standards et de fonctionnalités à l'aide d'un système unique. L'espace de conception défini est constitué d'une partie de la chaîne de réception avec le CAN et la tête numérique associée dont l'ensemble constitue la tête mixte. Lors du travail d'exploration des solutions architecturales, deux aspects sont à prendre en compte : la reconfigurabilité de la TRM, et le développement de modules fonctionnels de base dont il faudra définir la granularité.

Ce premier chapitre met en évidence la nécessité de développer des procédés pour dimensionner le CAN et la TRN en fonction de spécifications, ce qui donne lieu respectivement aux chapitres 2 (Exploration architecturale de CAN) et 3 (Synthèse systématique de TRN). Les études qu'ils exposent ont été menées en parallèle et se basent sur l'idée d'un découpage modulaire des fonctions. L'objectif est alors de proposer pour chaque partie une méthode de définition de briques élémentaires afin de faciliter la généricité et la réutilisabilité des fonctions à mettre en œuvre. Ce travail conduira à la construction de systèmes modulaires, ce qui s'inscrit dans le cadre de développement de flots de conception de SoC-AMS. La mutualisation de ces deux axes de recherche est présentée dans le chapitre 4 (Mise en œuvre).

Chapitre 2

Exploration architecturale des CAN

Le premier chapitre insiste sur la nécessité de développer des méthodologies de conception descendantes concernant les systèmes analogiques et mixtes. Cette problématique est développée dans ce chapitre en se focalisant sur la sélection de CAN pour la réception radio logicielle restreinte. Cette exploration architecturale s'appuie d'une part sur un découpage modulaire du système de conversion et d'autre part sur une fonction de mérite adaptée aux contraintes de la RLR.

2.1 Introduction

Selon [Lou02] et [Mit02], le CAN représente le composant le plus critique de la RLR et constitue un « goulet d'étranglement » dans la chaîne de réception. Les performances spectrales de ce composant sont déterminantes et doivent faire l'objet d'une attention particulière.

L'exploration architecturale proposée dans ce cadre repose sur la volonté de mettre au point une méthode automatisée permettant de comparer rapidement les solutions candidates et de les classer de façon relative.
Les caractéristiques liées à un composant analogique ou mixte font l'objet de compromis : par exemple, une fréquence d'utilisation élevée, un faible bruit ou encore un grand gain pour un amplificateur sont possibles au détriment de la surface de substrat nécessaire et/ou de la puissance consommée. Ces grandeurs doivent donc être regroupées au sein d'une métrique unique reflétant ces compromis : c'est ce que l'on appelle la fonction de coût ou fonction de mérite (FDM, ou FoM pour *Figure of Merit*).

Les relations entre le développement d'IP-AMS, la mise au point d'une FDM et les contraintes concernant les CAN pour la RLR sont illustrés sur la Figure 2.1.

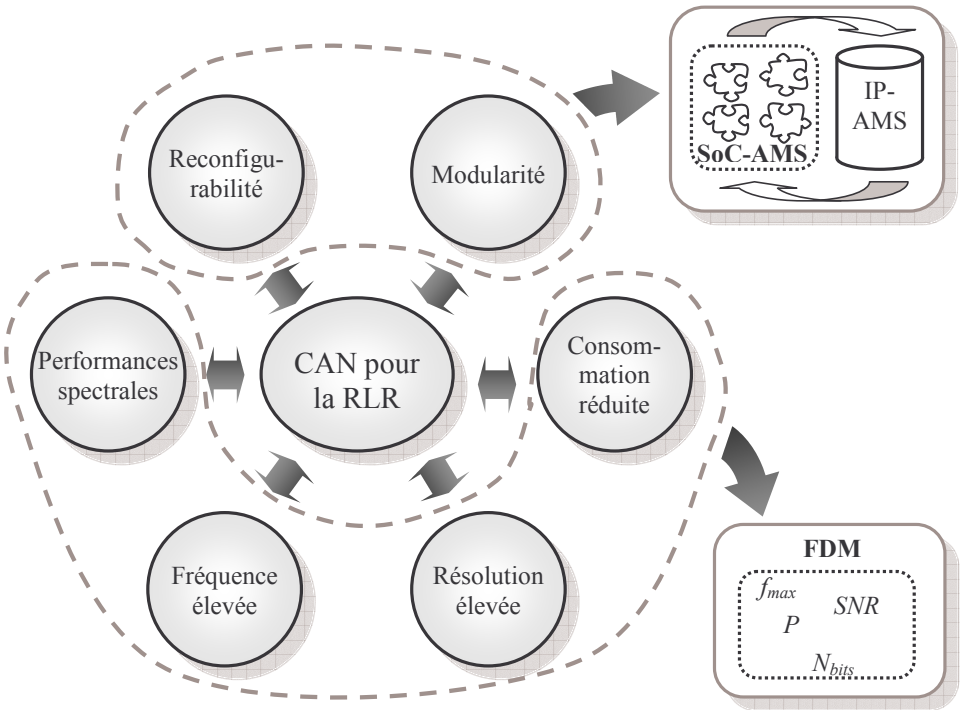


Figure 2.1. Le CAN dans le contexte RLR : besoins et développement.

Dans un premier temps, quelques généralités sur les CAN (caractéristiques, architectures) donnent les définitions utiles et posent les bases de ce chapitre. Ensuite, la mise au point d'un modèle VHDL-AMS de CAN générique incluant la définition d'imperfections est détaillée. Ces différents paragraphes permettent d'aborder le développement d'une méthode d'exploration de l'espace de conception concernant les CAN dédiés à la RLR. Cette partie présente un outil d'aide à la sélection d'architecture se basant sur un découpage modulaire de la fonction de conversion. Le concept d'IP-AMS est donc placé au centre de ce travail avec le souci de proposer une démarche de conception descendante. Les résultats de ces recherches sont alors exposés et commentés.

2.2 Paramètres pour la sélection des CAN

Avant de développer le travail d'exploration architecturale en lui-même, ce paragraphe propose d'une part de définir les caractéristiques des CAN, essentielles à cette étude, et d'autre part de décrire les techniques permettant de les mesurer.

2.2.1 Caractéristiques

2.2.1.1 SNR, SNDR, ENOB

La définition du rapport signal à bruit (SNR) d'un CAN, déjà évoquée dans le premier chapitre est rappelée ici (2.1) :

$$SNR = 6{,}02N + 1{,}76 + 10.\log\left(\frac{f_s}{2.f_{\max}}\right) \qquad (2.1)$$

N représente le nombre de bits du CAN considéré, f_s la fréquence d'échantillonnage utilisée et f_{max} la fréquence maximale contenue dans le signal à numériser.

Le terme « rapport signal à bruit » peut être ambigu dans certains cas puisqu'il est utilisé pour représenter le rapport signal à bruit avec distorsion (SNDR, SINAD ou SND pour *Signal to Noise and Distorsion Ratio*) ainsi que le rapport signal à bruit sans harmoniques (SNHR pour *Signal to Non-Harmonic Ratio*) [Std01, An2, Lin, Max]. Le calcul du SNDR exclut typiquement les cinq harmoniques les plus puissantes pour ne refléter que le bruit de fond du CAN. L'expression (2.1) correspond donc en toute généralité au SNDR.

Le nombre effectif de bits (ENOB, *Effective Number Of Bits*) répondant à la relation (2.2) représente le nombre de bits significatifs du CAN par rapport au nombre de bits physiquement disponibles en sortie du composant.

$$ENOB = \frac{SNDR - 1{,}76}{6{,}02} \qquad (2.2)$$

Si l'on se réfère à l'expression (2.1), l'ENOB reflète l'écart entre le niveau de bruit mesuré en sortie d'un CAN donné et le niveau de bruit théorique d'un CAN idéal fonctionnant à la fréquence de Nyquist et numérisant le signal sur le même nombre de bits N. Cette observation mène à conclure que l'ENOB peut dépasser la valeur N lorsque la fréquence d'échantillonnage est supérieure à la fréquence de Nyquist (voir le §2.3.4).

2.2.1.2 SFDR

Le SFDR (*Spurious Free Dynamic Range* – dynamique sans fréquence parasite) correspond à la différence entre la puissance d'une sinusoïde numérisée en pleine échelle et la puissance de la fréquence parasite la plus élevée. Cette définition est illustrée par la Figure 2.2.

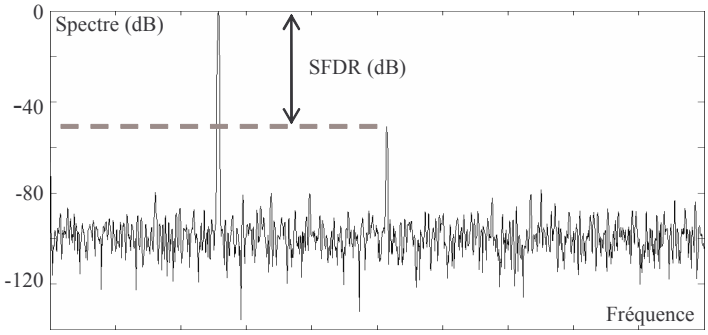


Figure 2.2. SFDR d'un CAN.

Dans [Col93] et [Pan99], le niveau des harmoniques induites par la numérisation idéale d'une sinusoïde est calculé à partir d'une décomposition en série de Fourier. Les développements relatifs à ce calcul sont présentés en Annexe A. Cette même annexe définit une formule empirique simple évaluant le SFDR d'un CAN idéal. L'optimisation par la méthode des moindres carrés utilisée fournit le résultat suivant (2.3) :

$$SFDR_{id} \approx 8{,}1\,N + 3{,}3 \tag{2.3}$$

La relation (2.3) permet de simplifier la relation (2.4) couramment utilisée [Pan99].

$$SFDR_{id} \approx 9\,N - n \text{ avec } 0 < n < 6 \text{ environ en fonction de } N \tag{2.4}$$

2.2.1.3 Fréquence d'échantillonnage

L'échantillonnage des signaux est considéré comme étant à période constante. Dans ce cadre, la fréquence de conversion maximum (f_{max}) est habituellement celle à laquelle la caractérisation est effectuée. L'observation montre que cette condition correspond généralement au fonctionnement présentant un ENOB d'environ *N-1*.
La fréquence de conversion minimum (f_{min}) correspond à la fréquence d'horloge pour laquelle le rapport signal à bruit est de 3 dB inférieur au SNR nominal (c'est-à-dire correspondant à f_{max}).
Les fréquences d'échantillonnage sont souvent exprimées en MHz ou en MSPS (*Mega Samples Per Second* – millions d'échantillons par seconde) qui sont deux unités équivalentes.

2.2.1.4 Bande passante analogique

La bande passante analogique d'un CAN est la plage de fréquences d'entrée pour laquelle la puissance spectrale du fondamental (déterminée par analyse FFT) diminue de 3 dB au maximum.
Cette notion est à distinguer de la fréquence de conversion maximum qui correspond à la fréquence de l'horloge cadençant le composant. La bande passante analogique peut être supérieure à f_{max}, ce qui permet d'utiliser le principe du sous échantillonnage évoqué dans le chapitre 1 (§1.2.3.1).

2.2.1.5 Non linéarité différentielle, non linéarité intégrale

Les grandeurs de NLI (Non Linéarité Intégrale ou INL pour *Integral Non Linearity*) et de NLD (Non Linéarité Différentielle ou DNL pour *Differential Non Linearity*) rendent compte des défauts de position des paliers de conversion d'un CAN. La NLI représente l'erreur de position d'un palier de conversion (en amplitude) et la NLD l'erreur sur la largeur d'un palier de conversion. Ces valeurs dépendent l'une de l'autre (2.5) et sont exprimées en LSB (*Least Significant Bit*), ce qui correspond à un pas de quantification.

$$NLI(i) = \sum_{j=0}^{i} NLD(j) \tag{2.5}$$

La NLI correspond au cumul des NLD précédentes, d'où son nom.
Ces notions sont illustrées sur la Figure 2.3 autour du palier de conversion *i*.

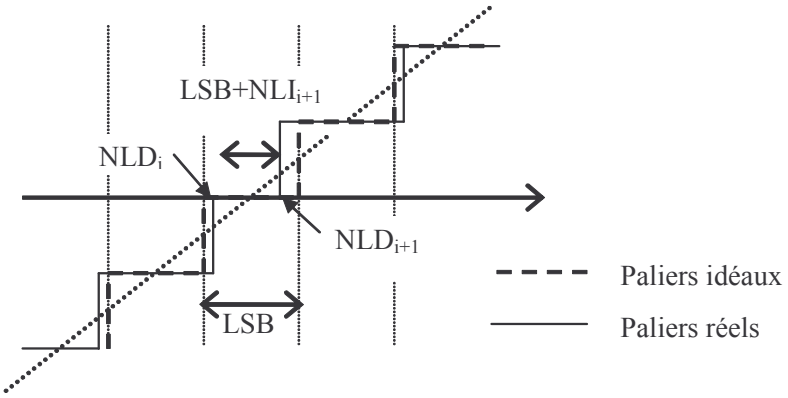


Figure 2.3. Exemple de NLD et de NLI.

2.2.1.6 Jitter d'horloge

Quel que soit le type de convertisseur, il existe une variation aléatoire du déclenchement de l'échantillonnage par rapport aux fronts d'horloge appelée *jitter* d'horloge [Kob99]. Ce paramètre est défini par son écart type en ps. Malgré sa valeur apparemment très faible (de l'ordre de quelques dixièmes de ps), il peut dégrader significativement l'ENOB d'un CAN, soit plusieurs dixièmes de bits pour une fréquence d'entrée élevée (cf. §2.4.2.3.2).

Il est possible de quantifier l'effet du *jitter* avec l'équation (2.6) en calculant la puissance de bruit qu'il induit pour un signal d'entrée sinusoïdal [Kob99].

$$P_j = A^2.\left[1 - \exp\left(-2\pi^2 f_{in}^2 \sigma_j^2\right)\right] \tag{2.6}$$

P_j est la puissance de bruit du jitter, A l'amplitude du signal sinusoïdal d'entrée, f_{in} la fréquence de la sinusoïde numérisée et σ_j la valeur RMS du jitter (écart type) de distribution gaussienne.
La présence du terme f^2_{in} indique que plus la variation du signal d'entrée est rapide, plus le *jitter* introduit de bruit dans le signal. Ce phénomène est illustré sur la Figure 2.4.

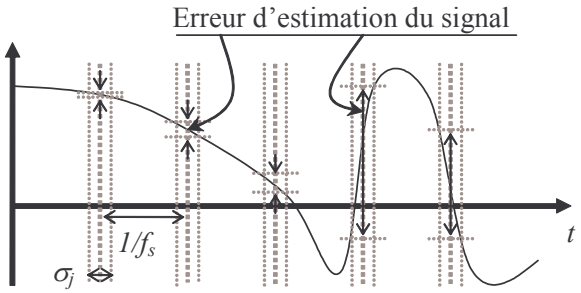


Figure 2.4. Relation entre le jitter et l'erreur de numérisation d'un signal.

Il est possible de généraliser la relation (2.6) aux signaux quelconques en utilisant une décomposition en séries de Fourier [Kob99].

2.2.2 Principes de mesure

Les grandeurs définies dans le paragraphe précédent sont obtenues à l'aide de différentes techniques de caractérisation. Elles sont réparties de la façon suivante : techniques temporelles, techniques fréquentielles et méthodes d'ajustement.

2.2.2.1 Techniques temporelles

Connaissant les caractéristiques du signal d'entrée employé, les techniques de mesure temporelles permettent d'extraire les paliers de conversion d'un CAN à partir de l'analyse d'un signal mesuré en sortie.

2.2.2.1.1 Balayage DC

La NLI et la NLD statiques (ou DC) peuvent être mesurées en appliquant un balayage DC lent en entrée du CAN et en relevant chacune des transitions. Il s'agit d'évaluer directement la position des paliers à l'aide d'un banc de mesure simple à mettre en œuvre. La comparaison entre les paliers mesurés et les valeurs théoriques idéales permet de calculer la NLD et la NLI correspondant à chaque code.
La NLI et la NLD statiques peuvent être déterminantes dans quelques applications d'instrumentation. La NLI statique diffère souvent beaucoup de la NLI dynamique, c'est-à-

dire que, soumis à des fréquences élevées, un CAN ne présente pas les mêmes caractéristiques qu'en statique. Certains phénomènes, tels que le filtrage induit par l'échantillonneur bloqueur, et des non linéarités apparaissent à de hautes fréquences. De plus, la NLI peut être soumise à un effet d'hystérésis [Rah02].

2.2.2.1.2 Rampe modulée

Afin de mesurer la NLI et la NLD en dynamique [Cru03], une solution consiste à appliquer un signal de test suffisamment rapide en entrée pour pouvoir observer l'effet des phénomènes évoqués précédemment. Une évolution de la technique de balayage DC consiste à utiliser une rampe modulée par une sinusoïde en entrée, ce qui est équivalent à une sinusoïde dont l'*offset* est incrémenté de *LSB* / *k* à chaque période d'horloge (2.7).

$$e(t) = A.\sin(\omega t + \varphi) + \frac{LSB}{k}.\frac{t}{f_s} \tag{2.7}$$

Cette méthode peut être facilement utilisée en simulation mais présente des difficultés de réalisation puisque la génération d'un tel signal s'avère complexe et la mesure peut être très sensible au bruit.

2.2.2.1.3 Méthode de l'histogramme

La méthode de l'histogramme [Rah02] consiste à comparer les densités de probabilité des signaux d'entrée et de sortie d'un CAN pour en déduire la NLD et la NLI. Le signal de test soumis en entrée d'un CAN est de forme connue. Il s'agit alors de comparer la répartition statistique des sorties du CAN (codes binaires) avec la fonction densité de probabilité (*fdp*) du signal d'entrée (Figure 2.5). Le signal utilisé est typiquement une sinusoïde.

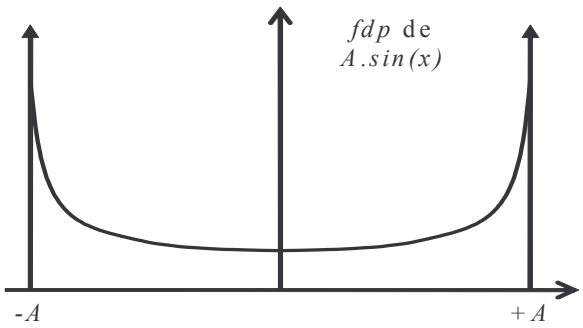


Figure 2.5. Fonction densité de probabilité d'une fonction sinusoïdale d'amplitude A.

La Figure 2.6 met en évidence le phénomène de discrétisation de la *fdp* théorique par le nombre limité de codes (2^N) d'un CAN *N* bits.

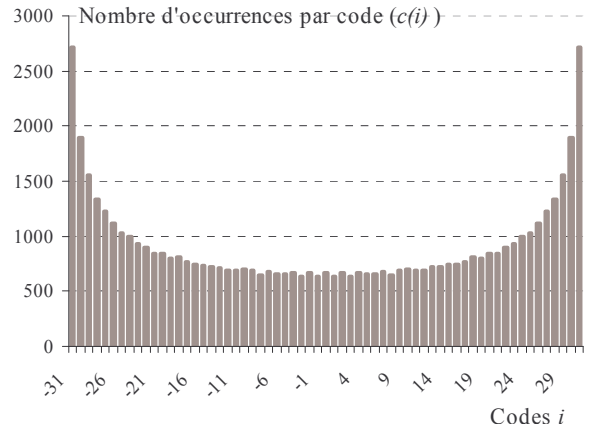


Figure 2.6. Histogramme d'une sinusoïde numérisée par un CAN 7 bits idéal.

La difficulté consiste à choisir une fréquence de signal de test adéquate. En effet, si cette fréquence est multiple de la fréquence d'horloge du convertisseur, seuls quelques codes seront utilisés par le CAN. Par exemple, pour le cas critique où $f_{clk} = f_{sig}/4$, seuls 3 à 4 codes seront activés, selon la phase du signal. Il est difficile de choisir une fréquence quelconque en étant certain de balayer tous les codes. [Rah02] indique une méthode permettant d'éviter les cas particuliers. Pour obtenir des résultats fiables, l'expérience montre que le nombre de mesures par code doit être d'au moins 100 occurrences.

La mesure consiste à comparer le nombre d'occurrences de chaque code au nombre théorique prévu pour un CAN parfait. Afin de limiter le nombre de points de l'histogramme, ces paliers peuvent être excités avec une densité de probabilité uniforme. Trois formes de signaux remplissent cette condition : le bruit blanc, la rampe périodique et le signal triangle. Pour effectuer ces mesures, la forme triangulaire peut être préférée puisque son spectre est relativement concentré autour de la fréquence fondamentale, les conditions de l'expérience sont donc plus proches de celle utilisant un signal sinusoïdal.
Connaisant la tension de référence V_{ref} et l'*offset* du convertisseur, le calcul des paliers de conversion à partir des mesures est presque immédiat dans le cas d'une probabilité d'apparition théorique des codes uniforme. Il utilise le principe de la Figure 2.7. Plus l'écart séparant deux paliers de conversion est grand, plus le code correspondant a, en moyenne, de probabilités d'apparaître. Il s'agit donc de compter le nombre de fois où le code en question apparaît pour évaluer la taille réelle du palier.

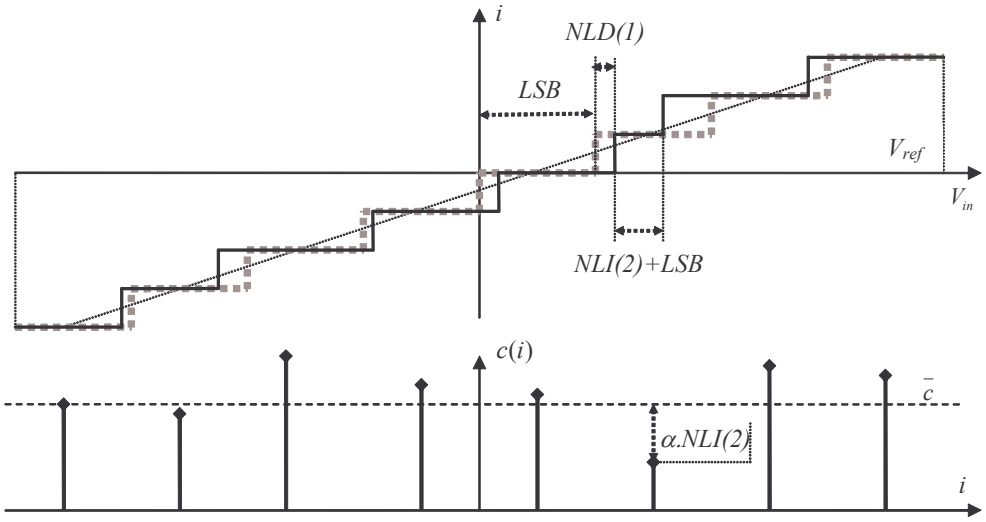


Figure 2.7. Principe de la méthode de l'histogramme avec un signal de fdp constante.

Tout d'abord, le nombre d'occurrences *c(i)* de chaque code *i* est enregistré et le nombre d'occurrences moyen $\bar{c}$ est calculé (2.8) :

$$\bar{c} = \frac{\sum_{i=1}^{2^N} c(i)}{2^N} \tag{2.8}$$

Comme l'illustre la Figure 2.7, il est possible de déduire la non linéarité différentielle du code *i* (*NLD(i)*) exprimée en LSB à partir de la différence entre *c(i)* et $\bar{c}$ (2.9).

$$NLD(i) = \frac{c(i)}{\bar{c}} - 1 \tag{2.9}$$

A partir de la NLD, la NLI est déduite grâce à (2.5). La NLI représentant l'erreur sur la position d'un palier, il suffit d'ajouter *NLI(i)* à la position théorique correspondant au $i^{ème}$ palier (2.10) pour obtenir une estimation de la position réelle d'un palier (*p(i)*).

$$p(i)_{Volts} = -V_{ref} + i.LSB + NLI(i) + offset \tag{2.10}$$

2.2.2.2 Techniques spectrales

Une des méthodes les plus communes pour caractériser un CAN est la mesure spectrale [Car00]. Le CAN est excité par un signal sinusoïdal. Le résultat obtenu est enregistré puis analysé par FFT afin d'en extraire différents paramètres.

La longueur *E* de la séquence d'échantillons analysée doit être prise en compte dans les calculs [Tro04] (Figure 2.8). Lorsqu'une sinusoïde pleine échelle est appliquée à un CAN de résolution *N* bits, le rapport théorique (2.1) peut être utilisé. Si le bruit de quantification n'est pas corrélé avec le signal (fréquence d'échantillonnage non multiple de la fréquence d'entrée), on obtient un bruit gaussien uniforme dans la bande de Nyquist. La FFT agit

comme un filtre bande étroite avec une largeur de Δf, avec $\Delta f = f_s / N$, et le seuil de bruit de la FFT se situe 10.log(E/2) en dessous du niveau de bruit de quantification large bande (c'est-à-dire 6,02.N+1,76 *dB*). Le seuil de bruit de la FFT diminue donc de 3 dB à chaque doublement de la longueur de l'échantillon. Cette réduction du seuil de bruit est le même effet obtenu en réduisant la largeur de la bande d'un analyseur de spectre numérique à une largeur de bande f_s / E.

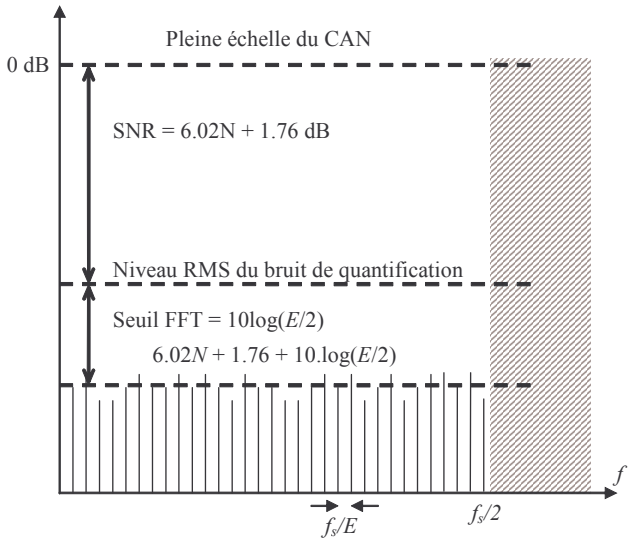


Figure 2.8. Analyse FFT d'une sinusoïde numérisée.

Le rapport signal à bruit est extrait du calcul FFT en utilisant la relation (2.11).

$$SNDR = 10.\log\left(NBW.\frac{P_{signal}}{\sum_{n=1}^{E/2} P_n} \right) = 10.\log\left(NBW.\frac{P_{signal}}{P_{bruit}} \right) - 10.\log\left(\frac{E}{2} \right) \quad (2.11)$$

NBW représente la bande de bruit de la fenêtre de pondération FFT utilisée et P_{bruit} la puissance du bruit de fond du signal, c'est-à-dire la puissance du spectre après élimination du pic de la fréquence test.

2.2.2.3 Méthodes d'ajustement 3 et 4 paramètres

Les méthodes d'ajustement (*three* ou *four-parameter sine-fit*) consistent à définir une sinusoïde théorique à partir de données numérisées [Std01, Rah02].
En utilisant un enregistrement comportant M échantillons y_1, y_2, …, y_M correspondants à des instants t_1, t_2, …, t_M, l'algorithme d'ajustement 3 paramètres estime les valeurs A_0, B_0 et C_0 en minimisant la somme suivante :

$$\sum_{n=1}^{M}\left[y_n - A_0\cos(\omega_0 t_n) - B_0\sin(\omega_0 t_n) - C_0\right]^2 \qquad (2.12)$$

A l'aide de calculs matriciels et de fonctions d'optimisation de type moindres carrés (dits *LMS* pour *Least Mean Square*), les valeurs de A_0, B_0 et C_0 sont évaluées et permettent d'aboutir à l'expression de la sinusoïde y_n' la plus proche de la séquence d'entrée y_n.

$$y_n' = A.\cos(\omega_0 t_n + \theta) + C_0 \text{ avec :}$$
$$A = \sqrt{A_0^2 + B_0^2} \text{ et } \theta = \tan^{-1}\left(-\frac{B_0}{A_0}\right) \qquad (2.13)$$

Grâce à la relation (2.13), l'algorithme indique l'amplitude, la phase et l'offset du signal numérisé. Le calcul de l'erreur entre la sinusoïde théorique issue de l'optimisation et le signal analysé permet d'estimer le SNDR.
Cette méthode nécessite de connaître avec précision la valeur de ω_0. S'il existe un décalage entre la pulsation ω_0 indiquée et la valeur réelle, le résultat d'optimisation n'est pas fiable.

Pour contourner ce problème, l'algorithme d'ajustement 4 paramètres, basé sur le même principe, estime automatiquement ω_0. Pour ce faire, il utilise un processus itératif pour converger vers la valeur réelle.

Pour s'assurer de la cohérence des résultats, l'amplitude, la phase et l'offset doivent être estimés (3 paramètres) et éventuellement la fréquence (4 paramètres). Alors que l'ajustement 3 paramètres est plus simple et plus rapide, il nécessite une bonne connaissance de la fréquence du signal analysé. L'ajustement 4 paramètres, plus lent, peut éventuellement diverger et ne fournir aucun résultat cohérent. Cependant, il est incontournable lorsque la fréquence du signal numérisé n'est pas précisément connue. Des travaux mettent en œuvre cet algorithme [Mar, Mar01] et développent des méthodes pour l'optimiser afin d'améliorer sa convergence [Fon01, Bil02].

L'amplitude et la phase du fondamental et des harmoniques peuvent aussi être obtenues par FFT mais le fenêtrage affecte les résultats estimés. L'adaptation se fait dans le domaine temporel sur des données « non fenêtrées », évitant ainsi la distorsion spectrale.

2.3 Architecture des principaux CAN

Certaines techniques classiques de conversion analogique-numérique sont présentées dans cette partie [An2, All03, Lou02, Max, Sum02, Tro04]. Les CAN flash et à approximations successives (aussi appelé SAR pour *Successive Approximation Register*) sont des exemples respectivement de parallélisation et de sérialisation de l'opération de conversion. Le CAN pipeline constitue un intermédiaire entre ces deux premières structures tant du point de vue de la technique de numérisation (semi-parallèle) que des performances. Le CAN Sigma-Delta peut être vu comme une évolution des CAN SAR et propose des performances assez proches du CAN pipeline.

Un aperçu des performances actuelles est indiqué pour chaque famille de CAN et permettra de dégager l'orientation de l'exploration architecturale développée par la suite, compte tenu des contraintes de la RLR.

2.3.1 CAN flash

Le convertisseur flash est une architecture de CAN dont le principe de fonctionnement est très intuitif (Figure 2.9). Ce circuit utilise 2^N-1 comparateurs pour fournir un résultat numérique sur N bits.

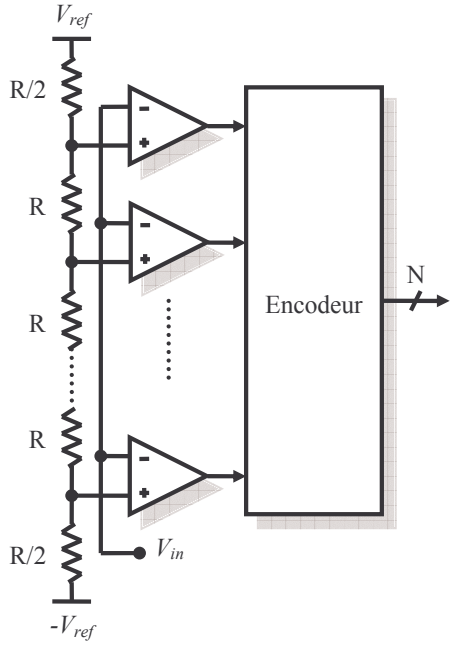


Figure 2.9. Schéma bloc d'un CAN flash.

Les seuils des comparateurs sont fixés par le réseau de résistances pour qu'ils soient équirépartis et distants d'un LSB. La quantification s'effectue en une première étape, la seconde consistant à traduire le code thermométrique issu du réseau de comparateurs en un code binaire à l'aide d'un circuit d'encodage numérique.

L'avantage majeur de ce CAN est sa capacité à travailler à des fréquences très élevées, éventuellement de l'ordre du GHz (1 GSPS). L'inconvénient principal provient du réseau de comparateurs dont la taille est doublée pour chaque bit supplémentaire. Les résolutions proposées avec les CAN flash sont donc en général de l'ordre de 8 bits afin de limiter la consommation, la taille du circuit et les erreurs de seuil dues à la difficulté de produire un réseau de comparateurs uniforme de grande taille.

2.3.2 CAN à approximations successives

Contrairement au CAN flash ou l'opération de conversion est totalement parallélisée, ce CAN est un système bouclé contenant un CNA dans sa boucle de retour (Figure 2.10).

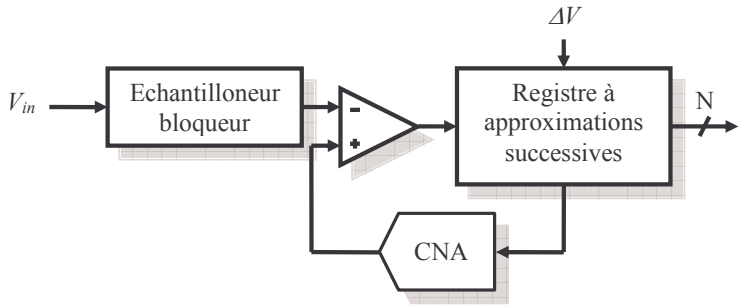


Figure 2.10. Schéma bloc d'un CAN à approximations successives.

Le rôle du registre à approximations successives (SAR) est de présenter en entrée du CNA successivement les différents bits en commençant par celui de poids le plus fort. La plage de recherche de la valeur à numériser étant divisée par 2 pour chaque nouveau bit à évaluer, ce CAN utilise le principe de la dichotomie.

Pour une résolution de N bits, la conversion s'effectue en N périodes d'horloge. Ces CAN offrent de grandes capacités d'intégration puisqu'ils ne contiennent qu'un comparateur. Ce principe ne permet pas d'obtenir des fréquences de fonctionnement élevées (inférieures à 1 MSPS), mais il représente un exemple de sérialisation de l'opération de conversion.

2.3.3 CAN pipeline (ou CAN pipeliné)

Le CAN pipeline a la particularité de répartir la quantification le long d'une chaîne de convertisseurs (Figure 2.11). Au niveau de la parallélisation des opérations, cette architecture est un compromis entre les deux solutions précédemment indiquées. En effet, alors que le CAN flash requiert 2^N-1 comparateurs et que le CAN à approximations successives n'en contient qu'un seul, le CAN pipeline représente un intermédiaire en répartissant l'effort de conversion sur M étages.

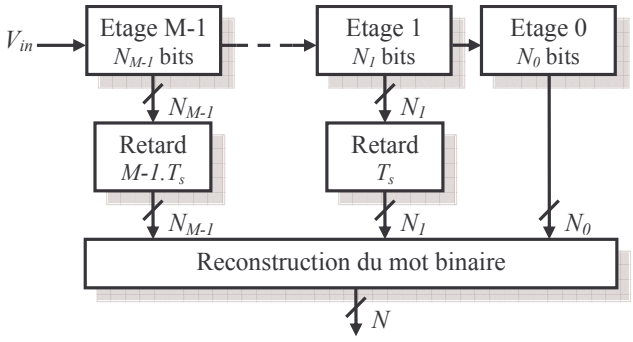


Figure 2.11. Schéma bloc d'un CAN pipeline.

Chacun des *M* étages (Figure 2.12) est un bloc de conversion élémentaire qui numérise le signal présenté à son entrée et fournit à l'étage suivant le signal d'erreur analogique (reste des opérations antérieures).

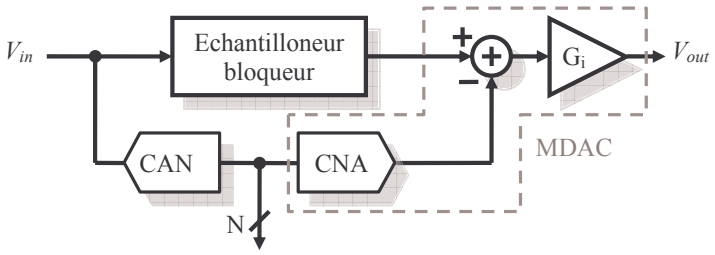


Figure 2.12. Etage de conversion d'un CAN pipeline.

Chaque étage comprend un CAN fournissant une portion du code binaire final. Ils sont généralement de type flash afin que l'information transite le plus rapidement possible le long du pipeline. L'autre partie de ces étages est appelée MDAC (Multiplying DAC) et contient un CNA, un sommateur et un amplificateur. Le dernier étage ne comporte pas de MDAC puisqu'il ne fournit pas de signal d'erreur, il est donc uniquement constitué d'un CAN.

La cohérence de la reconstruction du signal binaire est assurée par des registres à décalage synchronisant les sorties des *M* étages. Dans le cas le plus simple, la sortie correspond à une simple concaténation des mots fournis par les *M* étages.

Cette solution conduit à une architecture efficace en terme de résolution et de vitesse avec une consommation relativement faible. L'état de l'art actuel des CAN commercialisés propose des débits supérieurs à 100 MSPS avec des résolutions de l'ordre de 10 à 14 bits.

2.3.4 CAN sigma delta

La relation (2.1) indique qu'un échantillonnage à une fréquence supérieure à la fréquence de Nyquist permet d'améliorer le rapport signal à bruit. Cette technique est limitée, car f_s est bornée par f_{max} (§2.2.1.3) ce qui correspond à une limitation matérielle. Ce constat amène à étudier la structure ΣΔ (Figure 2.13), partie intégrante de la famille des CAN à

suréchantillonnage. Elle est constituée d'une boucle appelée modulateur sigma delta et d'un filtre numérique *H(z)*.

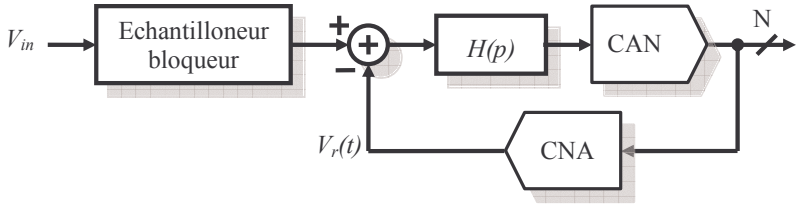


Figure 2.13. Schéma général d'un CAN ΣΔ à simple rétroaction (premier ordre).

Le cas le plus commun consiste à utiliser un simple intégrateur pour *H(z)* et un CAN 1 bit (un seul comparateur), ce qui évire l'utilisation d'un CNA en rétroaction. La sortie est constituée d'une trame de bits traitée par un filtre décimateur pour obtenir une image numérique de l'entrée à la fréquence de Nyquist.

Certains travaux traitent de l'exploration architecturale sur ce type de CAN et visent à déterminer le nombre de rétroactions optimal ainsi que *H(z)* et le nombre de bits du CNA et du CAN [Abo04, Bei03].

Cette architecture est très performante jusqu'à une fréquence d'échantillonnage de quelques MHz et permet d'atteindre des résolutions élevées allant jusqu'à 24 bits.

2.3.5 Architecture retenue

Pour des récepteurs à conversion directe monopuces, la puissance consommée et la miniaturisation sont les points les plus critiques. La coexistence de plusieurs standards de communication requiert l'utilisation de récepteurs multimodes comprenant des CAN avec des résolutions et des fréquences de conversion variables. Parmi les architectures présentées, seuls les CAN pipeline et ΣΔ représentent les solutions les plus pertinentes en terme de résolution et de fréquence.
La structure pipeline est finalement retenue car c'est celle qui se rapproche actuellement le plus des besoins de la RLR, les CAN ΣΔ ou hybrides ΣΔ–pipeline [Bro97] pouvant faire l'objet de développements similaires à ceux exposés dans la suite de ce chapitre.

2.4 Développement d'un modèle de CAN pipeline

L'exploration de l'espace de conception de la TRM nécessite l'utilisation de modèles comportementaux réalistes. Ce constat conduit à remplacer le modèle de description de CAN par tableau présenté dans le chapitre 1 par un modèle plus flexible et générique. Il doit être une aide à l'exploration architecturale en mettant en évidence certains phénomènes caractéristiques du CAN et leurs influences. Il représente également une référence permettant de valider des méthodes de test et de justifier des résultats issus d'autres méthodes de modélisation (§1.3.2.3.1, modèle LUT).

2.4.1 Le CAN pipeline

Avant d'aborder le sujet de l'exploration architecturale sur les CAN pipeline (§2.5), il est indispensable de détailler le fonctionnement de ces composants afin de définir avec précision l'espace de conception. Nous allons donc voir dans ce paragraphe les principes utiles à la compréhension de ces convertisseurs, les imperfections perturbant leur fonctionnement et quelques techniques permettant de les corriger.

2.4.1.1 Principe de fonctionnement

2.4.1.1.1 Gain inter-étage

Le paragraphe 2.3.3 traitant des principes généraux de fonctionnement des CAN pipeline indique la présence d'amplificateurs entre chaque étage de conversion. Ces amplificateurs permettent de conserver la dynamique du signal le long de la chaîne et sont dimensionnés de sorte que tous les étages aient à leur entrée un signal de même amplitude. Ce procédé permet de relâcher les contraintes sur les CAN de chaque étage. La valeur de l'amplification est fonction du nombre de bits de numérisation de l'étage en question. Un signal utilisant la pleine échelle FS du CAN numérisé sur N bits produira un signal d'erreur de quantification d'amplitude $FS/2^N$. Afin de fournir un signal d'amplitude FS à sa sortie, l'amplificateur multiplie donc le signal d'erreur par un facteur 2^N.

En tant que circuits analogiques, les amplificateurs présentent des imperfections. Celles-ci dégradent le fonctionnement global du système et sont un point critique des CAN pipeline. Ce point est développé ultérieurement.

2.4.1.1.2 Latence

La conversion est répartie le long d'une chaîne. Chacun des M étages est cadencé par une fréquence d'horloge commune f_{clk} (ou $1/T_{clk}$) et fournit un résultat à chaque période d'horloge. La conversion d'une entrée analogique est donc obtenue après un temps de latence de $M.T_{clk}$ augmenté de la durée du traitement numérique de reconstruction du mot

binaire. Afin de réduire cette latence sans perturber le fonctionnement global du système, il est possible d'utiliser une technique d'alternance des fronts d'horloge illustrée sur la Figure 2.14. Elle permet également d'économiser des ressources en réduisant la taille nécessaire pour les registres à décalages de synchronisation des données.

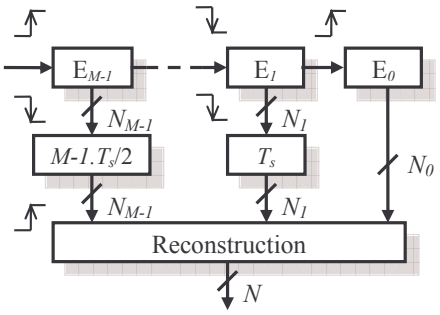


Figure 2.14. Alternance des fronts d'horloge pilotant les étages de conversion d'un CAN pipeline.

A titre d'exemple, le CAN pipeline AD9224 [An2] utilise ce principe. En effet, il comporte 4 étages et présente une latence de $3.T_{clk}$. Avec cette technique de synchronisation, le signal se propage le long des étages en $2.T_{clk}$, il reste alors une période d'horloge pour effectuer la reconstitution numérique du mot binaire.

2.4.1.2 Imperfections liées aux étages de conversion

Les amplificateurs inter-étages présentent naturellement des imperfections par rapport à un amplificateur idéal et linéaire de gain 2^M. L'influence de ces imperfections sur le comportement global d'un CAN pipeline sera vu en §2.5.2.2.

2.4.1.2.1 Erreur d'offset des paliers de conversion des sous-CAN

Cette erreur se traduit par le décalage des paliers de quantification, produit par un changement des niveaux de décision. L'effet de cette erreur sur la fonction de transfert d'un étage 2 bits est présenté sur la Figure 2.15.

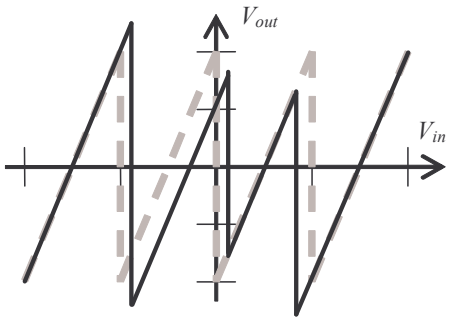


Figure 2.15. Erreur d'offset des paliers de conversion des CAN.

L'influence de ces décalages peut être supprimée dans une certaine mesure par la technique du bit de redondance (*Redundant Sign Digit* – RSD). Ce point est développé en §2.4.1.3.

2.4.1.2.2 Erreur d'offset sur le résidu

L'erreur d'*offset* sur le résidu est produite par l'ajout d'une tension constante sur le signal d'erreur provenant essentiellement de l'amplificateur (Figure 2.16).

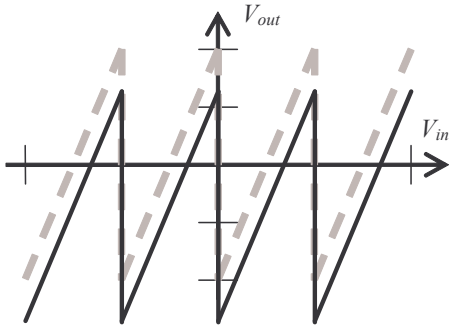


Figure 2.16. Erreur d'offset sur le résidu d'un étage.

Ce décalage constant du résidu d'un amplificateur (ou erreur DC) peut provoquer une saturation des étages postérieurs à un amplificateur donné. Cet effet peut être réduit par une technique de calibration automatique (« *auto-zeroing* ») en connectant l'amplificateur à une boucle de retour unitaire durant la phase d'échantillonnage ou en mesurant et en compensant ce décalage avec des techniques analogiques ou numériques.

2.4.1.2.3 Erreur relative de gain

Le gain d'un amplificateur n'est jamais exactement celui pour lequel il est conçu et peut différer de la valeur idéale 2^M (Figure 2.17).

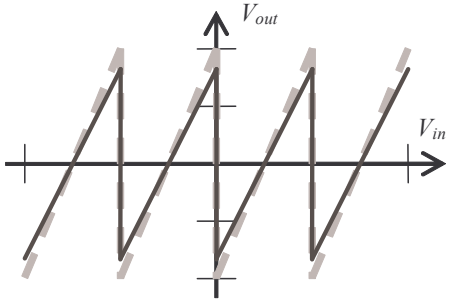


Figure 2.17. Erreur relative de gain d'un amplificateur de résidu.

Cette erreur est critique à partir du moment où le défaut d'amplification est supérieur à 1 LSB. La conséquence d'un tel décalage est l'apparition de codes manquants.

2.4.1.2.4 Non linéarité de l'amplificateur

Pour un fonctionnement à des fréquences élevées, l'amplificateur est soumis à deux phénomènes : la limitation du s*lew rate* et la bande passante (produit gain-bande) de l'amplificateur opérationnel.

La conséquence est une distorsion de la fonction de transfert (Figure 2.18). Elle peut être modélisée par une décomposition polynomiale ou plus directement par une fonction tangente hyperbolique inverse *atanh(x)* [Lot03, Nuz03].

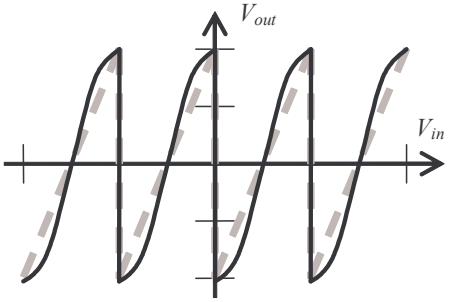


Figure 2.18. Non linéarité de l'amplification de résidu.

2.4.1.3 Correction numérique des erreurs

La plupart des convertisseurs pipeline emploie une technique dite de correction d'erreur numérique pour réduire de façon significative les contraintes de précision des convertisseurs flash et donc des comparateurs pris individuellement. Cette technique, dite du bit de redondance (RSD pour *Redundant Sign Digit*), utilise un bit de conversion supplémentaire par étage. Ainsi, le LSB du $m^{ième}$ étage a le même poids que le MSB de l'étage *m+1*. Le système devient alors plus précis mais nécessite un surdimensionnement des CAN.

Considérons le cas d'une architecture sans RSD. Si le résultat de conversion est surestimé ou sous-estimé à l'étage *m*, le signal d'erreur analogique produit se trouve décalé. L'étage *m*-1 ne possède aucun moyen de corriger cette déviation. Avec un bit de redondance, si le signal d'erreur de l'étage *m* est décalé, l'étage *m*-1 réévalue automatiquement le dernier bit de l'étage *m*.
Avec le bit de redondance, bien que chaque étage génère un mot sur N_m bits, leur résolution effective est de N_m-1 bits puisque l'amplification du signal d'erreur entre étages est de 2^{N_m-1} seulement au lieu de 2^{N_m} sans RSD. Par conséquent, cette technique consiste à appliquer en entrée des étages de conversion des signaux d'amplitude inférieure à la moitié de leur pleine échelle. Ainsi, une marge d'erreur de *FS / 2* est permise en entrée d'un étage sans le saturer. La reconstitution du mot binaire correspondant à un échantillon n'est plus une simple concaténation. Il faut tenir compte du recouvrement des bits générés par 2 étages successifs comme le montre la Figure 2.19.

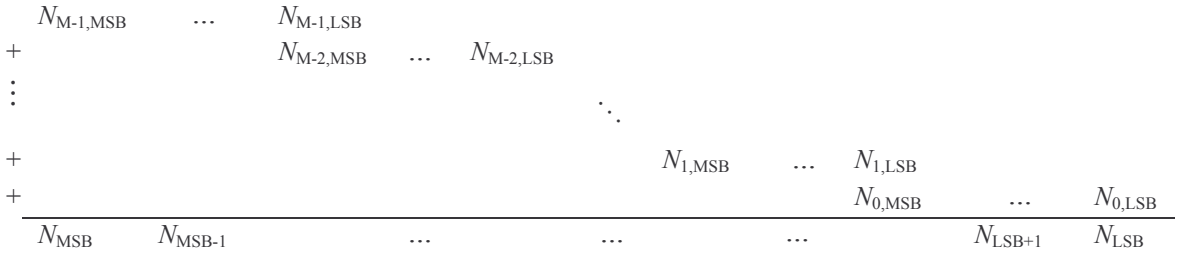


Figure 2.19. Calcul de la sortie numérique d'un CAN pipeline à partir des sorties de ses différents étages en utilisant la correction numérique des erreurs.

Prenons le cas d'un convertisseur dont la résolution des étages est successivement de 5, 3, 3 et 4 bits (partant de l'étage *M-1*). Pour le calcul de la résolution globale, il faut veiller à ne pas prendre en compte le bit de redondance qui sert à améliorer la précision d'un CAN sans augmenter sa résolution. Cette précaution ne s'applique pas au dernier étage de conversion, qui contribue donc pour N_0 bits à la résolution globale. Le nombre de bits effectifs est donc de :

$$N = N_0 + \sum_{m=1}^{M-1}(N_m - 1) = (5\text{-}1) + (3\text{-}1) + (3\text{-}1) + 4 = 12 \text{ bits} \qquad (2.14)$$

La correction d'erreurs numérique est inopérante pour le dernier convertisseur flash ce qui impose une précision supérieure pour le dernier étage par rapport aux précédents.

Le RSD corrige le décalage des paliers de conversion des sous CAN. La correction est automatique : si, par exemple, un palier est décalé vers le haut, le signal d'entrée est sous-estimé mais en même temps le résidu généré sera surestimé, les deux effets se compensant dans la limite de l'échelle de conversion de l'étage suivant. Cette source d'erreur peut être écartée dans le cadre de l'utilisation d'un bit de redondance dans la limite indiquée par la relation (2.15) [Sum02b].

$$\Delta_{off} = \pm 1 / V_{ref} . 2^{N_i + 1} \qquad (2.15)$$

Parmi les imperfections citées précédemment, l'erreur de gain ainsi que la non linéarité des amplificateurs de résidu ne peuvent pas être corrigées par cette technique. Elles ont donc une influence sur les caractéristiques globales du CAN pipeline.

2.4.1.4 Etages de conversion 1,5 bit

Le concept de bit de redondance amène à considérer que l'amplitude du signal d'entrée des étages de conversion est limitée dans l'intervalle [$-V_{ref}/2$, $V_{ref}/2$]. De ce fait, les codes les plus éloignés du centre de la gamme de tensions à convertir ont des probabilités d'apparition plus faibles que les autres. Afin de limiter le nombre de paliers de conversion, et donc le nombre de comparateurs, une solution consiste à recentrer la fonction de transfert. Cette technique est illustrée sur la Figure 2.20 pour un étage de 2 bits. En

supprimant un comparateur sur cet étage, il ne reste plus que deux paliers de conversion. L'étage obtenu est appelé « étage 1,5 bit ».

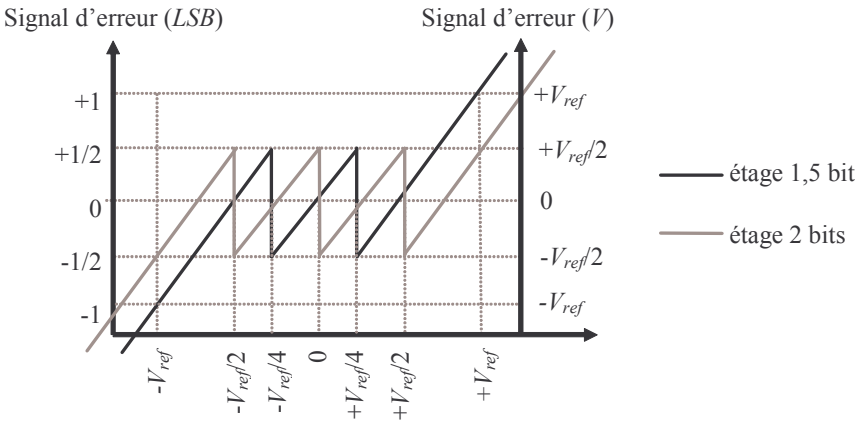


Figure 2.20. Fonctions de transfert d'un étage 2 bits et d'un étage 1,5 bit.

Cette technique est très largement appliquée pour des étages 1,5 bit et elle peut être employée pour toute résolution comme le montre [Sum02b] avec des étages 2,5 bits.
Les étages 1,5 bit amplifient le signal d'erreur par 2 et utilisent 2 comparateurs. Dans la limite définie par [-3.V_{ref} / 4, 3.V_{ref} / 4], l'efficacité d'un étage 1,5 bit est identique à celle d'un étage 2 bits tout en économisant un comparateur.
Il est préférable d'éviter d'utiliser un étage 1,5 bit en premier dans la chaîne de conversion (étage *M*-1) sous peine de ne pas bénéficier pleinement de la technique de correction RSD (Figure 2.21). Si un étage 1,5 bit reçoit un signal pleine échelle, le résidu se trouve lui aussi avec une amplitude pleine échelle. Si le 2ème étage présente également une caractéristique 1,5 bit, le signal d'erreur se propageant dans la chaîne conserve alors une amplitude *FS*. Dans ce cas, la technique RSD n'apporte aucun bénéfice.

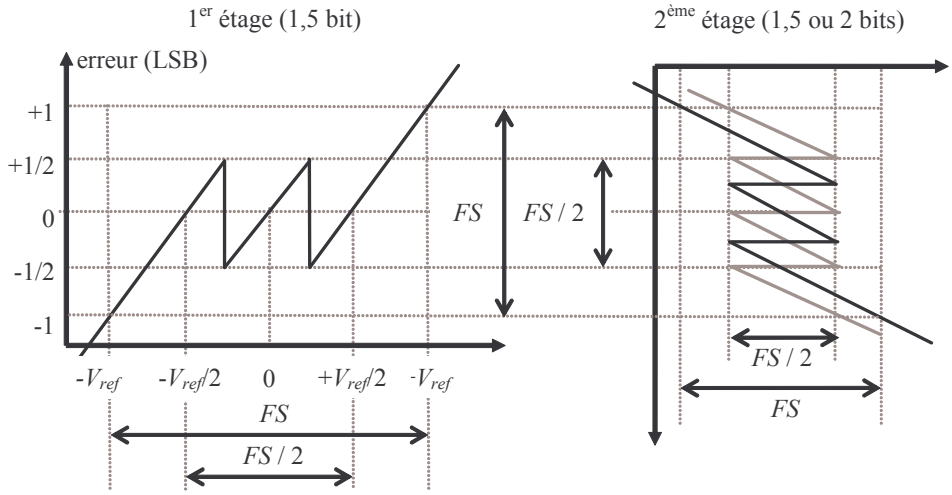


Figure 2.21. Etage 1,5 bit en tête de la chaîne de conversion.

2.4.2 Mise en œuvre

2.4.2.1 Méthodologie

La partie précédente a explicité les paramètres principaux permettant d'envisager la modélisation d'un CAN pipeline [Aco99, Aco00, Arp02, Bac96, Cra04, Gin02]. Ce travail est développé selon deux volets : le premier traite de la modélisation générique et modulaire de CAN pipeline en VHDL-AMS et le deuxième présente les méthodes de simulation mises en œuvre pour caractériser et valider les CAN modélisés.
La construction d'un modèle comportemental de CAN pipeline suit la méthode décrite sur la Figure 2.22.

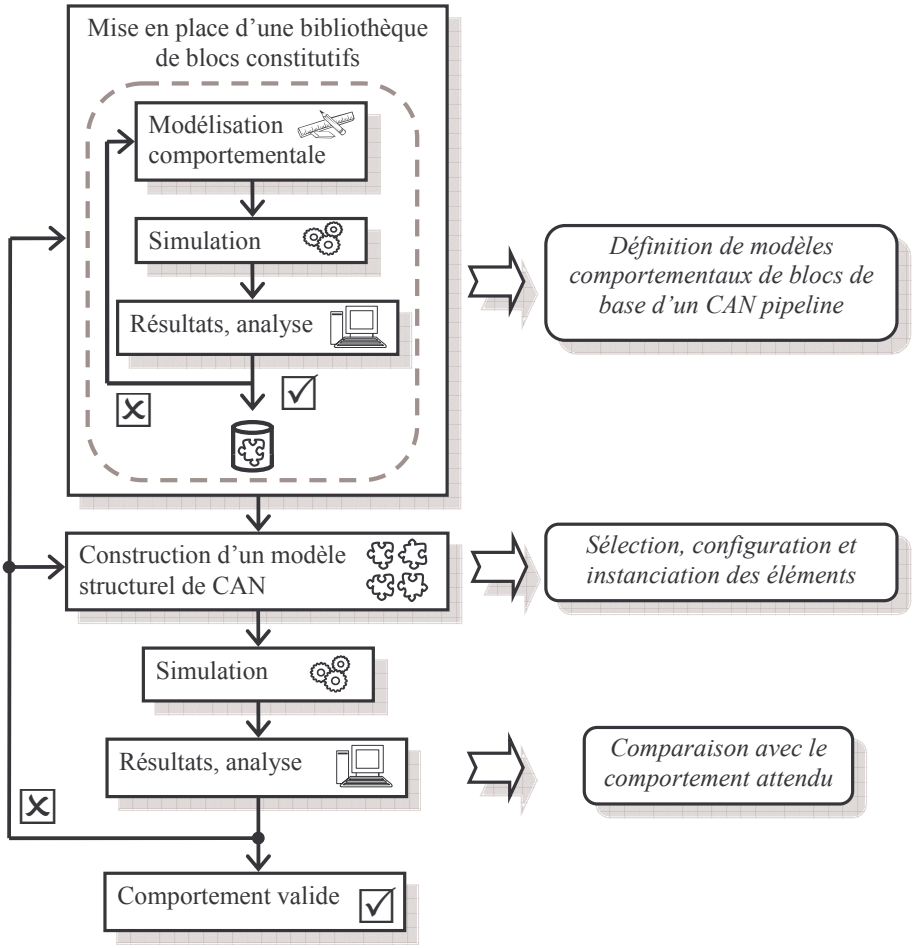


Figure 2.22. Construction et validation d'un modèle comportemental de CAN pipeline.

La constitution de la bibliothèque de modules de CAN pipeline se base sur le langage VHDL-AMS. Ces modules sont développés puis testés au niveau comportemental. Ils sont ensuite assemblés (conception structurelle) pour constituer la fonction désirée. Suivant la même démarche que pour la validation d'un module, le modèle structurel est simulé puis vérifié. Dans le cas où le modèle n'est pas satisfaisant, les améliorations à apporter doivent être effectuées soit au niveau du modèle structurel du CAN soit au niveau de la description comportementale des blocs constitutifs.

2.4.2.2 Modélisation de CAN pipeline

Cette partie retrace les étapes décrites sur la figure précédente dans la perspective de modéliser un CAN pipeline. Le modèle est constitué des blocs suivants : étage de conversion (§2.4.2.2.1), retards (§2.4.2.2.2) et module de reconstruction des mots binaires (§2.4.2.2.3). Dans le but de favoriser leur réutilisabilité, un modèle générique est utilisé pour chacun de ces blocs. Ils font l'objet d'une description comportementale et leur utilisation permet l'élaboration d'un modèle de CAN structurel de haut niveau (§2.4.2.2.4).

2.4.2.2.1 Description d'un étage de conversion

L'étage de conversion d'un CAN pipeline décrit ici doit répondre au comportement illustré sur la Figure 2.12 (étage de conversion d'un CAN pipeline) en incluant les spécificités et les imperfections présentées au §2.4.1. Pour l'utilisateur, il peut être considéré comme une boîte noire (Figure 2.23) comprenant des accès et un certain nombre de paramètres génériques.

Accès

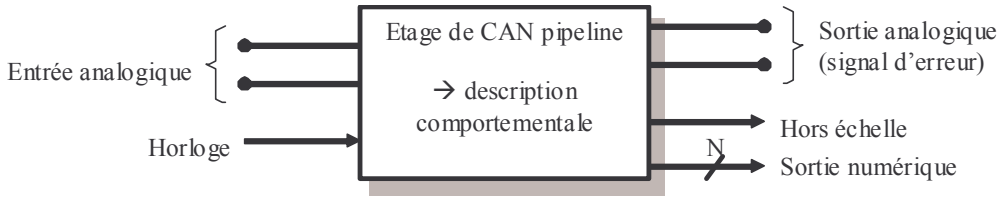


Figure 2.23. Accès d'un étage de conversion de CAN pipeline.

L'*entrée analogique* correspond soit à l'entrée globale du convertisseur, soit au signal d'erreur de l'étage précédent. Ce signal est échantillonné puis numérisé en complément à 2 pour fournir la *sortie numérique* (Figure 2.24). La différence entre l'équivalent analogique de la *sortie numérique* (V_{CNA}) et l'entrée analogique échantillonnée (V_{SH}) permet de générer la *sortie analogique* (ou signal d'erreur, V_{diff}). Lorsque le signal d'entrée dépasse la plage de tensions permise par l'étage, le signal *hors échelle* est alors actif. Toutes ces opérations sont cadencées par l'*horloge*.

Paramètres génériques

Le comportement de ce bloc est défini à l'aide de paramètres génériques représentant soit un dimensionnement soit une imperfection. La description de leur rôle permet d'appréhender le fonctionnement du bloc dans son ensemble.

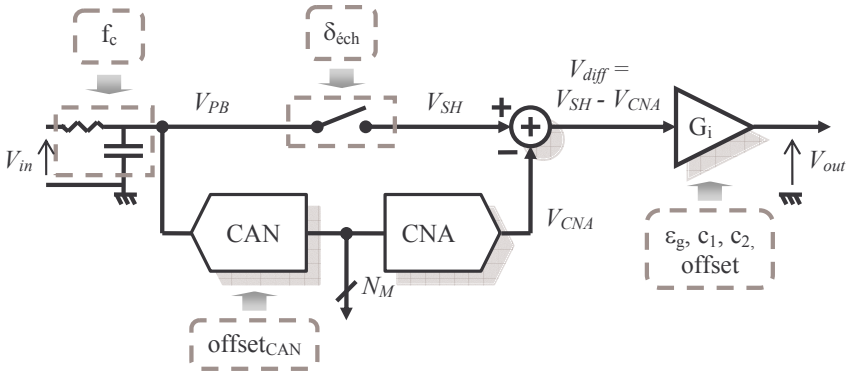


Figure 2.24. Etage de conversion de CAN pipeline avec imperfections.

Les paramètres de dimensionnement indiquent la tension de référence V_{ref} et le nombre de bits N_M de la conversion A/N (voir Figure 2.20). La fonction de transfert correspond à un étage 1,5 bit lorsque l'entier N_M est égal à 1.
Des paramètres d'imperfection permettent d'une part d'approcher le comportement réel d'un CAN et d'autre part d'évaluer l'influence individuelle de ces grandeurs sur les performances des étages et plus globalement du CAN complet. Le comportement du bloc de conversion repose sur les trois équations suivantes contenant un ou plusieurs paramètres d'imperfection.

$$V_{out} = \varepsilon_g .2^{N-1}.\left(V_{diff} + c_1 .V_{diff}^{\ 2} + c_2 .V_{diff}^{\ 3}\right) + offset \tag{2.16}$$

$$V_{out} = \varepsilon_g .k_{NL} .\tanh\left(c_1 .2^{N-1}.V_{diff}\right) + offset$$

$$\text{avec } k_{NL} = \frac{V_{ref}}{\tanh\left(V_{ref} .c_1\right)} \tag{2.17}$$

Les deux dernières équations indiquent la possibilité de modéliser la non linéarité de l'amplificateur de deux façons différentes (cf. Figure 2.18) : soit par décomposition polynomiale (2.16), soit par une fonction tangente hyperbolique (2.17) [Nuz03].
ε_g est l'erreur relative de gain de l'amplificateur (cf. Figure 2.17), c_1 et c_2 représentent les facteurs de dimensionnement de la non linéarité, et *offset* le décalage constant du résultat analogique. Il est également possible de décaler globalement les paliers de conversion du CAN d'une valeur constante $offset_{CAN}$ (cf. Figure 2.15) et d'appliquer une latence à l'échantillonneur bloqueur avec $\delta_{éch}$.
Le code VHDL-AMS d'un étage de conversion est fourni en Annexe B.

2.4.2.2.2 Retards

Les retards permettent de synchroniser les résultats des *M* étages du CAN. Ce sont de simples registres à décalage pilotés par l'horloge des étages de conversion. Les blocs de retards ont des vecteurs d'entrée et de sortie de taille générique.

2.4.2.2.3 Module de reconstruction des mots binaires

Le module de reconstruction reçoit séparément les résultats binaires des M étages après synchronisation. Si la méthode RSD n'est pas utilisée, ce bloc effectue une simple concaténation des mots reçus. Dans un cas plus général, il fonctionne selon le principe décrit sur la Figure 2.19 (calcul de la sortie avec RSD) et pondère les résultats intermédiaires selon le rang de l'étage correspondant et additionne ces M mots.
Ce bloc présente donc M entrées sur N_M bits et une sortie sur N bits, N, N_M et M étant des paramètres génériques.

2.4.2.2.4 Niveau structurel

Les blocs ainsi décrits (étages de conversion, retards, construction du résultat binaire) sont instanciés et interconnectés afin de constituer un modèle structurel de CAN pipeline tel que présenté sur la Figure 2.11. Ce dernier est générique et ses paramètres, contenus dans un *package*, sont destinés à définir les blocs constitutifs.
Une entrée analogique, une sortie numérique sur N bits et l'horloge constituent les accès du CAN pipeline.

Le *jitter* pourrait être inclus dans la description comportementale du convertisseur à ce niveau hiérarchique. Cependant, à cause de sa très faible valeur (une fraction de ps RMS [An2]), la simulation requerrait une résolution temporelle rédhibitoire. C'est pourquoi il n'est pas modélisé mais pris en compte dans la caractérisation spectrale du convertisseur en appliquant la relation donnant directement la quantité de bruit qu'il introduit (équation (2.6)).

2.4.2.3 Outils de caractérisation

La validation est la dernière étape du flot proposé sur la Figure 2.22. Ce travail ne s'appuie que sur la simulation et s'affranchit donc d'éventuelles difficultés techniques inhérentes à la réalisation (bruit, précision des appareils de mesure). Les outils développés utilisent les techniques de caractérisation vues au §2.2.2.

2.4.2.3.1 Mesure de la NLI et de la NLD

La mesure de la NLI et de la NLD d'un CAN modélisé utilise la méthode de la rampe et la méthode des triangles (§2.2.2.1). La première technique, plus rapide, est suffisante lorsque le modèle de CAN est indépendant de la fréquence du signal d'entrée. L'une et l'autre conduisent au banc de test de la Figure 2.25.

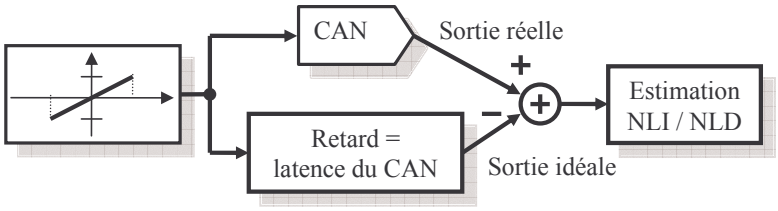


Figure 2.25. Banc de mesure de la NLI et de la NLD – méthode de la rampe.

Pour déterminer la NLD avec une précision de LSB / k, k périodes d'horloge sont nécessaires pendant que le signal d'entrée croit d'un LSB.

2.4.2.3.2 Paramètres spectraux

Les performances spectrales des CAN sont obtenues soit par FFT soit par ajustement 4 paramètres à partir de la sortie du banc de test de la Figure 1.23 (test du modèle LUT). L'extraction des paramètres nécessite deux étapes (Figure 2.26) : une simulation temporelle où un signal sinusoïdal est appliqué à l'entrée du CAN puis une analyse spectrale de la sinusoïde numérisée.

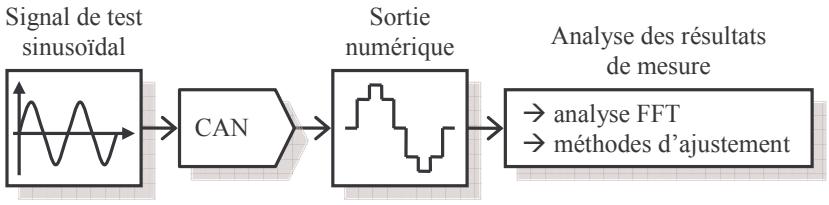


Figure 2.26. Banc de test relatif aux techniques spectrales.

Le *jitter* d'un CAN ne pouvant être considéré lors de la modélisation, il doit intervenir au niveau de l'extraction des paramètres. La méthode FFT facilite cette prise en compte. Partant de la relation (2.11), il est possible d'inclure la puissance de bruit P_{jitter} introduite par le *jitter* (2.6).

$$SNDR = 10.\log\left(NBW.\frac{P_{signal}}{P_{jitter} + P_{bruit}} \right) - 10.\log\left(\frac{E}{2}\right) \tag{2.18}$$

Connaissant la valeur du jitter et la fréquence du signal de test, le calcul FFT associé à l'équation (2.18) permet d'évaluer le SNDR du CAN. Des données relatives à un CAN simulé ont été analysées par FFT avec et sans *jitter* d'horloge. La Figure 2.27 montre l'influence d'un *jitter* d'horloge de 0,25 ps RMS sur le bruit de fond d'une sinusoïde de 10,3 MHz numérisée par un CAN 12 bits.

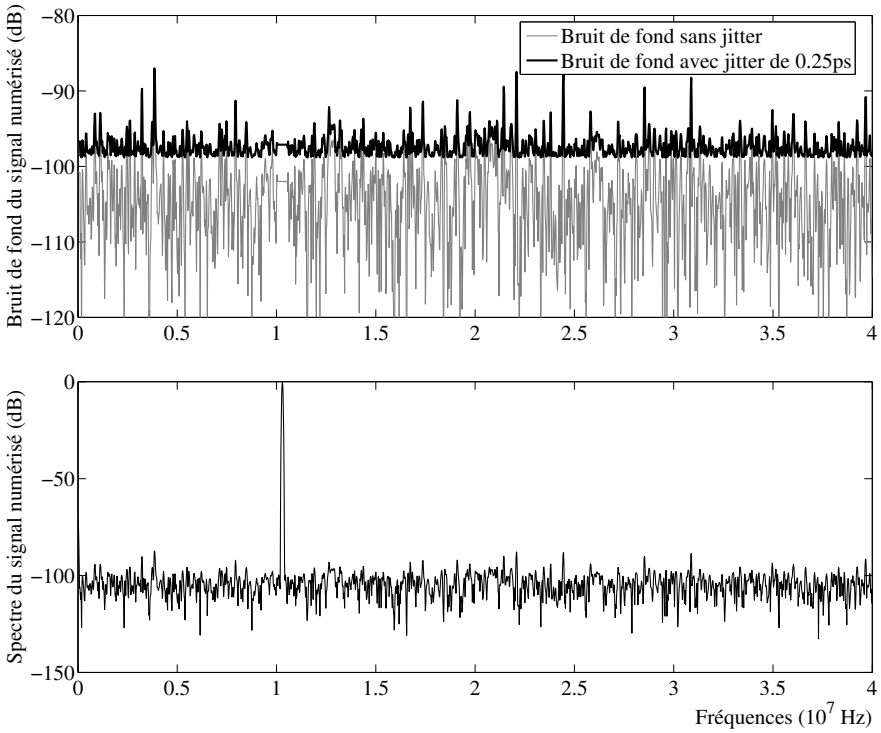


Figure 2.27. Spectre d'une sinusoïde numérisée : influence du jitter.

Les résultats correspondants aux spectres de la figure précédente sont répertoriés dans le Tableau 2.1.

	Sans jitter	Avec jitter
SNDR (dB)	72,2	67,7
ENOB (bits)	11,7	11,0
SFDR (dB)	87,1	86,8

Tableau 2.1. Performances d'un CAN 12 bits sans et avec jitter de 0,25 ps.

Nous pouvons constater que le *jitter*, même faible, dégrade considérablement le rapport signal à bruit. Par contre, le niveau des fréquences parasites n'augmente pas sensiblement sous l'influence de ce phénomène, ces pics ponctuels ne pouvant être considérés comme partie intégrante du bruit de fond.

2.5 Sélection systématique d'une architecture de CAN pipeline

2.5.1 Introduction

Comme le montre le chapitre 1, les évolutions récentes des méthodologies de conception des circuits analogiques et mixtes et la complexité croissante de ces systèmes conduisent à favoriser l'utilisation d'une démarche descendante (top-down) et non plus ascendante (bottom-up). L'idée de développer des techniques d'exploration architecturale de blocs AMS avec un haut niveau d'abstraction consiste à réduire la fracture méthodologique entre les mondes analogique et numérique. Dans ce contexte, un outil d'aide à la sélection d'une architecture de CAN pipeline est présenté ici. L'objectif final n'est pas de fabriquer un CAN mais de mettre au point une méthodologie pour aider à son dimensionnement.

L'exploration architecturale est effectuée sans se préoccuper de la technologie de fabrication, ce qui correspond aux techniques de conception descendante pour lesquelles la caractérisation d'une fonction est relative à une architecture. La comparaison entre diverses solutions conduit alors à un choix relatif.

Ces travaux reposent sur l'architecture modulaire des CAN pipeline. Les étages de conversion, organisés autour d'un CAN flash et d'un CNA, sont considérés comme des éléments de base.
De nombreux paramètres sont à prendre en compte dans la conception et l'analyse d'un CAN (résolution, surface, puissance consommée, performances spectrales, linéarité…). Ce constat mène à intégrer toutes ces caractéristiques dans une métrique unique dite fonction de mérite. Une FDM permet de comparer, à un niveau d'abstraction élevé, différentes solutions entre elles lorsque le problème de départ est soumis à un compromis entre plusieurs paramètres. Ces derniers sont pondérés en fonction de leur importance relativement à l'application visée.
Le calcul de FDM s'inscrit dans une méthodologie décrite par la Figure 2.28 [Vog03]. Le concepteur doit disposer de l'ensemble de contraintes de son application. Les performances des différentes solutions sont évaluées puis comparées avec les objectifs. Si elles ne répondent pas aux contraintes de départ, de nouvelles spécifications sont utilisées jusqu'à l'obtention d'une FDM satisfaisante.

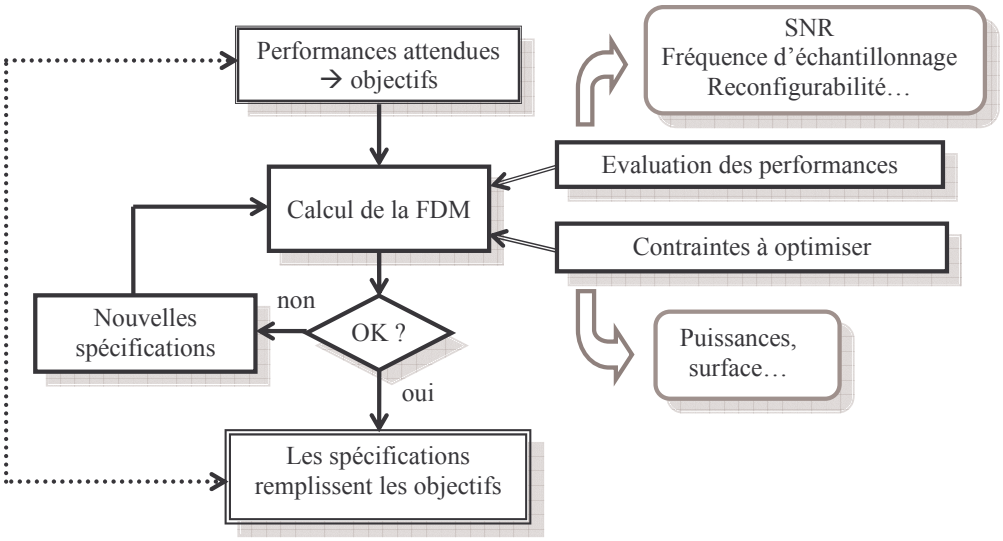


Figure 2.28. Aspects méthodologiques d'une FDM.

Le but est de sélectionner l'architecture optimale grâce à une fonction de mérite. Un comparatif exhaustif de toutes les architectures possibles est effectué. Pour ce faire, il est primordial d'évaluer efficacement les performances des différentes solutions avant de les comparer entre elles et de sélectionner la meilleure au vu de la FDM. Les contraintes imposées consistent à éviter la simulation temporelle (niveau comportemental, électrique…) et l'analyse FFT, trop coûteuses en temps de calcul [Chi05].

La simulation exhaustive de toutes les architectures est à proscrire vu le nombre de solutions à comparer. Le Tableau 2.2 indique le nombre de structures pipeline (avec RSD) en fonction du nombre d'étages mis en œuvre et de la résolution globale du CAN.

	M étages											
N bits	1	2	3	4	5	6	7	8	9	10	11	total
2	1											1
3	1	2										3
4	1	3	4									8
5	1	4	8	8								21
6	1	5	13	20	16							55
7	1	6	19	38	48	32						144
8	1	7	26	63	104	112	64					377
9	1	8	34	96	192	272	256	128				987
10	1	9	43	138	321	552	688	576	256			2584
11	1	10	53	190	501	1002	1520	1696	1280	512		6765
12	1	11	64	253	743	1683	2972	4048	4096	2816	1024	17711

Tableau 2.2. Dénombrement des architectures de CAN pipeline.

2.5.2 Fonction de mérite

2.5.2.1 Formulation

Pour chaque convertisseur, la fonction de mérite est calculée directement à partir de ses spécifications architecturales (résolution et nombre de bits par étage) et des imperfections dégradant ses caractéristiques spectrales. Cette fonction de mérite est utilisée comme une métrique permettant d'explorer l'espace de conception et de classer ainsi différentes solutions architecturales entre elles.
Les paramètres choisis permettent d'illustrer la complexité et les performances des convertisseurs. Cette FDM est un exemple développé pour la RLR pour laquelle les performances spectrales (SNDR et SFDR) sont prépondérantes [Lou02]. Le nombre de comparateurs *Comp* représente une estimation relative de la surface de silicium.
La FDM développée (2.19) [Bar05a, Bar05c], dans laquelle chaque paramètre est comparé à sa valeur idéale, est définie de la manière suivante :

$$FDM = \delta\left[\alpha\left(\frac{SNDR}{SNDR_{max}}\right)+\beta\left(\frac{SFDR}{SFDR_{max}}\right)+\gamma\left(\frac{Comp_{min}}{Comp}\right)\right] \tag{2.19}$$

$$\begin{aligned}
&\alpha = SNDR_{\lim}/SNDR_{\max}\\
&\beta = SFDR_{\lim}/SFDR_{\max}\\
&\gamma = Comp_{\min}/Comp_{\lim}\\
&\delta = 1/\left[3.(\alpha+\beta+\gamma)\right]
\end{aligned} \tag{2.20}$$

FDM représente la fonction de mérite et α, β et γ sont des coefficients de pondération. δ borne la FDM entre 0 et 1, l'unité correspondant à un CAN idéal.

$$SNDR_{\min} = 6{,}02 \times N + 1{,}76 \tag{2.21}$$

$Comp_{min}$ = $2N$+1 est le nombre minimum de comparateurs pour un CAN pipeline de résolution N avec le premier étage sur 2 bits (3 comparateurs) et les étages suivants sur 1,5 bit (2 comparateurs). N_{comp} (2.22) est le nombre de comparateurs nécessaires à un CAN pipeline quelconque.

$$N_{comp} = \sum_{i=0}^{M-1}\left\lceil 2^{N_i} - 1\right\rceil \tag{2.22}$$

$\lceil x \rceil$ représente l'entier immédiatement supérieur à x et avec $\left\lceil 2^{1,5} - 1\right\rceil = 1$.

Les coefficients de pondération α, β, γ *et* δ sont calculés de sorte que *FDM* soit bornée entre 0 et 1.
Les indices *max* et *min* indiquent les valeurs optimales (cas idéal) et les indices *lim* représentent les performances extrêmes que le CAN final doit respecter pour une application donnée.

L'exemple suivant permet d'illustrer cette relation.
L'objectif fixé par le concepteur est un CAN de 12 bits ayant un SNDR d'au moins 70 dB, un SFDR d'au moins 92,5 dB et un nombre de comparateurs maximal de 50.
La relation (2.21) indique

$SNDR_{id} = 6{,}02.N+1.72 = 73.96$ dB

L'égalité (2.3) implique

$SFDR_{id} \sim 8{,}1.N+3.3 = 100.5$ dB

Le nombre minimal de comparateurs est obtenu en considérant la chaîne pipeline la plus développée et l'utilisation d'un maximum d'étage de conversion 1,5 bit. Pour un CAN 12 bits, nous obtenons l'architecture 2/1,5/1,5/1,5/1,5/1,5/1,5/1,5/1,5/1,5/1,5.

$Comp_{id} = 3+10*2 = 23$

CAN pipeline 12 bits idéal	Objectifs	Coefficients
$SNR_{max} \sim 74$ dB	$SNDR_{lim} = 70$ dB	$\alpha = 0{,}95$
$SFDR_{max} \sim 100.5$ dB	$SFDR_{lim} = 92.4$ dB	$\beta = 0{,}92$
$Comp_{min} = 23$	$Comp_{lim} = 50$	$\gamma = 0{,}48$
		$\delta \sim 0{,}43$

Tableau 2.3. Exemple de calcul des coefficients de la FDM.

La FDM, selon les objectifs fixés, prend la forme suivante :

$$FDM = 0{,}43\left[\, 0{,}95\left(\frac{SNDR}{74}\right) + 0{,}92\left(\frac{SFDR}{100{,}5}\right) + 0{,}48\left(\frac{24}{Comp}\right)\right] \qquad (2.23)$$

Le programme de sélection d'architecture de CAN développé fonctionne selon le principe décrit sur la Figure 2.29.

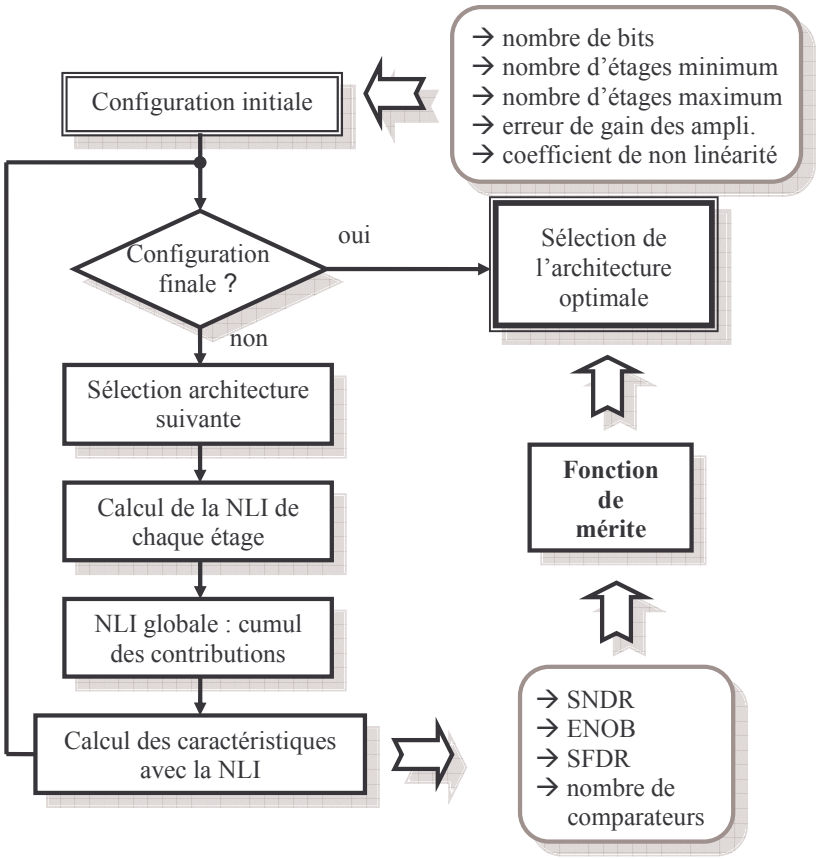


Figure 2.29. Synoptique du programme de sélection d'architecture de CAN.

Ce programme calcule les caractéristiques et la FDM correspondant à toutes les architectures de CAN pipeline dans les limites indiquées par l'utilisateur (nombre d'étages minimum et maximum). Les entrées du programme sont le nombre de bits du CAN considéré et les erreurs d'amplification entre étages. La méthode développée pour parvenir à calculer la FDM à partir d'une configuration et des imperfections d'amplification des résidus est présentée dans la partie suivante.

2.5.2.2 Calcul des facteurs de la FDM

2.5.2.2.1 Méthode

Le calcul de la FDM s'effectue en quatre étapes (Figure 2.30) [Bar05a, Bar05c]. La première consiste à définir une configuration et à préciser ses imperfections. La NLI et la NLD du CAN sont alors calculées. Ces caractéristiques permettent ensuite d'évaluer le SFDR et le SNDR de la structure considérée, le nombre de comparateurs étant quant à lui directement calculé à partir de l'architecture. La quatrième et dernière étape intègre les caractéristiques obtenues dans la FDM.

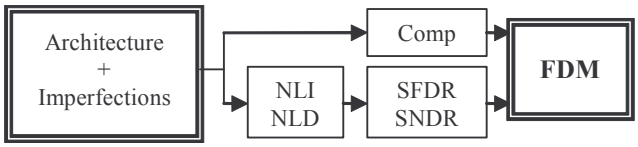


Figure 2.30. Méthodologie d'évaluation des performances.

La démarche utilisée se rapproche de [Tah04], mais la méthode d'évaluation du SFDR et du SNDR à partir de la NLI présentée ici s'effectue à un niveau d'abstraction supérieur et peut être directement réutilisée pour toute architecture de CAN.

2.5.2.2.2 Imperfections

Parmi les différentes imperfections d'un étage de CAN pipeline, seules deux d'entre elles ne sont pas corrigées automatiquement par la méthode RSD (cf. §2.4.1.2) : l'erreur de gain de l'amplificateur du signal d'erreur et la non linéarité de cette amplification. Par contre, l'erreur de décalage DC et le décalage des paliers de conversion des CAN constitutifs de chaque bloc sont corrigés dans la limite indiquée par la relation (2.15).

Erreur de gain

La fonction de transfert FT_ε, prenant en compte l'erreur relative de gain ε_{gain}, est définie à partir de la fonction de transfert idéale FT :

$$FT_\varepsilon = \left(1 + \varepsilon_{gain}\right) FT \tag{2.24}$$

Non-linéarité

La non linéarité est modélisée grâce à une fonction tangente hyperbolique inverse [Cli95, Nuz03]. L'équation utilisée est la suivante :

$$FT_{NL} = \frac{1}{\alpha_{NL}} \tanh^{-1}\left(\tanh\left(\alpha_{NL}\right) FT\right) \tag{2.25}$$

α_{NL} est un coefficient relatif à l'amplitude de la non linéarité. Pour distinguer l'erreur de gain ε_{gain} de la non linéarité α_{NL}, FT_{NL} doit être comprise entre -1 et +1. Le rôle de $1/\alpha_{NL}$ et $\tanh(\alpha_{NL})$ est de normaliser FT_{NL} pour respecter cette condition.

Le *jitter* d'horloge n'est pas pris en compte dans l'exploration puisque son influence ne dépend pas de la structure du CAN.

2.5.2.2.3 Calcul de la NLI et de la NLD à partir des imperfections du pipeline

Une simulation comportementale permet de constater que la NLI globale d'un convertisseur pipeline est égale à la somme des contributions de chaque étage à cette NLI ce qui peut s'exprimer sous la forme suivante :

$$NLI_{globale} = \sum_{m=0}^{M-1} NLI(m) \tag{2.26}$$

Le calcul de la NLI est basé sur la comparaison entre la fonction de transfert idéale d'un étage et sa fonction de transfert réelle. La Figure 2.31 illustre le calcul de la NLI à partir de l'erreur de gain.

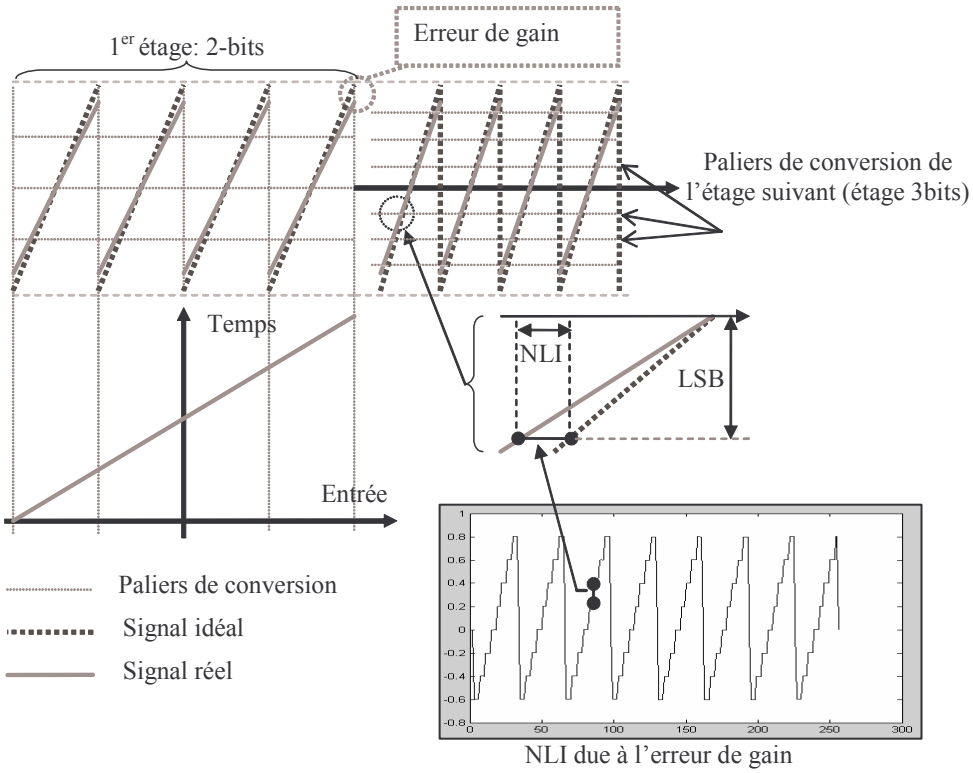


Figure 2.31. Mécanisme liant l'erreur d'amplification inter-étage à la NLI.

La fonction de transfert réelle est calculée à partir de la fonction idéale en prenant en compte l'erreur de gain et sa non linéarité. La NLI correspondant à un étage pipeline peut s'écrire :

$$NLI(i) = \left[\left(1 - \varepsilon_{gain}\right) \left(\frac{1}{\alpha_{NL}} \tanh^{-1}\left(\tanh\left(\alpha_{NL}\right) p(i)\right) \right) \right] - p(i) \tag{2.27}$$

p(i) est la position idéale du palier de conversion *i* de l'étage considéré et *NLI(i)*, la NLI correspondant à *p(i)*.

2.5.2.2.4 Calcul du SNDR à partir de la NLI

Le calcul du SNDR consiste à utiliser la démarche classique aboutissant à la formule du SNDR d'un CAN idéal (cf. équation (2.1)) et à considérer les paliers de conversion avec leur NLI correspondante. La Figure 2.32 montre le lien entre l'erreur de quantification et la NLI permettant de calculer le SNDR.

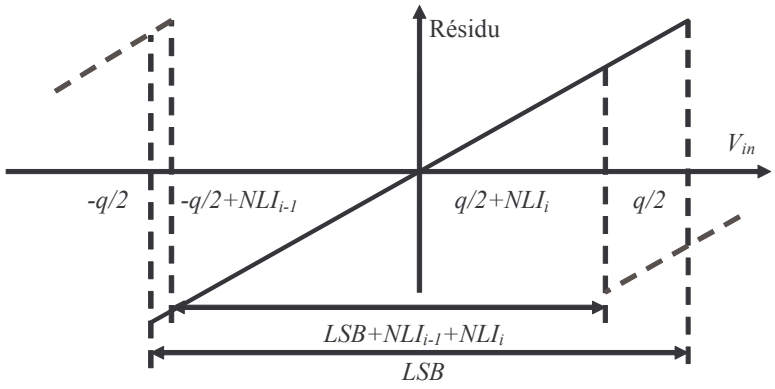


Figure 2.32. Signal d'erreur autour du code i, sans et avec NLI

Pour calculer le SNDR, il faut en premier lieu connaître l'erreur quadratique moyenne du signal de sortie par rapport à l'entrée en prenant en compte la NLI :

$$\begin{aligned}\overline{\varepsilon_i^2} &= \frac{1}{q}\int_{-q/2+NLI_{i-1}}^{-q/2+NLI_i} V_{in}^2\, dV_{in} \\ &= \frac{1}{3q}\left[\left(\frac{q}{2}+NLI_i\right)^3+\left(\frac{q}{2}-NLI_{i-1}\right)^3\right] \\ &= \frac{q^2}{12}+\frac{q}{4}\left(NLI_i-NLI_{i-1}\right)+\frac{1}{2}\left(NLI_i^2-NLI_{i-1}^2\right)+\frac{1}{3q}\left(NLI_i^3-NLI_{i-1}^3\right)\end{aligned} \quad (2.28)$$

Cette expression comprend le terme $q^2/12$ relatif à l'erreur de conversion d'un CAN idéal. A partir de celle-ci se déduit le SNDR d'un signal sinusoïdal :

$$SNDR = \frac{V_{max}^2/2}{\varepsilon^2/2^N} \text{ avec } \varepsilon^2 = \sum_{i=0}^{2^N-1}\varepsilon_i^2 \quad (2.29)$$

2.5.2.2.5 Calcul du SFDR à partir de la NLI

L'Annexe A présente la méthode permettant de calculer le SFDR d'un CAN idéal. Afin d'adapter celle-ci au calcul du SFDR d'un CAN réel, le calcul du coefficient a_k (2.30) du développement en série de Fourier n'est pas simplifié au maximum afin de conserver les termes relatifs à la position des paliers de conversion x_i et x_{i+1}.

$$a_k = \frac{2}{\pi k}\sum_{i=1}^{2^n} y_i\left[\sin\left(k.\cos^{-1}(x_i)\right)-\sin\left(k.\cos^{-1}(x_{i+1})\right)\right] \quad (2.30)$$

L'originalité de la démarche proposée ici réside dans l'idée d'additionner la NLD à chaque palier de conversion idéal lui correspondant.

$$x_i = i.LSB + NLD(i) \tag{2.31}$$

Le SFDR correspond à l'harmonique la plus élevée correspondant à la relation (2.30).

2.5.2.2.6 Nombre de comparateurs

Le calcul du nombre de comparateurs ne présente pas de difficultés. Un étage 1,5 bit contient 2 comparateurs, et un étage N bits en contient 2^N-1, ce qui conduit à la formule (2.22).

2.5.3 Résultats

La première phase de ce travail consiste à évaluer la pertinence de la FDM développée.
Une FFT effectuée sur le résultat de cette simulation permet de déduire le SNDR, l'ENOB et le SFDR du CAN testé.
La comparaison entre les résultats obtenus avec le programme FDM et les simulations VHDL-AMS porte sur des convertisseurs 8 bits. Les valeurs obtenues par simulation sont analysées soit par FFT, soit avec un programme d'ajustement 4 paramètres [Mar]. Les architectures présentées dans le Tableau 2.4 et dans le Tableau 2.5 comportent 4 étages, le premier étage (MSB) étant sur 2 bits, et le dernier étage (LSB) sur 3 ou 5 bits. Les résultats sont indiqués pour différentes valeurs d'imperfections (ε_{gain} et α_{NL}), identiques pour tous les étages.

			Simulation VHDL-AMS +				Programme FDM	
			Ajust. 4 param		FFT			
Architecture	ε_{gain}	α_{NL}	$SNDR_{dB}$	$ENOB_{bits}$	$SNDR_{dB}$	$ENOB_{bits}$	$SNDR_{dB}$	$ENOB_{bits}$
2/3/3/3	-0,02	0,4	47,75	7,64	47,62	7,63	47,69	7,63
2/3/3/3	0	0,4	44,92	7,17	44,76	7,15	44,26	7,06
2/3/3/3	-0,02	0	46,01	7,35	45,82	7,33	45,40	7,25
2/3/3/3	-0,01	0,2	49,20	7,88	49,06	7,87	49,35	7,90
2/3/3/3	-0,05	0,5	44,32	7,07	44,12	7,05	44,24	7,06
2/2/2/5	-0,02	0,4	47,21	7,55	47,06	7,53	47,34	7,57
2/1,5/1,5/5	-0,02	0,4	46,67	7,46	46,51	7,44	46,49	7,43

Tableau 2.4. SNDR obtenus en simulation temporelle et avec le programme FDM.

Ces résultats montrent une bonne concordance sur l'évaluation du rapport signal à bruit entre les trois techniques. Le SFDR ne peut être déterminé par ajustement 4 paramètres, seules les deux autres méthodes sont prises en compte dans le Tableau 2.5 pour l'évaluation du SFDR.

			$SFDR_{dB}$		Différence
Architecture	ε_{gain}	α_{NL}	FFT	FDM	Δ $SFDR_{dB}$
2/3/3/3	-0,02	0,4	57,91	57,71	-0,20
2/3/3/3	0	0,4	52,42	49,63	-2,79
2/3/3/3	-0,02	0	54,26	53,08	-1,18
2/3/3/3	-0,01	0,2	61,08	62,14	1,06
2/3/3/3	-0,05	0,5	53,07	53,38	0,31
2/2/2/5	-0,02	0,4	56,54	57,14	0,60
2/1,5/1,5/5	-0,02	0,4	56,09	54,70	-1,39

Tableau 2.5. SFDR obtenus en simulation temporelle et avec le programme FDM.

Les valeurs de SFDR estimées par les deux méthodes sont suffisamment proches les unes des autres pour valider les algorithmes de calcul utilisés dans le programme de fonction de mérite.

Le Tableau 2.6 présente des extremums obtenus avec le programme pour des CAN de 10 bits avec ε_{gain} = -1,5% et α_{NL} = 0,2 pour tous les étages [Bar05a].
$Comp_{lim}$ = 60 = 12% de $Comp_{max}$
$SFDR_{lim}$ = 75 dB = 88% de $SFDR_{max}$
$SNDR_{lim}$ = 56 dB = 90% de $SNDR_{max}$
α= 0,909 ; β= 0,885 ; γ= 0,333 ; δ= 0,470.

Architecture (MSB → LSB)	Nombre de comparateurs	SNDR	ENOB	SFDR	FDM
2\2\2\7	136	46,62	7,45	79,38	**0,69**
2\1\2\2\2\2\1\1	24	48,00	7,68	**77,08**	0,83
2\2\2\2\2\2\2\1	26	**46,26**	**7,39**	77,62	0,81
2\9	**514**	49,99	8,01	84,44	0,72
5\1\2\1\1\2	43	60,80	9,81	**87,11**	0,89
9\2	**514**	**61,97**	**10,00**	84,57	0,79
3\1\1\1\1\1\1\1	21	55,41	8,91	83,73	**0,94**
2\1\1\1\1\1\1\1\1	**19**	50,16	8,04	81,56	0,91

Tableau 2.6. Extremums obtenus avec le programme FDM pour un CAN pipeline 10 bits.

L'estimation des performances permet de comparer les solutions entre elles pour une caractéristique donnée et d'isoler rapidement l'architecture optimale par l'intermédiaire de la FDM. Le classement exhaustif des solutions possibles est rapide puisque les résultats sont estimés pour 1596 architectures en 3 minutes environ pour un PC Athlon 1800+/512Mo. Signalons que cette durée peut être portée à quelques heures pour des résolutions plus élevées : d'une part, la relation (2.30) nécessite un calcul incluant l'évaluation d'un plus grand nombre d'harmoniques et doit prendre en compte plus de paliers de conversion et d'autre part, le nombre d'architectures possibles croit rapidement avec la résolution (cf. Tableau 2.2).

Cet exemple illustre l'intérêt d'appliquer des méthodes de haut niveau d'abstraction face à la simulation électrique, puisque dans [Chi05], l'évaluation des caractéristiques d'un seul CAN se compte en jours.

L'architecture 9/2 est celle qui se rapproche le plus de la structure flash. Ses performances sont les plus satisfaisantes mais le nombre de comparateurs est également le plus élevé.
Une chaîne uniquement composée d'étages 2 bits présente les performances les moins intéressantes. Cependant, du fait de son faible nombre de comparateurs, elle ne conduit pas à la FDM la plus faible.
Le meilleur compromis, au vu de la fonction de coût choisie, est un CAN pipeline constitué d'un premier étage N bits, avec N un entier dimensionné en fonction de ε_{gain} et α_{NL}, suivi d'une chaîne d'étages 1,5 bits.

Cette tendance se confirme pour des résolutions et des valeurs de ε_{gain} et de α_{NL} différentes. Dans ce cadre, le Tableau 2.7 présente des résultats concernant des CAN 12 bits. Le premier étage peut présenter une résolution comprise entre 2 et 9 bits. Il est suivi par une chaîne d'étages 1,5 bit. Le Tableau 2.7 ne répertorie que les architectures les plus performantes au regard de la FDM.
$\alpha = 0{,}306$; $\beta = 0{,}307$; $\gamma = 0{,}167$; $\delta = 1{,}283$

→ Augmentation des imperfections →

ε_g	α_{NL}	Architecture optimale	FDM	*Comp*
-0,05	0,2	8\1\1\1\1	0,81	263
-0,04	0,2	5\1\1\1\1\1\1\1	0,81	45
-0,03	0,2	4\1\1\1\1\1\1\1\1	0,84	31
-0,02	0,2	4\1\1\1\1\1\1\1\1	0,88	31
-0,01	0,2	3\1\1\1\1\1\1\1\1\1	0,93	25
-0,005	0,05	2\1\1\1\1\1\1\1\1\1\1	0,96	23

← Vers l'architecture flash ← → Vers la chaîne la plus longue →

Tableau 2.7. Dimensionnement du premier étage pipeline en fonction des imperfections.

Ces résultats mettent en évidence le dimensionnement du premier étage en fonction du niveau des imperfections. Si ces imperfections sont élevées, la comparaison des solutions possibles conduit à sélectionner une architecture avec un étage de tête prenant en charge un grand nombre de bits afin de réduire leur influence. L'augmentation du nombre de comparateurs sur le premier étage implique donc une meilleure linéarité du CAN. Si ces mêmes imperfections sont plus faibles, l'étage de tête peut alors être réduit et la chaîne du CAN devient plus longue.
Ce type d'architecture est utilisé par un certain nombre de constructeurs. Par exemple, le CAN AD9245 [An2] est une convertisseur 14 bits dont le premier étage 4 bits est suivi d'une chaîne de 8 étages 1,5 bits et se termine par un étage 3 bits. Cadencé à 80 MHz, ce convertisseur consomme 366 mW soit quatre fois moins qu'un AD6645 du même constructeur (1,5 W), de performances équivalentes, et d'architecture 5/5/6. Selon la

relation (2.22), 38 comparateurs sont nécessaires dans le premier circuit contre 125 dans le deuxième.

2.5.4 Perspectives sur la sélection systématique de CAN pipeline

A partir d'une estimation analytique des performances de CAN pipeline et de paramètres de haut niveau, le calcul d'une FDM permet de classer différentes solutions soumises à compromis avec une métrique unique. Se basant sur cette étude, ces travaux proposent une méthode simple et rapide pour effectuer un classement exhaustif d'architectures pipeline en fonction de leurs performances estimées. Le résultat à retenir est le principe sur lequel le choix d'une architecture de CAN pipeline repose principalement sur le dimensionnement de l'étage de conversion de tête relativement aux imperfections induites par les étages de conversion.

L'aspect à développer peut être le dimensionnement de ε_{gain} et α_{NL} en fonction de la technologie et de la résolution de l'étage de conversion considéré [Lot03, Nuz03, Pan99, Tah04], ces paramètres pouvant être différents pour chaque étage. Il peut également être intéressant de rajouter aux termes de la FDM des paramètres concernant la puissance consommée [Chi05].

2.6 CAN reconfigurables

2.6.1 Application à la RLR

Pour un standard donné, la résolution et la fréquence d'échantillonnage de la conversion AN sont définies par le cahier des charges et par l'architecture de tête de réception retenue (cf. chapitre 1). L'objectif d'un concepteur de CAN pour la RLR doit donc être de minimiser la consommation électrique en fonction des besoins du système. Dans un souci d'économie de ressources matérielles, la mise en œuvre de CAN partiellement ou totalement reconfigurables peut être évoquée. Partant de ce constat, nous allons nous intéresser aux différentes possibilités de reconfiguration des CAN actuels et nous verrons comment adapter le modèle générique de CAN pipeline étudié précédemment pour le rendre reconfigurable.

2.6.2 Etat de l'art

De nombreux travaux s'intéressent à la reconfiguration des CAN dont les références suivantes constituent un échantillon : [Gul01, Liu02, Set99, Sum02a, Sum02b]. La reconfiguration permet d'atteindre trois objectifs différents : l'optimisation de la résolution pour réduire la consommation au minimum, l'adaptation de la consommation à la fréquence d'échantillonnage ou l'optimisation des performances du CAN réalisé en intervenant sur sa configuration matérielle.

Elle concerne les trois niveaux de granularité suivants (Figure 2.33) : elle s'effectue soit à l'intérieur des blocs constituants le CAN [Set99, Sum02a], soit au niveau de la gestion des blocs du CAN [Gul01, Liu02, Sum02b], soit au niveau de la fonction de conversion en mutualisant plusieurs CAN comme pour la mise en œuvre de la technique d'entrelacement (*time-interleaving*) [Sum02b].

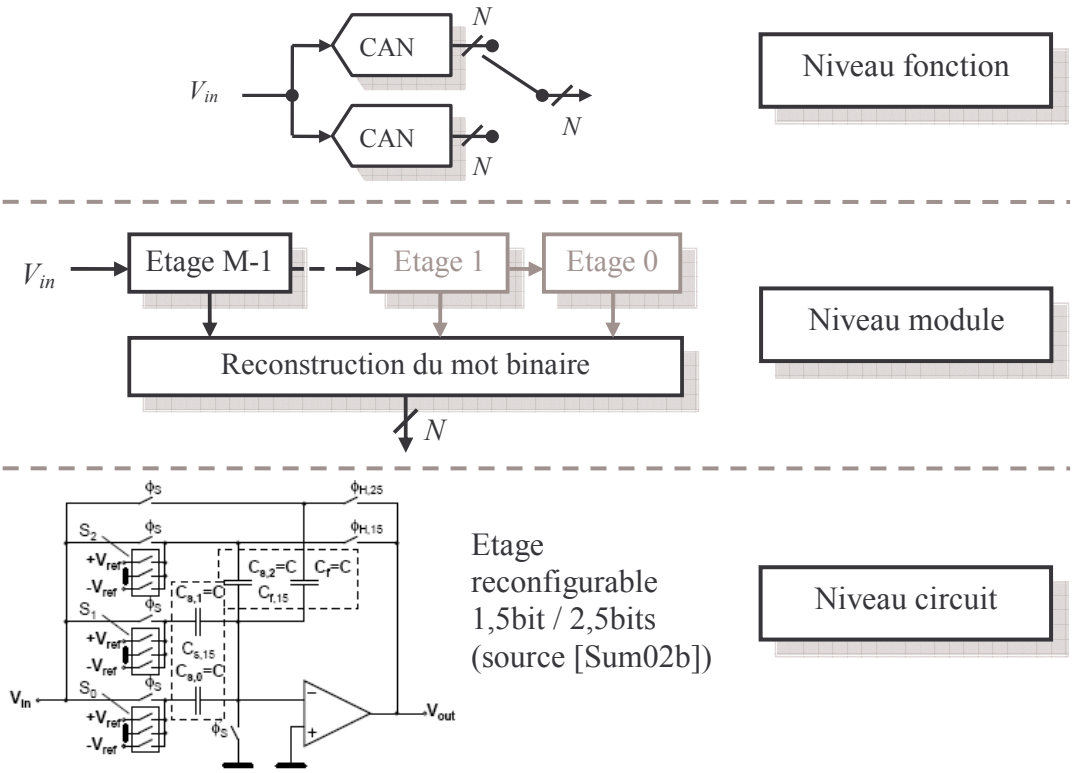


Figure 2.33. Niveaux de granularité de la reconfiguration d'un CAN.

Au niveau circuit, [Sum02a] développe une architecture de CAN pipeline 8 bits destinée à des applications de télécommunications. Celle-ci propose deux configurations : un mode 1 MSPS et un mode 15,36 MSPS. Selon la fréquence, la consommation des amplificateurs est optimisée.

Pour une application à la lecture de données (disques durs par exemple), [Set99] présente un schéma de CAN dont la résolution peut varier entre *N* bits pour le mode rapide et *N+1* bits pour le mode lent. Cette architecture est basée sur la structure flash et consiste à rajouter un étage évaluant au besoin un bit de poids faible supplémentaire.

Les CAN pipeline se prêtent bien à la reconfiguration modulaire [Sum02b]. En sélectionnant le nombre d'étages voulu, il est possible d'agir sur le compromis consommation/résolution.

Dans [Liu02], l'objectif n'est pas de modifier la résolution ou la consommation du CAN pipeline mais de rechercher la combinaison la plus performante à partir d'un groupe d'étages de conversion donné. La technique consiste à permuter les étages de la chaîne de conversion et à mesurer les performances du CAN global. Elle s'applique à un système réalisé et cette démarche peut être rapprochée de celle qui est développée analytiquement au §2.5 (Figure 2.29).

[Gul01] propose une solution hybride originale intermédiaire entre les niveaux circuit et module. Il s'agit d'un système proposant une configuration pipeline et une configuration ΣΔ. D'un fonctionnement à l'autre, l'ensemble des paramètres sont alors modifiés (SNR, consommation, résolution, fréquence d'échantillonnage).

La conversion AN peut être effectuée à partir de la mutualisation de différents CAN (niveau fonction).
La technique de parallélisation ou d'entrelacement temporel (*time-interleaving*) consiste à utiliser des CAN en alternance [Sum02b]. Si la fréquence d'utilisation maximale de ces CAN est de f_{max}, elle permet d'atteindre une cadence théorique de $k.f_{max}$ si k CAN sont utilisés en parallèle. Deux inconvénients pratiques limitent son utilisation : la puissance consommée est directement proportionnelle au nombre de CAN mis en œuvre ce qui peut aboutir à des consommations de plusieurs watts pour des convertisseurs performants [An2] et les CAN mis en œuvres expérimentalement n'ont jamais exactement les mêmes caractéristiques (*offset*, retard de déclenchement) ce qui dégrade les performances spectrales du système global [Kur01]. Des techniques de calibration et de correction d'erreur existent pour atténuer l'influence de ces imperfections [Bat02, Jin00].

Les niveaux module et fonction sont étroitement liés au concept de SoC-AMS. Il est possible d'imaginer des modules tels que des étages de conversion, des circuits numériques de configuration et de calibration associés à un réseau d'interconnexions pour constituer un CAN générique matériel. Des FPAA offrent ce type de possibilité au niveau circuit avec des performances pour l'instant modestes au regard des besoin de la RLR [An1].

2.6.3 Extension des méthodes exposées

Dans le développement de modèles de CAN pipeline (§2.4.1) ainsi que dans la méthodologie d'exploration de l'espace de conception (§2.5) présentés, la reconfiguration de l'opération de conversion n'a pas été abordée. Ce paragraphe se propose donc d'apporter des précisions sur ces deux points.

2.6.3.1 Modèle VHDL-AMS de CAN pipeline reconfigurable

Partant du modèle de CAN pipeline développé, un modèle de CAN reconfigurable en cours de simulation a été mis au point. Il permet d'intervenir sur l'architecture au niveau fonction et au niveau module. Au niveau fonction, l'utilisateur choisit le nombre k de CAN pipeline qu'il utilise en parallèle et il peut modifier le nombre de branches de conversion actives. Au niveau module, il est possible de choisir le nombre M d'étages actifs et de faire évoluer individuellement leur caractéristique (résolution N_i).
Ce modèle VHDL-AMS est reconfiguré en cours de simulation à l'aide d'un fichier de configuration écrit au préalable (Figure 2.34).

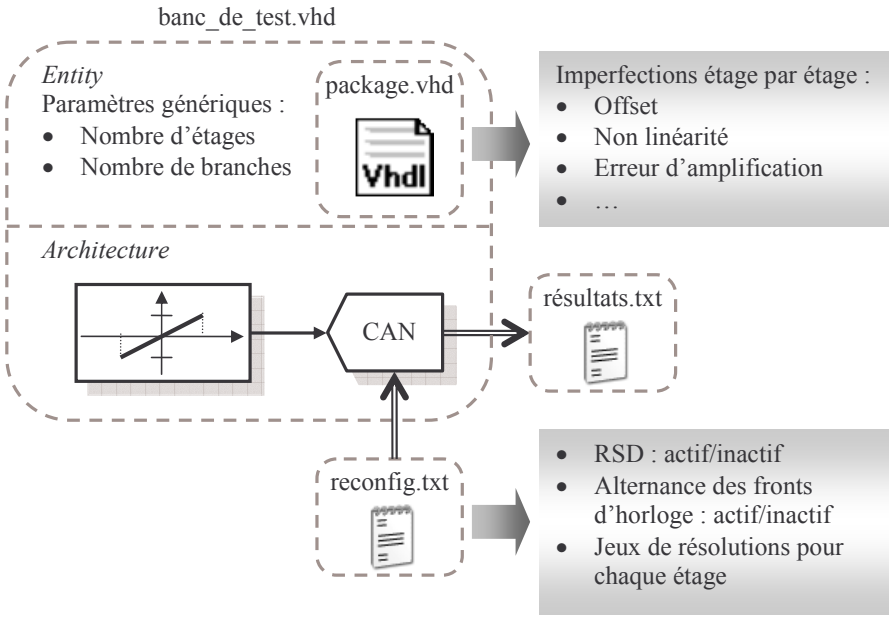


Figure 2.34. Utilisation du modèle de CAN pipeline générique.

Le choix de l'utilisation d'un fichier texte pour la reconfiguration se justifie par l'efficacité de cette solution : alors qu'un package doit être compilé pour être effectif, un fichier texte peut être modifié sans nécessiter d'intervention sur le code VHDL-AMS ni de compilation. Seuls le nombre d'étages de conversion et le nombre de branches à instancier sont choisis par l'utilisateur dans le code lui-même : le langage VHDL-AMS ne permet pas de changer les entités instanciées en cours de simulation (nombre, paramètres génériques).

L'entrelacement de plusieurs CAN permet de rendre compte de l'influence des différences entre les branches sur le signal numérisé. La Figure 2.35 présente dans les domaines temporels et fréquentiels une sinusoïde numérisée par un système comprenant deux convertisseurs entrelacés ayant des gains d'entrée différents. Ce déséquilibre entre les deux branches induit des fréquences harmoniques dont les positions, indiquées sur la Figure 2.35, sont confirmées par [Kur01]. Cette référence étudie l'impact de diverses sources de déséquilibre entre branches de CAN entrelacés telles que la différence de délai de déclenchement et la différence d'offset.

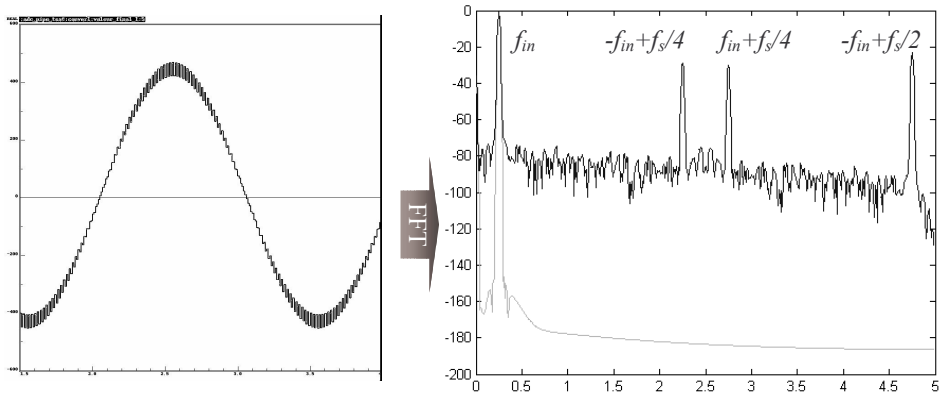


Figure 2.35. Influence de l'erreur de calibration de 2 CAN entrelacés sur le spectre d'une sinusoïde numérisée.

A partir du modèle de CAN développé, il est possible de générer un système entièrement générique de niveau hiérarchique plus élevé grâce aux fonctionnalités de modélisation structurelle proposées par le VHDL-AMS. Ce principe sera également développé dans le chapitre 3 autour de la construction de la chaîne de traitements numériques en VHDL.

2.6.3.2 Exploration architecturale

Dans le contexte de la RLR, la question de l'influence de la reconfigurabilité sur le dimensionnement du CAN se pose.

Si l'on se situe au niveau module, il est possible d'appliquer directement la méthodologie présentée en §2.5 pour chaque application visée, c'est-à-dire pour chaque standard de télécommunication à prendre en charge par le récepteur RLR. Dans ce cadre-là, l'aspect de reconfigurabilité n'influence pas l'exploration de l'espace de conception.
Le cahier des charges le plus exigeant sur le CAN conduit alors à l'architecture à mettre en œuvre. Cette architecture peut être reconfigurée pour optimiser sa consommation pour les standards moins contraignants.

Au niveau fonction, l'espace de conception est plus vaste puisqu'il faut prendre en compte le nombre de branches de convertisseurs en parallèle dans les variables du problème. De plus, le grand nombre d'imperfections à considérer ainsi que l'efficacité relative des techniques de correction rendent le problème plus complexe.
Cet aspect n'est pas développé ici, mais la mise en place d'une méthodologie telle que celle développée pour le niveau module et appliquée au niveau fonction peut être un axe de recherche intéressant.

2.7 Conclusion

La conversion analogique-numérique est une opération déterminante en RLR puisque ses performances définissent naturellement le partitionnement analogique numérique.

Les caractéristiques utiles à la sélection d'une architecture ainsi que les techniques de mesures correspondantes ont été présentées. Le choix de la technique pipeline a été discuté et a conduit à la conception d'un modèle VHDL-AMS modulaire, entièrement générique et incluant un certain nombre d'imperfections. Une méthodologie d'exploration architecturale concernant ce composant a été développée suivant le même découpage modulaire, c'est-à-dire en se situant au même degré de granularité. Elle s'appuie sur le calcul systématique d'une fonction de mérite, outil permettant de guider la sélection d'une solution soumise à un compromis. La FDM est écrite de manière à s'adapter aux besoins de la RLR : le concepteur peut pondérer les différentes caractéristiques (SNDR, SFDR, nombre de comparateurs) en fonction des priorités de son application.
Rappelons que le souci de proposer une technique de conception descendante a conduit à effectuer ce travail sans lien avec une éventuelle technologie de fabrication.

Les outils logiciels mis au point ont permis de dégager une architecture pipeline type correspondant a priori à la meilleure FDM : le premier étage à une résolution fonction de l'importance des imperfections de son amplificateur de résidu et les étages suivants ont une caractéristique dite 1,5 bit.

Dans le cadre de la RLR, la solution proposée doit être reconfigurable, c'est pourquoi le modèle VHDL-AMS de CAN pipeline a été étendu afin d'offrir la possibilité de simuler une reconfiguration dynamique de la structure au niveau module et au niveau fonction.

Cette étude est utilisée dans une perspective d'exploration de l'espace de conception de la TRM dans le chapitre 4. Elle est complétée dans le chapitre 3 par une méthodologie de synthèse systématique de la TRN, celle-ci s'appuyant sur le même principe de découpage modulaire des blocs de traitement.

Chapitre 3

Synthèse systématique de la tête numérique de réception

La tête mixte de réception radio logicielle restreinte définie dans le premier chapitre est composée d'un étage de conversion analogique-numérique et d'une tête de réception numérique. A l'image du deuxième chapitre concernant le dimensionnement de CAN, l'exploration architecturale proposée ici s'appuie sur un découpage modulaire de la TRN.

3.1 La tête numérique dans un contexte RLR

Dans le premier chapitre, le choix d'un récepteur superhétérodyne dans lequel la conversion analogique–numérique est effectuée en fréquence intermédiaire a été discuté. La transposition de la bande système en fréquence intermédiaire est de nature analogique et la transposition en bande de base est réalisée en numérique.
Nous nous intéressons dans ce chapitre aux modules constituant la tête de réception numérique (TRN). Cette première partie vise à définir l'espace de conception concernant la TRN.

3.1.1 Architecture générale

La TRN prend en charge quatre opérations distinctes [Hen99] représentées sur le schéma bloc de la Figure 3.1.
Le premier filtre(A) a pour rôle d'isoler la bande système après conversion A/N. Ce filtre passe-bande reçoit un signal numérique en fréquence intermédiaire contenant le signal modulé autour de la porteuse. Il atténue les fréquences n'appartenant pas à la bande du standard de télécommunication considéré et limite le repliement dû à l'échantillonnage de fréquence f_s.
Le démodulateur(B) transpose le signal modulé en bande de base. Le canal à traiter dans la bande système est alors positionné à 0 Hz. A cette étape de la chaîne de réception, le signal est en bande de base et comprend tous les canaux de la bande système. Le débit binaire est toujours f_s.
Le canal à traiter est isolé des autres grâce au filtre de sélection de canal(C). Le gabarit passe-bas correspondant doit être très sélectif, la bande réservée aux canaux étant généralement étroite relativement à la bande système.
La dernière étape, avant de fournir les données binaires aux modules de traitement en bande de base, est la conversion de fréquence d'échantillonnage(D) (CFE). La CFE permet de modifier le débit binaire afin de l'adapter au standard considéré. Ce débit passe donc de f_s correspondant à l'échantillonnage du CAN à f_{bb}, imposé par le cahier des charges.

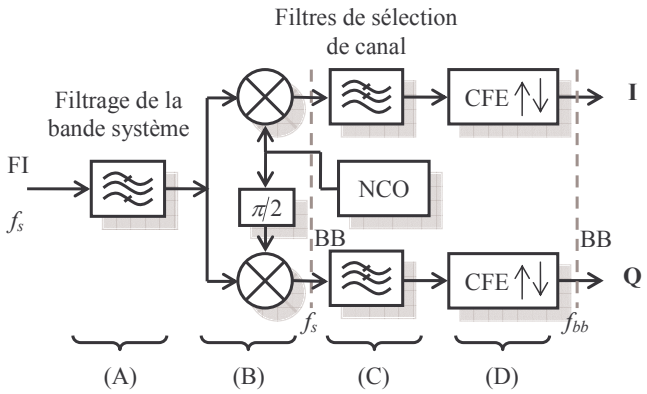


Figure 3.1. Architecture générale de la TRN.

3.1.1.1 Démodulation

La transposition en bande de base s'appuie sur une architecture hétérodyne [Mit02]. Elle est composées de deux multiplieurs (un par voie) et d'un générateur numérique de fonctions sinusoïdales (NCO) afin d'extraire du signal reçu ses composantes réelle et imaginaire et de traiter celles-ci en tant que deux signaux réels distincts.
Une autre architecture de démodulation possible est l'algorithme CORDIC (*Coordinate Rotation Digital Computation*). Il repose sur un principe itératif et nécessite la mise en œuvre de registres à décalage, d'additionneurs et de soustracteurs. Il a été développé à l'origine pour calculer les coordonnées cartésiennes d'un vecteur subissant une rotation d'un angle arbitraire. Il peut donc être utilisé en continu pour effectuer une rotation de phase du signal complexe reçu par la TRN et opérer ainsi la transposition en bande de base. Cette solution n'est pas développée ici car l'effort matériel correspondant à ce système est équivalent à 3 multiplieurs [Mit02]. Elle n'apporte pas *a priori* d'économie face aux deux multiplieurs auxquels s'ajoute la mémoire nécessaire à la définition des fonctions *sinus* et *cosinus* du NCO dans l'architecture de la Figure 3.1.

3.1.1.2 CFE et filtrage du canal

Cette partie intitulée CFE et filtrage du canal met en évidence l'intérêt de coupler ces deux opérations de la TRN dans un même module appelé filtre multifréquence (ou *multirate filter*). Seule la théorie est présentée dans cette partie. Des topologies de filtres correspondantes sont développées dans le §3.2.1.2.

3.1.1.2.1 Changement de cadence d'un facteur entier

Le principe de CFE le plus simple consiste à augmenter ou à diminuer la fréquence d'échantillonnage en la multipliant (ou respectivement en la divisant) par un facteur entier. Il conduit aux notions d'interpolation et de décimation, utiles à la mise en œuvre de facteurs de CFE rationnels et irrationnels.

Interpolation

L'opération d'interpolation se décompose en deux parties : l'insertion et le filtrage. L'insertion est l'opération qui consiste à intercaler des échantillons de valeur nulle entre deux échantillons consécutifs de la suite initiale *x(n)*. Le facteur entier *I* correspond à l'insertion d'un nombre *I-1* de zéros pour chaque échantillon d'entrée. Le débit binaire initial de f_x s'élève donc à $f_y = I \,.\, f_x$.

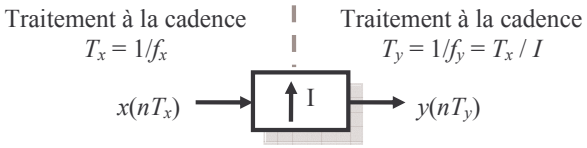


Figure 3.2. Opération d'insertion.

L'effet de l'insertion dans le domaine fréquentiel est représenté par la relation (3.1).

$$Y(f) = \frac{f_y}{f_x}.X(I.f) = \frac{1}{I}.X(I.f) \tag{3.1}$$

Le sur-échantillonnage permet de compresser l'information dans le domaine fréquentiel sans perdre d'information.
Afin de remplacer les zéros de l'insertion par des valeurs non nulles, un filtrage de fonction de transfert *H(z)* est appliqué à la suite $y(nT_y)$ après l'opération d'insertion (Figure 3.3).

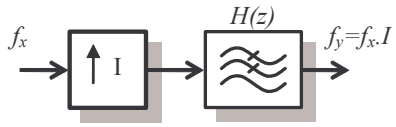


Figure 3.3. Schéma bloc d'un filtre d'interpolation.

Ce principe fournit un signal sur-échantillonné. Pour sur-échantillonner un signal d'un facteur *I* sans le dégrader, une solution consiste à cascader une insertion de facteur *I* et un filtre de bande passante [-*I* / 2 ; *I* / 2] en apportant un gain *I*.

Décimation

La décimation (Figure 3.4) de facteur *D* est une opération qui consiste à augmenter la période d'échantillonnage de la suite initiale *x(n)* d'un rapport *D*. Le facteur *D* correspond à la suppression de *D-1* échantillons dans une suite de *D* échantillons d'entrée. Le débit binaire initial de f_x est donc réduit à $f_y = f_x / D$.

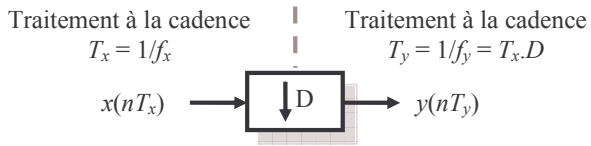


Figure 3.4. Opération de décimation.

L'effet de la décimation dans le domaine fréquentiel est représenté par la relation (3.2) :

$$Y(f) = \frac{f_x}{f_y}.X(f/D) = D.X(f/D) \tag{3.2}$$

Le sous-échantillonnage résulte de la mise en cascade d'un filtre de réponse *H*(*z*) et de l'opération de décimation (Figure 3.5).

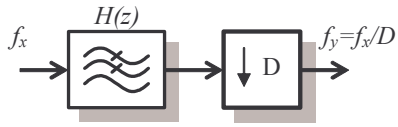


Figure 3.5. Schéma bloc d'un filtre de décimation.

$H(z)$ correspond à un filtre anti-repliement limitant la bande de fréquences occupée par le spectre de $x(nT_x)$, la réduction de fréquence d'échantillonnage entraînant un risque de superposition des différentes harmoniques du signal sous-échantillonné.

3.1.1.2.2 Changement de cadence d'un facteur rationnel

La mise en œuvre d'un facteur de CFE rationnel est issue directement de la combinaison des principes d'interpolation et de décimation, ce qui conduit aux schémas de principe proposés sur la Figure 3.6.

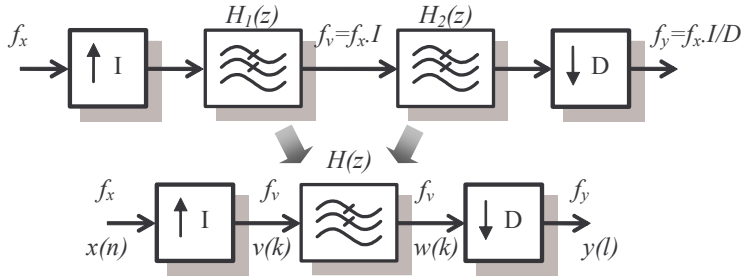


Figure 3.6. Schéma bloc d'un filtre de CFE de facteur rationnel.

Il est important de placer l'interpolation en première position dans la chaîne de traitement afin d'éviter le recouvrement et de préserver les caractéristiques spectrales de l'entrée *x(n)*. Il faut veiller à l'optimisation du filtre *H(z)* dont la fréquence d'utilisation est différente de celle des données d'entrée *x(n)*. Le filtrage s'opère donc à une cadence f_v plus élevée que le débit f_x du signal d'entrée *x(n)*. Les équations (3.3) mettent ce phénomène en évidence.

$$\begin{aligned} V(f_v) &= H(f_v).X(f_v.I) \\ &= \begin{cases} I.X(f_v.I) & pour \quad 0 \le |f_v| \le \min(1/(2.D);\, 1/(2.I)) \\ 0 & sinon \end{cases} \\ &= \begin{cases} \dfrac{I}{D}.X\left(\dfrac{f_y}{D}\right) = \dfrac{I}{D}.X\left(\dfrac{I}{D}f_x\right) & pour \quad 0 \le |f_v| \le \min(1/2\,;\, D/(2.I)) \\ 0 & sinon \end{cases} \end{aligned} \tag{3.3}$$

3.1.1.2.3 Changement de cadence d'un facteur quelconque

Pour certaines applications de traitement numérique du signal, il est nécessaire d'évaluer les valeurs d'un signal intercalées entre les échantillons d'entrée existants $x(n)$ à une fréquence ne correspondant pas à l'un des cas explicités dans les deux parties précédentes (facteurs entier et rationnel). Ce besoin conduit à la notion de filtres numériques continus en temps, ou filtres à délai fractionnaire.

Le rôle de ces filtres, illustré sur la Figure 3.7, est de calculer les nouveaux échantillons $y(i) = y_a(t_i)$ pour des instants arbitraires $t_i = (r_i + \mu_i).T_x$. Les échantillons de sortie sont déterminés par l'intervalle (ou délai) fractionnaire μ_i et l'entier r_i. C'est une opération de

convolution entre le signal d'entrée $x(n)$ et les échantillons de la réponse impulsionnelle à temps continu $h(t)$ du filtre en question qui délivre un signal $y(i)$ à la nouvelle fréquence d'échantillonnage.

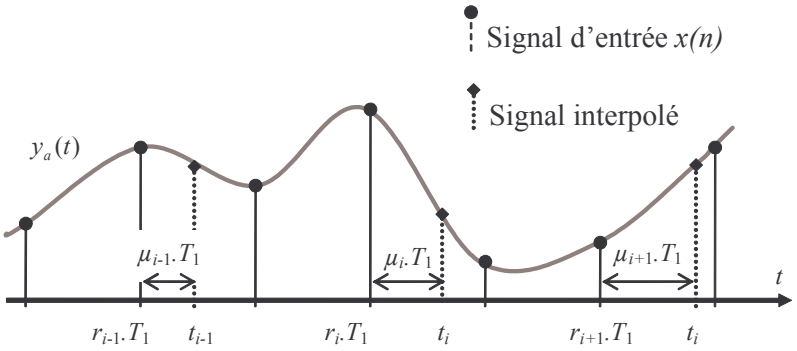


Figure 3.7. Illustration de l'opération de CFE dans le cas général (facteur quelconque).

Les paramètres d'entrée r_i et μ_i sont utilisés pour déterminer l'instant t_i du $i^{ème}$ échantillon de sortie tel que $y(i) = y_a(t_i)$ et $t_i = (r_i + \mu_i)T_1$. t_i permet de déterminer les deux paramètres r_i (3.4) et μ_i (3.5), constantes relatives au facteur de CFE à réaliser.

$$r_i = \lfloor t_i / T_1 \rfloor \tag{3.4}$$

$$\mu_i = t_i / T_1 - \lfloor t_i / T_1 \rfloor \tag{3.5}$$

Connaissant r_i et μ_i, le signal $y(i)$ peut être calculé par le filtre d'interpolation (3.6).

$$y(i) = \sum_{k=-N/2}^{(N/2)-1} x(r_i - k)\,h(k, \mu_i) \tag{3.6}$$

$h(k, \mu_i)$ est la réponse impulsionnelle du filtre d'interpolation et N (pair) sa longueur.
L'échantillon existant, précédent l'instant d'interpolation t_i est situé en $r_i.T_1$. En se basant sur l'emplacement de cet échantillon, les échantillons existants situés en $r = r_i - N / 2+1$, $r_i - N / 2 + 2, \ldots, r_i + N / 2$ sont utilisés.
La durée séparant t_i de $r_i.T_1$ est une fraction de T_1 définie par l'intervalle fractionnaire μ_i. Ce paramètre est *de facto* compris dans l'intervalle [0 ; 1).
Dans la convolution, μ_i détermine la valeur des coefficients $h(k, \mu_i)$: la réponse impulsionnelle dépend donc de l'instant t_i d'où l'utilisation du terme « filtre continu en temps ». La réponse impulsionnelle $h(k, \mu_i)$ doit être optimisée pour conserver l'égalité $y(i) = ((n_i+\mu_i).T_1)$ quelle que soit la valeur de μ_i.

3.1.2 Réalisation

3.1.2.1 Reconfigurabilité, paramétrisation

Pour des applications multistandards typiques en RLR, les contraintes sont les suivantes. Les facteurs de CFE peuvent être irrationnels, ce qui ne limite pas la recherche de solutions aux seules théories vues en §3.1.1.2.1 et §3.1.1.2.2 (CFE de facteur entier ou rationnel) mais suppose l'utilisation de systèmes plus complexes et dans le même temps reconfigurables. Au sein d'un même standard, les canaux adjacents à celui qui doit être traité peuvent être beaucoup plus puissants (de l'ordre de 80 à 100 dB supérieurs au canal désiré). Le système combinant CFE et filtrage de canal doit donc être relativement sélectif en fréquence. Enfin, pour travailler à des cadences élevées (de l'ordre de quelques dizaines de MSPS), l'implémentation matérielle doit être la plus simple possible et fait l'objet d'une attention particulière. L'étude théorique seule masque certaines difficultés de mise au point, et l'optimisation des systèmes en question en vue de leur réalisation est développée dans une large partie de ce chapitre (§3.2 et §3.3).

3.1.2.2 Cible matérielle : composants candidats

Les premières recherches sur la réalisation d'applications radio logicielle se sont principalement focalisées sur les processeurs de traitement numérique du signal (DSP). Les DSP représentent une technologie relativement simple à mettre en œuvre mais les performances de tels systèmes ne sont pas suffisantes pour prendre en charge les débits imposés par les télécommunications mobiles de troisième génération [Mit02]. Des travaux récents décrivent des applications RLR sur FPGA, le principe de la reconfiguration dynamique (changement de configuration matérielle en cours de fonctionnement) étant de plus en plus étudié et connu [Del05, Kou02].
La Figure 3.8 représente les différentes technologies de traitement numérique du signal et les capacités qu'elles offrent. D'un côté, les circuits dédiés (ASIC) ne peuvent répondre qu'à un nombre limité de fonctions mais sont souvent très performants en terme de consommation et de fréquence d'utilisation puisqu'ils sont optimisés pour une application bien définie. A l'autre extrémité de l'éventail de possibilités, les DSP opèrent des traitements du signal sous forme logicielle, ils sont donc très flexibles mais offrent des performances plus réduites. Le compromis entre ces deux solutions est l'utilisation de FPGA qui offrent à la fois des performances intermédiaires avec une grande capacité de configuration [Mit02, Moe03].

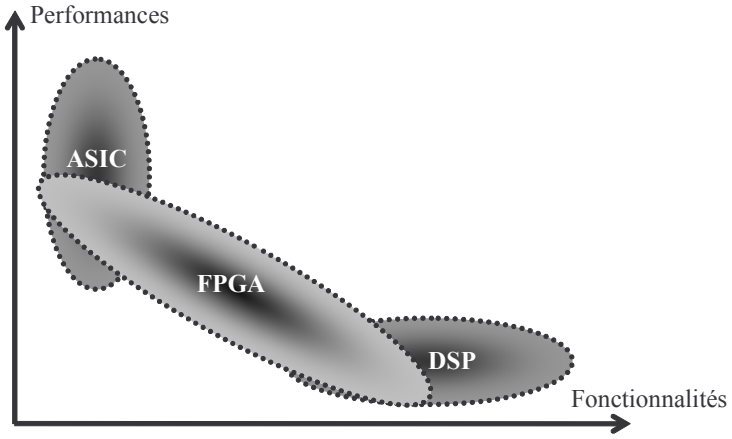


Figure 3.8. Comparaison qualitative des technologies candidates (source : [Mit02]).

Alors que les DSP ont été commercialisés dans les années 80, leur essor s'est accéléré avec l'utilisation des compilateurs C pour remplacer la conception basée sur l'assembleur. Des outils dédiés ont ensuite démocratisé leur utilisation.
Les FPGA, issus d'une autre génération technologique, se sont diffusés dans les années 90. Ils bénéficient des avancées méthodologiques en utilisant la notion d'IP et nécessitent l'utilisation d'outils dédiés. Les applications à implanter sur FPGA peuvent être spécifiées dans un premier temps dans un langage de description de haut niveau (dit HDL pour *Hardware Description Language*) tel que le Verilog ou le VHDL.
Par rapport aux DSP, les FPGA permettent une utilisation à des niveaux de granularité allant du système à la porte logique et laissent donc une grande liberté au concepteur. De plus, les possibilités de parallélisation des opérations en technologie FPGA facilitent la gestion des applications. C'est le besoin de performances de la TRN qui guide le choix vers ces composants. Notons que pour des traitements complexes requerrant de grandes capacités de calcul à des fréquences de l'ordre de quelques MSPS, la question de l'utilisation conjointe DSP et FPGA se pose. L'étude de la distribution des traitements à effectuer entre les deux cibles, appelé couramment *codesign* (conception conjointe logicielle–matérielle), est un sujet de recherche qui n'est pas développé ici et qui concerne directement les traitements en bande de base dans un contexte RLR [Len03].

3.1.2.3 Vue d'ensemble de la technologie FPGA

La Figure 3.9 situe le FPGA dans le contexte SoC avec la mise en place d'une bibliothèque d'IP numériques et l'associe aux besoins de la TRN en radio logicielle restreinte [Cum99, Moe03].

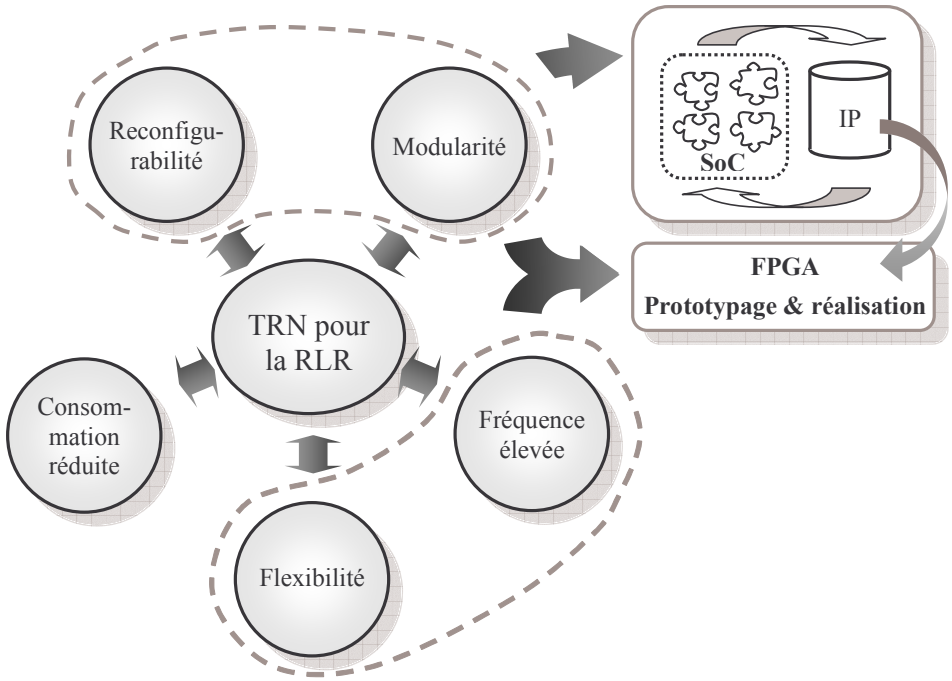


Figure 3.9. Besoins méthodologiques et matériels de la TRN.

Les architectures de FPGA sont multiples et chaque fabricant propose différentes séries. Ils disposent tous de logique distribuée, c'est-à-dire de cellules numériques élémentaires programmables à la base du principe des FPGA. Ces cellules sont connectées entre elles grâce à un réseau d'interconnexion également programmable. A ces deux principes de base s'ajoutent des modules dédiés, également accessibles via le réseau d'interconnexion. Ces modules font des FPGA des composants correspondant au concept de SoC (cf. chapitre 1) : les SoPC (*System on Programmable Chip*). Selon les modèles, les FPGA peuvent disposer de mémoires de type RAM (pour réaliser des FIFO par exemple), de fonctions arithmétiques précâblées telles que des multiplieurs et d'accès ou port paramétrables (direction, fréquence, tension). Certains proposent des circuits de gestion d'horloge permettant de synthétiser différentes fréquences à partir d'un cristal unique et de stabiliser les horloges (*deskewing*). Les composants les plus récents incluent éventuellement des processeurs précâblés (PowerPC dans les Virtex II Pro [Xil], Arm 922T dans les Apex [Alt]).

Il appartient au concepteur d'un système soit de dimensionner son application en fonction des ressources disponibles sur son composant, soit de sélectionner le FPGA adapté à ses besoins (nombre de portes, nombre de broches, fonctions précâblées…). Dans le chapitre 4, un Virtex II 1 million de portes (Figure 3.10) [Xil] sera mis en œuvre pour y implanter en partie une TRN développée à l'aide des méthodes et outils présentés dans ce chapitre.

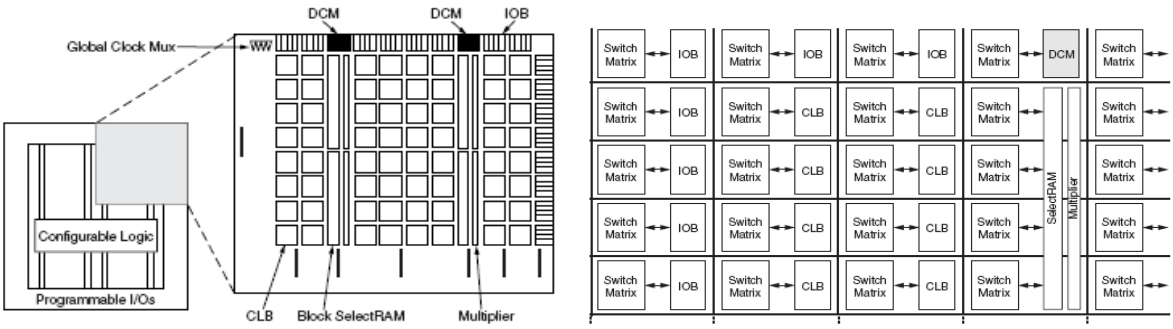


Figure 3.10. Vue générale d'un Virtex II et de son réseau d'interconnexions (source [Xil]).

3.1.3 Objectif : des spécifications à l'implantation matérielle

Les étapes décrites dans le flot suivant (Figure 3.11) [Bar05c] vont être détaillées tout au long du chapitre.

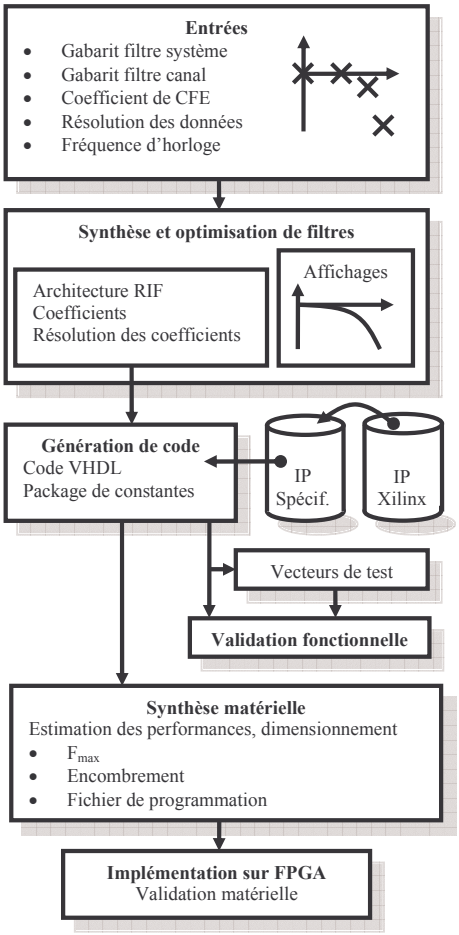


Figure 3.11. Flot de conception de la TRN complète.

L'objectif est d'automatiser ce flot de conception de tête numérique et de permettre au concepteur de spécifier ses objectifs et d'obtenir toutes les informations nécessaires à son implémentation matérielle. Le flot est conçu de telle sorte que les étapes de conception intermédiaires sont transparentes [Bos02].

Il comporte deux étapes principales :

- l'optimisation haut niveau de la TRN fournit un code VHDL synthétisable à partir des spécifications ;
- la synthèse matérielle s'opère à l'aide de l'outil Leonardo [Men] intégré dans le flot ISE [Xil] qui fournit les fichiers de configuration du FPGA à partir des codes VHDL.

3.2 Espace de conception

Le §3.1 (la tête numérique dans un contexte RLR) a permis de définir les besoins méthodologiques et de dégager les principaux axes de cette étude. Les contraintes de la RLR favorisent à l'heure actuelle le choix des FPGA comme technologie d'implantation. Le VHDL est le langage utilisé pour la description des fonctions à mettre en œuvre.
Ce paragraphe définit les différents systèmes pouvant répondre à la description de la Figure 3.1 et il fournit également des critères permettant d'isoler des solutions par rapport aux contraintes de l'application (§3.2.1). Le développement d'une IP générique favorisant l'optimisation matérielle et la reconfigurabilité des filtres est également proposé (§3.2.2).

3.2.1 Délimitation de l'espace de conception

Ce paragraphe isole un ensemble de solutions architecturales satisfaisant aux principes de portabilité, de généricité et de reconfigurabilité inhérents à la RLR. Il représente une réflexion sur une mise en œuvre systématique et efficace des fonctions constituant la TRN.
L'étude de l'espace de conception est menée de sorte à développer les fonctions de la TRN en proposant des solutions flexibles, c'est-à-dire en ne préjugeant pas d'un cahier des charges en particulier. Ce travail expose donc des architectures permettant de réaliser des gabarits de filtrage quelconques et tout type de facteurs de CFE.

3.2.1.1 Contraintes à respecter

3.2.1.1.1 Filtrage

L'exploration des solutions de filtrage est l'objet d'une attention particulière puisque ce sont les blocs de la TRN les plus complexes et les plus exigeants en terme de ressources logiques. Afin de ne pas limiter l'utilisation des outils d'optimisation de filtres mis au point, l'ensemble des systèmes de filtrage doit *a priori* répondre à toutes les spécifications. Ainsi, les gabarits de filtre peuvent respecter différentes formes : passe-bas, passe-bande, passe-haut, stop-bande et éventuellement multi-bande.
Ces systèmes doivent également prendre en charge la CFE dans le cadre du filtrage du canal à démoduler. Des techniques pouvant répondre à tous les facteurs de CFE doivent être proposées.

3.2.1.1.2 Critères de choix

Le choix d'une architecture de filtre est sujet à compromis. En effet, il ne s'agit pas seulement de concevoir un système pouvant répondre à toutes les contraintes, il faut également optimiser la quantité de ressources nécessaire à l'implémentation.
Dans l'espace de conception proposé, deux niveaux de complexité sont utilisés : le fonctionnement individuel de quatre types de filtres est tout d'abord explicité (§3.2.1.2)

avant d'aborder l'étude de la mise en cascade de plusieurs filtres en tant que degré de liberté supplémentaire dans l'exploration architecturale (§3.2.1.3).

3.2.1.2 Filtres utilisés

L'espace des solutions a été volontairement limité aux filtres répertoriés dans le Tableau 3.1 en fonction de la littérature [Hen99a, Hen00a, Hen00b].

Structure de filtre	Facteur de CFE	Compacité (multiplieurs)	Contrôle de la fonction de transfert	Utilisation
RIF direct	Aucun	-	++	Pas de CFE
CIC	Entier	++	--	Difficile à utiliser seul réponse peu conformable
RIF polyphase	Entier	-	++	Facteurs de CFE entiers
Filtre de Farrow	Quelconque	--	++	Aucune restriction

Tableau 3.1. Architectures de filtres présentées et leur utilisation.

Les filtres numériques proposés sont à réponse impulsionnelle finie (RIF). Ils offrent l'avantage d'apporter un déphasage linéaire dans la bande passante ce qui est indispensable pour le type d'application visé. L'inconvénient principal par rapport aux filtres à réponse impulsionnelle infinie (RII) est leur encombrement plus important pour un effort de filtrage équivalent.
La conception de systèmes effectuant une CFE de facteur entier ou rationnel est simple puisqu'elle se résume par l'utilisation de filtres interpolateurs et décimateurs tels que des RIF polyphases (§3.2.1.2.3) [Pro96] ou des filtres CIC (*Cascaded Integrator Comb,* §3.2.1.2.1) [Hog81]. Les filtres d'interpolation, plus complexes, peuvent assurer une CFE quelconque (§3.2.1.2.4) [Bab02, Far88, Hen00b].

3.2.1.2.1 Filtres CIC

Les filtres CIC [Hog81, Don] sont des filtres de décimation ou d'interpolation de facteur entier. Ils ont la particularité de n'utiliser aucun multiplieur (Figure 3.12) ce qui rend leur réalisation efficace en termes de fréquence d'utilisation et de compacité.
Un filtre CIC d'ordre N se compose de deux chaînes : une suite de N intégrateurs et une suite de N blocs *comb* qui correspondent à un calcul de dérivée lorsque $M = 1$. Sa réponse en fréquence correspond aux équations (3.7) et (3.8).

$$H_{CIC}(z) = H_I^N(z) H_C^N(z^R) = \frac{\left(1 - z^{-RM}\right)^N}{\left(1 - z^{-1}\right)^N} = \sum_{k=0}^{RM-1} z^{-k} \tag{3.7}$$

$$\left|H_{CIC}(f)\right| = \left|\frac{\sin(\pi M f)}{\sin(\pi f / R)}\right|^N \approx \left|R M \frac{\sin(\pi M f)}{\pi M f}\right|^N \text{ pour } 0 \le f < \frac{1}{M} \tag{3.8}$$

H_{CIC} dépend de R, le facteur de décimation ou d'interpolation, de N, le nombre d'étages de filtrage et de M, le retard de rebouclage des blocs *comb* également appelé délai différentiel. En considérant un jeu de valeurs donné N, $|R|$ et M, la réponse en fréquence d'un filtre CIC possède les mêmes caractéristiques qu'il soit interpolateur ou décimateur.

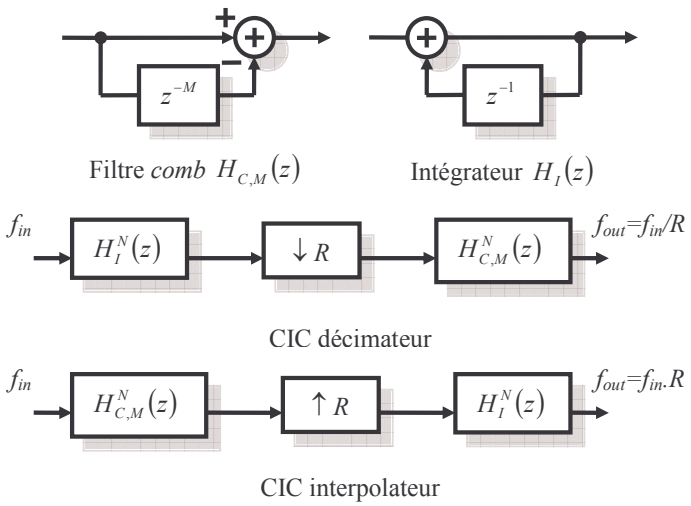


Figure 3.12. Schémas bloc des filtres CIC interpolateurs et décimateurs.

Les filtres CIC sont sujets à deux limitations majeures : ce sont des filtres passe-bas et leurs réponses en fréquence sont échelonnées puisqu'elles ne dépendent que des valeurs entières N, R et M. Le gabarit de filtrage des filtres CIC n'est donc pas ajustable et l'optimisation se fait au plus près des contraintes de départ. Il faut également prendre en considération les bandes repliées par le filtre dans le cas d'une utilisation en décimateur (Figure 3.13).

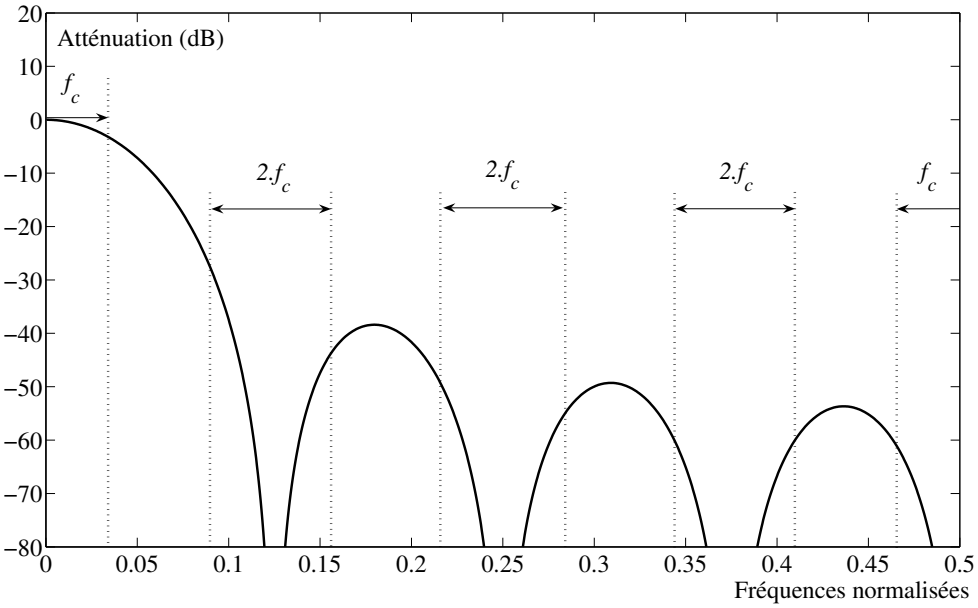


Figure 3.13. Réponse en fréquence et repliements d'un filtre CIC décimateur (R = 4, M = 2, N = 3).

La fréquence de coupure est définie par f_c. Si des composantes du signal d'entrée sont présentes dans les bandes de repliement ($0{,}5.R / k \pm 2.f_c$ avec $1 < k < R$), elles peuvent perturber le signal dans la bande passante.

3.2.1.2.2 Filtres RIF directs

Contrairement aux filtres CIC, les filtres RIF directs disposent de coefficients permettant d'obtenir une réponse en fréquence optimisée. Cette flexibilité induit un encombrement matériel non négligeable dû à l'utilisation de multiplieurs (Figure 3.14).

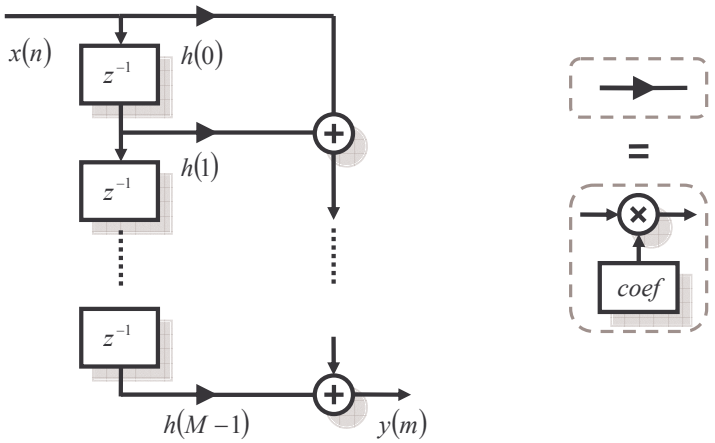


Figure 3.14. Schéma bloc d'un filtre RIF direct.

Le débit binaire de sortie est égal au débit d'entrée ce qui signifie que ces filtres ne peuvent pas directement être utilisés en tant que systèmes de CFE. Leur réponse temporelle, composée des coefficients $h(k)$, est définie par la relation suivante :

$$y(n) = \sum_{i=0}^{N-1} x(i)h(n-i) = \sum_{i=0}^{N-1} h(i)x(n)z^{-i} \tag{3.9}$$

Afin de construire un système convertissant la fréquence d'échantillonnage avec des filtres RIF directs, l'idée la plus immédiate consiste à associer un filtre et un décimateur et/ou un interpolateur (Figure 3.6 et Figure 3.15). On peut ainsi obtenir un coefficient de conversion de fréquence d'échantillonnage entier ou rationnel.

Figure 3.15. Utilisation d'un filtre RIF direct pour la mise en place d'un système de CFE.

Cette solution, bien que simple, est peu efficace [Pro96]. En effet, l'insertion produit I-1 zéros entre chaque échantillon d'entrée. Si I est grand, la plupart des échantillons à traiter par le filtre sont nuls. De plus, l'opération de décimation en sortie du filtre implique que seul un échantillon sur D est utilisé. Les filtres RIF polyphases contournent ces difficultés.

3.2.1.2.3 Filtres RIF polyphase

L'amélioration de la structure de la Figure 3.15 conduit à imaginer des solutions améliorant le rapport entre le nombre d'échantillons traités et le nombre d'échantillons utiles. L'objectif est d'utiliser la partie filtrage à la fréquence d'horloge la plus faible : le nombre d'opérations par seconde devient alors optimal. Les architectures de filtres RIF remplissant ce critère sont dites polyphases pour des raisons explicitées ultérieurement [Pro96].
Dans un premier temps, l'intégration de l'opération d'insertion dans le filtre RIF est envisagée. La topologie répondant à cet objectif est présentée par la Figure 3.16.

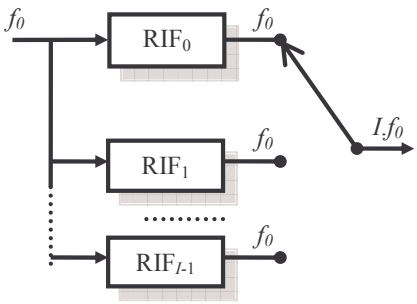


Figure 3.16. Filtre RIF polyphase interpolateur.

Ce filtre est composé de I branches de filtrage. Chaque branche représente un filtre RIF direct fonctionnant à f_0. Ces sous-filtres constitutifs correspondent à la description de la Figure 3.14. Un multiplexeur accède alternativement aux sorties de chaque branche à la fréquence $I \,.\, f_0$ pour constituer le signal sur-échantillonné.

L'architecture réciproque de la Figure 3.16 conduit à la construction de filtres RIF polyphases décimateurs. La topologie correspondante est présentée sur la Figure 3.17.

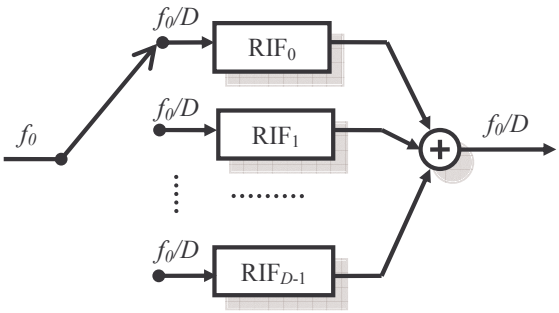


Figure 3.17. Filtre RIF polyphase décimateur.

La décimation est placée en tête du système afin de réduire la fréquence d'échantillonnage avant d'effectuer le filtrage en lui-même (risque de repliement). De la même façon que pour le filtre d'interpolation, le filtrage est décomposé en D branches de filtres RIF direct.

Les coefficients des filtres polyphases s'obtiennent en deux étapes. Tout d'abord, un filtre RIF direct ayant la réponse désirée est optimisé. Les coefficients $h(n)$ ainsi obtenus sont ensuite répartis sur les branches polyphases (repérées par k) selon les relations suivantes :

$$p_k(n) = h(nI - k) \qquad \text{avec } k \text{ entier} \in [0;\ I-1] \qquad \text{(filtre interpolateur)} \qquad (3.10)$$

$$p_k(n) = h(nD - k) \qquad \text{avec } k \text{ entier} \in [0;\ D-1] \qquad \text{(filtre décimateur)} \qquad (3.11)$$

Si le nombre de coefficients initial n'est pas un multiple entier du facteur de CFE (c'est-à-dire multiple de I ou de D), les coefficients polyphases $p_k(n)$ n'ayant pas de valeur attribuée par l'opération (3.10) ou (3.11) prennent la valeur 0.

L'origine de l'emploi du terme polyphase peut aider à comprendre le principe de ces filtres. Il provient des caractéristiques dont peuvent disposer les sous-filtres si le gabarit de départ est de type passe-bas. En effet, pour un filtre d'interpolation, les coefficients $p_k(n)$ des I sous-filtres sont obtenus par décomposition de $h(n)$. Par conséquent, si la réponse en fréquence de départ est plate sur l'intervalle $0 \le |\omega| \le \pi/I$ (filtre passe-bas), chaque branche possède alors une réponse relativement constante sur $0 \le |\omega| \le \pi$. Finalement, un filtre est composé de sous-filtres ne différant en première approximation que par leurs caractéristiques de déphasage d'où l'appellation filtre polyphase.

Les différents filtre proposés (CIC, RIF direct, RIF polyphase) ne peuvent être utilisés que pour des facteurs de CFE entiers. Il est possible d'obtenir des facteurs rationnels en cascadant un filtre d'interpolation avec un décimateur et réciproquement un filtre de décimation avec un interpolateur.
D'autres solutions doivent être étudiées afin d'assurer un changement de fréquence d'échantillonnage avec un rapport irrationnel.

3.2.1.2.4 Filtrage numérique à temps continu : les filtres de Farrow

Pour obtenir un facteur de CFE quelconque, il est nécessaire de connaître à chaque instant la valeur de μ_i représentant la position de l'échantillon à évaluer entre deux échantillons d'entrée connus (§3.1.1.2.3). A partir de μ_i, le système doit avoir à sa disposition la réponse impulsionnelle discrète permettant de calculer la sortie (Figure 3.18).

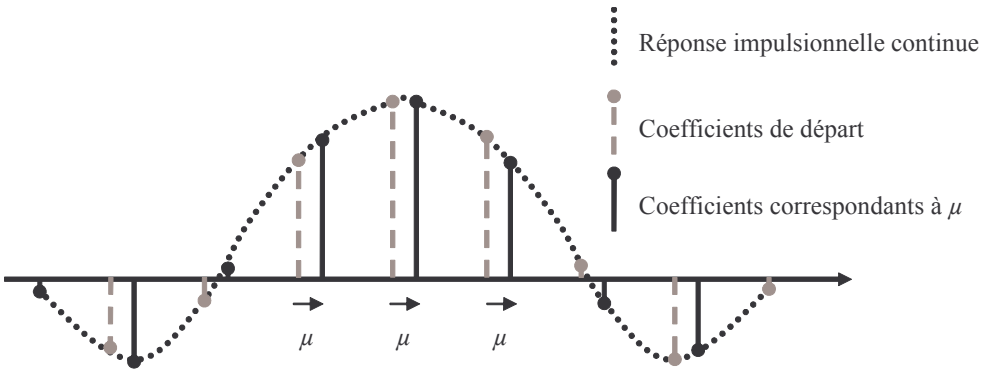


Figure 3.18. Principe des filtres numériques à réponse impulsionnelle variable.

Comme la réponse impulsionnelle du filtre dépend de la valeur courante de μ_i, il n'est pas intéressant de sélectionner et de mémoriser des jeux de valeurs pour $h(n)$. Cependant, la totalité de la réponse impulsionnelle continue $h(t)$ doit être connue ce qui est en contradiction avec le concept de temps discret inhérent au monde numérique.

Dans le cas où la CFE est de facteur entier ou rationnel, le système varie de façon périodique dans le temps. Seuls certains jeux de valeurs sont nécessaires aux calculs. Ces coefficients peuvent être mémorisés et employés dans certains systèmes dédiés à la CFE de facteur rationnel comme par exemple les filtres polyphases. Si L ou M deviennent grands, la taille de la mémoire devient critique. Ainsi, il est plus intéressant de calculer « à la demande » les valeurs de coefficients nécessaires pour $h(t)$ comme pour la CFE de facteur arbitraire.

Pour réduire l'effort nécessaire au calcul de ces coefficients, des fonctions simples pour décrire la réponse impulsionnelle à temps continu doivent être utilisées ; la décomposition polynomiale répond à ce besoin. Un jeu de coefficients devant être calculé pour chaque échantillon, une telle architecture n'est pas efficace en terme de volume de calcul et s'adapte donc difficilement aux traitements numériques rapides.

Dans le but d'optimiser les filtres polynomiaux, différentes architectures basées sur le filtre de Farrow [Far88] ont été développées. La plus simple est illustrée sur la Figure 3.19. Ces structures ne recalculent pas la réponse impulsionnelle pour chaque échantillon : seul le calcul en temps réel de μ_i (réalisable à l'aide d'un accumulateur) est nécessaire.

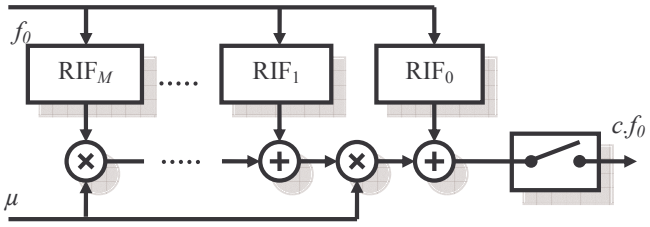


Figure 3.19. Filtre de Farrow direct d'ordre M.

Les M branches sont constituées de filtres RIF directs. A chacun de ces filtres correspond un ordre polynomial. La variable μ est appliquée à tous les multiplieurs cascadés à la sortie du système : elle est donc multipliée par elle-même M fois ce qui fournit un facteur $\mu_i{}^M$.
Soient m le rang de la branche du filtre de Farrow et n la position d'un coefficient dans la branche considérée. A l'aide des $m.n$ coefficients $c_m(n)$, il est possible de reconstituer la réponse impulsionnelle discrète instantanée du filtre de Farrow direct d'ordre M (3.12).

$$h\left(\left(k+\mu\right).T_{in}\right)=\sum_{m=0}^{M}c_m\left(k\right)\mu^m \quad \text{avec} \quad k\in\left[-N/2;\left(N/2\right)-1\right] \tag{3.12}$$

L'équation (3.12) correspond au fonctionnement présenté sur la Figure 3.18. Si le filtre est causal, les coefficients ont les propriétés de symétrie décrites par (3.13). Ces propriétés peuvent être facilement vérifiées sur la Figure 3.21.

$$c_m(N-1-n)=\begin{cases} c_m(n) & pour\ m\ pair \\ -c_m(n) & pour\ m\ impair \end{cases} \tag{3.13}$$

La réponse impulsionnelle $h(t)$ étant symétrique, le système global présente une phase linéaire. La fonction de transfert H_a (3.14) est calculée à partir des coefficients $c_m(n)$ et des fonctions polynomiales de base (cf. (3.15) et (3.16)).

$$H_a(j2\pi f)=\sum_{n=0}^{N/2-1}\sum_{m=0}^{M} c_m(n)G_m(n,T_n,t) \tag{3.14}$$

Les termes $G_m(n, T_n, t)$ sont les transformées de Fourier des fonctions de bases telles que :

$$g_m(n,T_n,t)=(-1)^m f_m\left(n,T_n,t-\sum_{n=0}^{N/2-1}T_n\right)+f_m\left(N-1-n,T_n,t-\sum_{n=0}^{N/2-1}T_n\right) \tag{3.15}$$

$$f_m(n,T,t)=\begin{cases} \left(\dfrac{2\left(t-\sum_{i=0}^{n-1}T_i\right)}{T_n}-1\right)^m & pour \sum_{i=0}^{n-1}T_i \le t < \sum_{i=0}^{n}T_i \\ O \quad ailleurs \end{cases} \tag{3.16}$$

Dans ces équations, la longueur T_n des sections polynomiales peut être définie comme étant $T_n = J_n.T$, avec J_n un entier supérieur ou égal à un. Dans le cas le plus général, la longueur de chaque section polynomiale peut être différente. Le cas le plus courant est celui pour lequel les longueurs des sections polynomiales sont égales, c'est-à-dire $T_n = T$ pour $n = 0, 1,\ldots, N\text{-}1$ [Bab02].
En fonction de T, trois cas peuvent se présenter :

- T est égale à la période d'échantillonnage d'entrée ou de sortie du filtre ;
- T est un entier multiple de la période d'échantillonnage d'entrée ou de sortie ;
- T est une fraction de la période d'échantillonnage d'entrée ou de sortie.

La structure de Farrow modifiée, correspondant au cas où T est égal à la période d'échantillonnage d'entrée, est retenue. Par rapport à la structure directe, cette solution permet d'économiser la moitié des coefficients en effectuant le changement de variable $\mu \rightarrow 2.\mu\text{-}1$. Elle est constituée de M sous filtres de longueur N, où N est le nombre de segments polynomiaux et M l'ordre du polynôme. Le nombre total de coefficients est donc $(M+1)\ .\ N/2$ auxquels se rajoutent M multiplieurs pour μ. Les fonctions polynomiales de

base exprimées par (3.16) sont représentées sur la Figure 3.20 pour les filtres de Farrow direct et modifié.

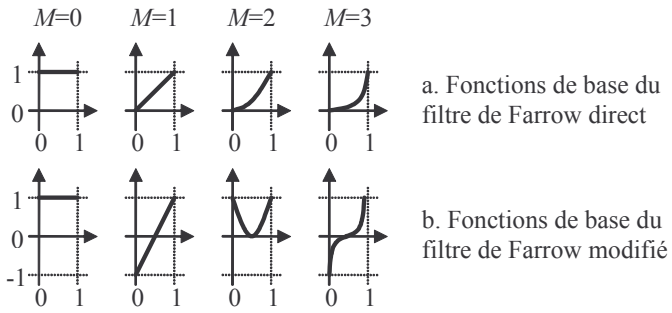


Figure 3.20. Fonctions polynomiales de base.

L'exemple de la décomposition d'une réponse impulsionnelle de filtre polynomial présenté sur la Figure 3.21 (source des coefficients : [Far88]) utilise les fonctions illustrées sur la Figure 3.20.a et permet d'expliciter la relation (3.12).

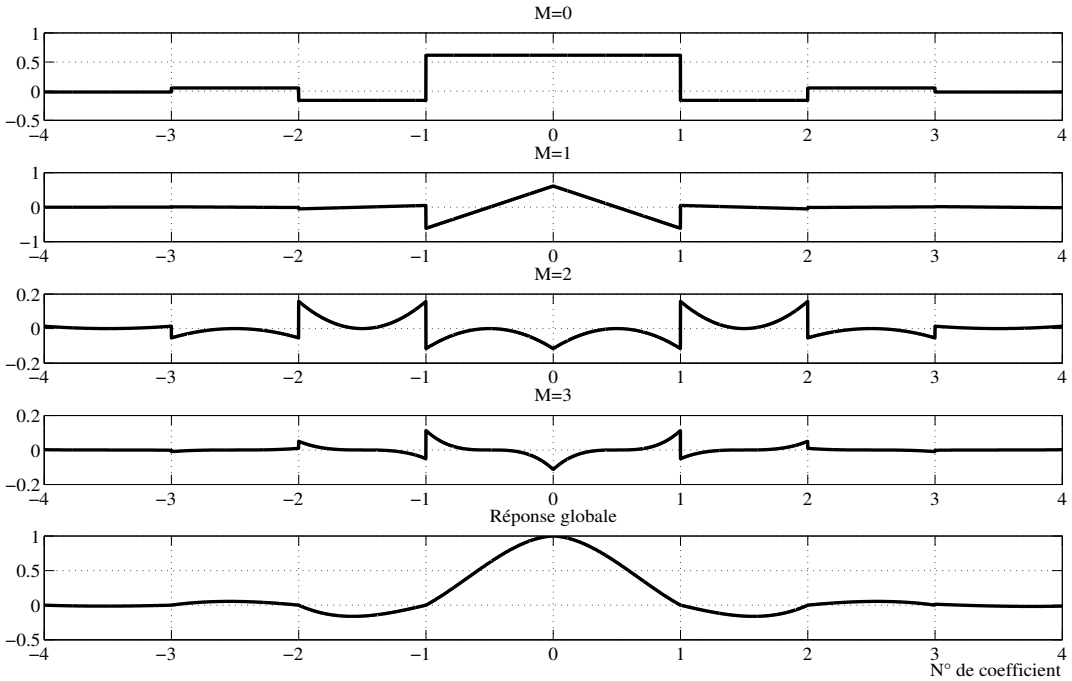


Figure 3.21. Construction d'une réponse impulsionnelle de filtre de Farrow pour N=8 et M=3.

Les filtres de Farrow permettent en théorie de prendre en charge toute forme de gabarit de filtrage et d'obtenir un facteur de CFE quelconque. Par conséquent, ce type d'architecture est très adapté aux besoins d'une TRN pour la radio logicielle restreinte. De plus, le calcul de μ suffit à effectuer l'interpolation pour chaque échantillon d'entrée. L'implémentation matérielle de ces filtres est donc facilitée et permet une utilisation à des fréquences supérieures à 10 MSPS.

Ces filtres présentent malgré tout un encombrement relatif assez important ce qui impose d'étudier les deux axes suivants : l'association du filtre de Farrow à d'autres filtres

numériques (§3.2.1.3) et l'optimisation du nombre de coefficients polynomiaux nécessaires (§3.3.1.2.4).

3.2.1.2.5 Bilan

Différentes techniques de filtrage numérique ont été présentées. La notion de compromis entre les degrés de liberté et l'utilisation de ressources matérielles apparaît comme fondamentale. Les filtres CIC n'utilisant aucun comparateur ne permettent pas de conformer leur fonction de transfert et les filtres de Farrow, répondant à toutes les contraintes de filtrage n'offrent pas de capacités d'intégration aussi intéressantes. Les filtres polyphases peuvent être considérés comme une solution intermédiaire dans ce compromis flexibilité/intégration. Le Tableau 3.1 présenté en début de ce paragraphe résume l'ensemble de ces remarques et peut être consulté *a posteriori*.

L'objectif de l'exploration architecturale exposée est de respecter les contraintes initiales en optimisant l'encombrement de la solution. Si un filtre de Farrow permet de répondre à toutes les contraintes de filtrage, ne retenir que cette architecture peut conduire à des solutions non optimales et trop encombrantes pour être implantées. Le sujet de la mise en cascade de plusieurs blocs de filtrage est traité dans le paragraphe suivant.

3.2.1.3 Mise en cascade

3.2.1.3.1 Etude préliminaire

Le but de cette étude est de réduire l'encombrement des solutions proposées précédemment grâce à l'association de plusieurs filtres.
La réponse fréquentielle globale de la mise en cascade de *m* filtres est la suivante :

$$H_{global\ dB} = \sum_{i=1}^{m} H_{i\ dB} \tag{3.17}$$

Deux stratégies d'optimisation sont alors possibles :

- soit l'effort de filtrage est réparti entre les filtres (Figure 3.22)
- soit une partie des filtres assure la fonction de filtrage et l'autre présente une réponse en fréquence constante dans la bande (0 dB) et assure principalement un rôle relatif au changement de fréquence d'échantillonnage [Nav04].

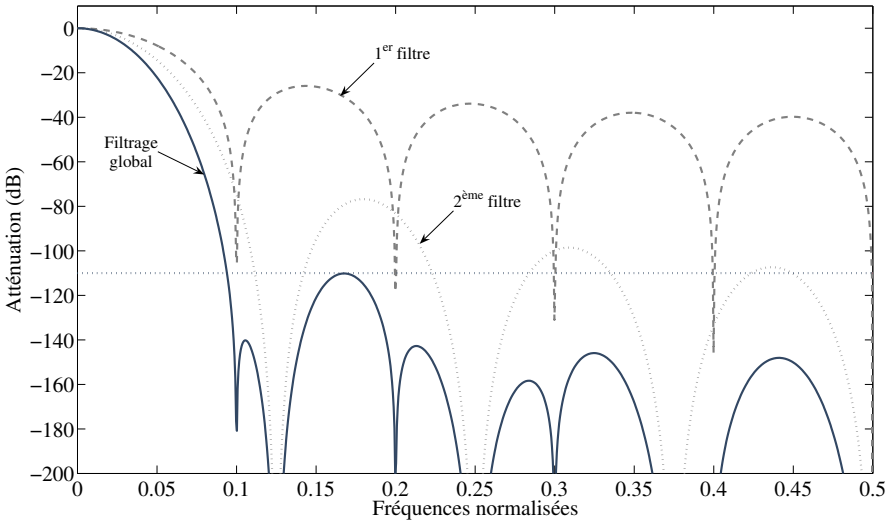


Figure 3.22. Mise en cascade de deux filtres : répartition de l'effort de filtrage.

Le facteur de conversion de fréquence d'échantillonnage issu de l'enchaînement de m filtres multifréquences est le produit des facteurs de CFE de chaque filtre :

$$CFE_{global} = \prod_{i=1}^{m} CFE_i \tag{3.18}$$

La prise en compte du facteur de CFE rajoute une contrainte à l'optimisation de la réponse en fréquence. Ces deux problèmes ne peuvent être traités séparément : la fonction de transfert des filtres CIC dépend directement du facteur d'interpolation ou de décimation. De plus, un facteur de CFE R ne peut être attribué qu'à un filtre polyphase dont le nombre de coefficients est au moins égal à R. Cette contrainte pose problème lorsque R est élevé et que les contraintes de filtrages sont faibles : le nombre de coefficients nécessaires est inférieur au nombre de branches polyphases.

L'espace de conception croit considérablement en ajoutant un degré de liberté sur le nombre de filtres utilisés. L'hétérogénéité des filtres utilisés augmente également la difficulté d'exploration des solutions : le nombre de combinaisons possibles exige un effort méthodologique afin d'isoler les associations les plus pertinentes et de restreindre l'optimisation du filtrage à quelques cas.

La mise en cascade d'un nombre de filtres supérieur à deux permet dans certains cas de diminuer le nombre d'opérations à effectuer par seconde (Figure 3.23). Par exemple, l'utilisation d'une chaîne constituée de filtres CIC [Hen00a] ou de filtres RIF polyphases décimateurs permet de réduire de proche en proche la fréquence d'horloge pilotant chaque étage de la chaîne.

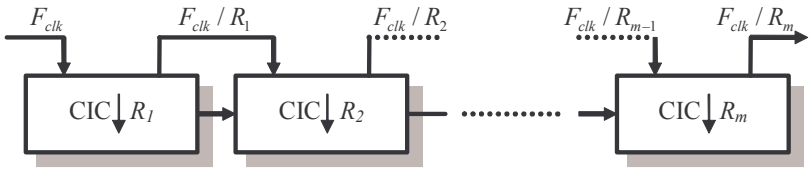


Figure 3.23. Mise en cascade de filtres CIC.

Ces cas sont très restrictifs au regard du facteur de CFE R à prendre en charge. En effet, R doit être le résultat d'un produit de m entiers et doit respecter la relation (3.19).

$$R = \prod_{i=1}^{m} R_i \text{ avec } R_i \in \aleph \ (\Rightarrow R \in \aleph) \tag{3.19}$$

La mise en œuvre de tels systèmes impose des contraintes de conception peu compatibles avec l'aspect reconfigurabilité de la RLR. L'étude proposée se limite donc à la mise en cascade de deux filtres.

L'optimisation se base directement sur le nombre de coefficients mis en œuvre. Pour des systèmes multifréquences, c'est-à-dire comprenant des blocs constitutifs travaillant avec des horloges de fréquences différentes, la puissance de calcul, qui est une métrique permettant de comparer différentes solutions entre elles, peut être utilisée [Nav04]. Pour un seul filtre, elle s'exprime de la manière suivante :

$$P_c = N \frac{f_s}{M} \tag{3.20}$$

Pour la mise en cascade de K filtres, cette relation devient :

$$P_c = N_1 \frac{f_s}{M_1} + N_2 \frac{f_s}{M_1.M_2} + \cdots = f_s \sum_{i=1}^{K} \frac{N_i}{\prod_{j=0}^{i-1} M_j} \text{ avec } M_0 = 1 \tag{3.21}$$

Cette technique d'estimation de P_c est implicitement utilisée dans l'analyse des solutions de mise en cascade de deux filtres. L'heuristique développée dans ces travaux permet donc *a priori* d'aboutir à une puissance de calcul optimisée, en supposant que le nombre de coefficients utilisés pour chaque filtre le soit aussi. Dans ce cadre, un simple dénombrement des coefficients évite l'évaluation systématique de la puissance de calcul.

3.2.1.3.2 Mise en cascade de deux filtres

L'espace de conception concernant la mise en cascade de deux filtres est défini par deux variables : le choix des filtres (architecture, gabarit, CFE) et l'ordre dans lequel ils sont placés. Selon le facteur R à assurer, les solutions sur lesquelles mener une étude comparative sont très différentes. Elles sont donc classées en fonction de la nature de R : unité, entier (supérieur à un), rationnel et irrationnel. Notons que dans la liste des solutions

proposées, celles qui contiennent un filtre CIC ne sont adaptées qu'à des cas de gabarit passe-bas.

CFE unité

La solution à retenir pour $R = 1$ est le RIF direct. La mise en cascade d'un filtre d'interpolation de facteur R avec un filtre de décimation compensant le premier avec un facteur R également est théoriquement envisageable. L'utilisation de deux filtres CIC peut conduire à une architecture plus compacte qu'un RIF direct, mais elle pose un certain nombre de problèmes. Si le filtre CIC interpolateur précède le filtre CIC décimateur, le nombre d'opérations par seconde peut être élevé. Dans le cas contraire (filtre décimateur puis filtre interpolateur), il faut veiller à ce que le signal ne soit pas perturbé par le repliement si le facteur de décimation est trop élevé. Finalement, le manque de liberté sur l'optimisation de la réponse globale conduit à écarter cette possibilité.

CFE de facteur entier

Pour R entier (non unitaire), trois solutions sont disponibles : les filtres CIC, les filtres polyphases et les filtres de Farrow.

L'utilisation de filtres CIC, étant *a priori* la moins coûteuse en ressources, doit être étudiée en premier lieu. Elle correspond malheureusement rarement aux contraintes du concepteur du fait de la faible conformation de sa fonction de transfert et de la dépendance de celle-ci vis-à-vis de R. Si R est le produit de deux entiers, l'association de deux CIC apporte un degré de liberté supplémentaire sur la fonction de transfert (cf. (3.19)).

Les filtres RIF polyphases, synthétisés à partir d'un filtre RIF direct, consistent à répartir les N coefficients du filtre de départ sur R branches. Au même titre que les filtres RIF directs, ils peuvent répondre à toutes les formes de gabarit de filtrage.

L'utilisation de filtres de Farrow n'est pas optimale par rapport à la précédente puisque sa fonction de transfert est une décomposition polynomiale des N coefficients du filtre direct de référence. Cette décomposition d'ordre M aboutit alors à la synthèse d'un filtre contenant $M.N$ coefficients et n'apporte pas de gain de performance par rapport aux filtres polyphases. De plus l'utilisation d'un facteur entier implique que la paramètre μ soit constant voire nul. Dans ce dernier cas de figure, toutes les branches sont inopérantes sauf celle correspondant aux coefficients d'ordre $M = 0$. Il en résulte donc un simple filtre RIF direct suivi d'une opération de décimation (cf. Figure 3.19).

CFE de facteur rationnel

La conversion de fréquence d'échantillonnage de facteur rationnel conduit soit à l'association d'un filtre multifréquence et d'une opération d'insertion ou de décimation, soit à l'association de deux filtres multifréquences.

La mise en cascade d'une opération d'insertion ou de décimation et d'un filtre CIC donne lieu aux quatre solutions suivantes (Figure 3.24) :

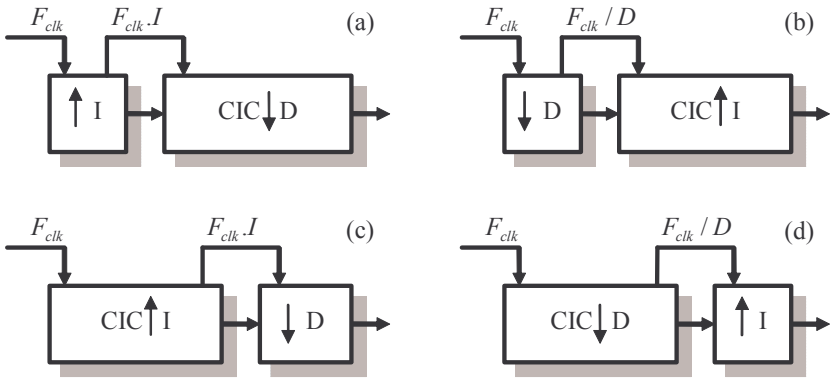


Figure 3.24. Mises en cascade de filtres CIC et de décimateurs/interpolateurs.

Les trois premières solutions peuvent être envisagées.
Dans le cas où une opération de décimation est utilisée en premier (b), il faut s'assurer d'éviter le repliement afin de préserver l'intégrité du signal traité.
La solution Farrow nécessite moins d'opérations par seconde que la solution (a) puisque la fréquence d'utilisation du filtre est f_{clk} et non $f_{clk}.I$ comme dans le premier cas.
La solution (d) est à écarter car l'insertion doit être suivie par un filtre pour interpoler le signal contenant essentiellement des échantillons nuls.

Si aucune de ces structures ne convient, l'utilisation de deux filtres CIC offre un plus grand degré de liberté sur la fonction de transfert (Figure 3.25). Seules deux combinaisons sont alors possibles.

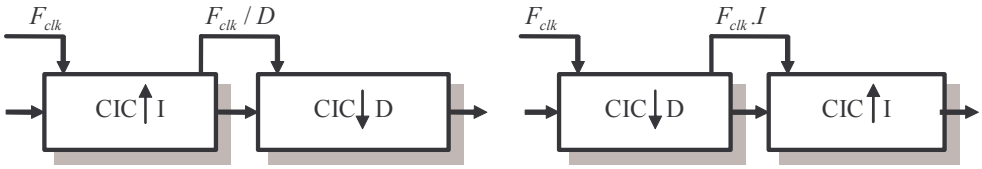


Figure 3.25. Mise en cascade de filtres CIC interpolateurs et décimateurs.

Si les deux solutions sont valides (CIC + interpolateur ou décimateur et deux CIC cascadés), la deuxième risque de nécessiter plus de ressources. Une comparaison des ordres des filtres dans les deux cas doit donc être menée.

La combinaison d'un filtre RIF polyphase et d'une fonction d'insertion ou de décimation reprend les configurations illustrées sur la Figure 3.24 ainsi que les analyses correspondantes. Cette architecture s'applique à tous les cas pour lesquels la CFE est de facteur rationnel puisque la fonction de transfert est quelconque, faisant disparaître la restriction des CIC pour les gabarits passe-bas.
L'association de deux filtres polyphases interpolateur et décimateur (adaptation de la Figure 3.25 aux filtres polyphases) paraît non optimale puisqu'un RIF polyphase seul associé à un décimateur ou un interpolateur peut remplir toutes les contraintes.

De la même manière que dans le cas d'une CFE de facteur entier, l'utilisation d'un filtre de Farrow n'est pas optimale par rapport aux RIF polyphases.

CFE de facteur irrationnel

Dans le cadre défini dans le §3.2.1.2, la réalisation d'un facteur de CFE irrationnel impose *de facto* l'utilisation d'un filtre continu en temps tel que l'architecture de Farrow. Seul, il peut répondre à toutes les spécifications, mais ses exigences en termes de ressources matérielles constituent une préoccupation majeure. Il faut donc chercher à réduire les contraintes de filtrage sur ce système en l'associant à d'autres filtres.

Deux approches sont envisageables : soit le filtre complémentaire prend en charge les contraintes sur le filtrage, soit il permet de réduire au minimum la fréquence d'utilisation du filtre de Farrow.

Les multiples combinaisons possibles sont discutées en se limitant au cas où le système de filtrage doit opérer une décimation de facteur quelconque, ce qui correspond au fonctionnement du récepteur pour lequel $f_s > f_{bb}$ (Figure 3.1).

L'association filtre de Farrow – filtre RIF direct repose sur l'idée de prendre en charge le filtrage avec le filtre RIF direct, peu encombrant et la conversion de fréquence d'échantillonnage avec le filtre Farrow. Le filtre RIF direct est optimisé sans prendre en compte le facteur de CFE à assurer. Le filtre de Farrow obtenu est d'ordre faible puisque sa fonction de transfert doit être quasiment plate. Dans le cas d'une décimation, il paraît judicieux de placer le RIF en deuxième position afin de réduire sa fréquence d'utilisation et donc le nombre de ses coefficients. La méthode inverse consistant à porter l'effort de filtrage sur le filtre Farrow n'apporte aucun avantage par rapport à l'utilisation d'un filtre Farrow seul. Cette solution est donc écartée.

L'association d'un filtre de Farrow et d'un filtre polyphase peut être pertinente dans les deux ordres d'enchaînement. Le filtre placé en premier doit réduire la fréquence au minimum pour diminuer les contraintes sur le deuxième. Le fait qu'il soit possible d'obtenir un facteur de CFE quelconque avec un filtre de Farrow offre de larges possibilités d'optimisation.

La même analyse peut être faite en remplaçant le filtre polyphase par un filtre CIC [Bab01a]. Afin de contourner la difficulté concernant la conformation de la fonction de transfert de ces filtres, une structure CIC décimateur peut être placée avant le filtre de Farrow. Il s'agit alors de réduire la fréquence d'horloge du filtre Farrow au minimum en respectant les contraintes indiquées par la Figure 3.13. Le $2^{ème}$ filtre, travaillant à une fréquence réduite de f_0 / D_{CIC} peut alors ajuster le facteur de décimation et la réponse en fréquence.

3.2.1.4 Heuristique d'exploration de l'espace de conception

L'exploration de l'espace de conception concernant les fonctions de filtrage se divise en deux étapes.

Tout d'abord, une réflexion a permis d'écarter certaines solutions architecturales. La sélection des architectures à comparer suit la méthode décrite précédemment et correspond à l'heuristique de la Figure 3.26.
Par la suite, une comparaison exhaustive des solutions restantes permet de classer les différentes possibilités et de sélectionner l'optimale en fonction de son encombrement relatif.

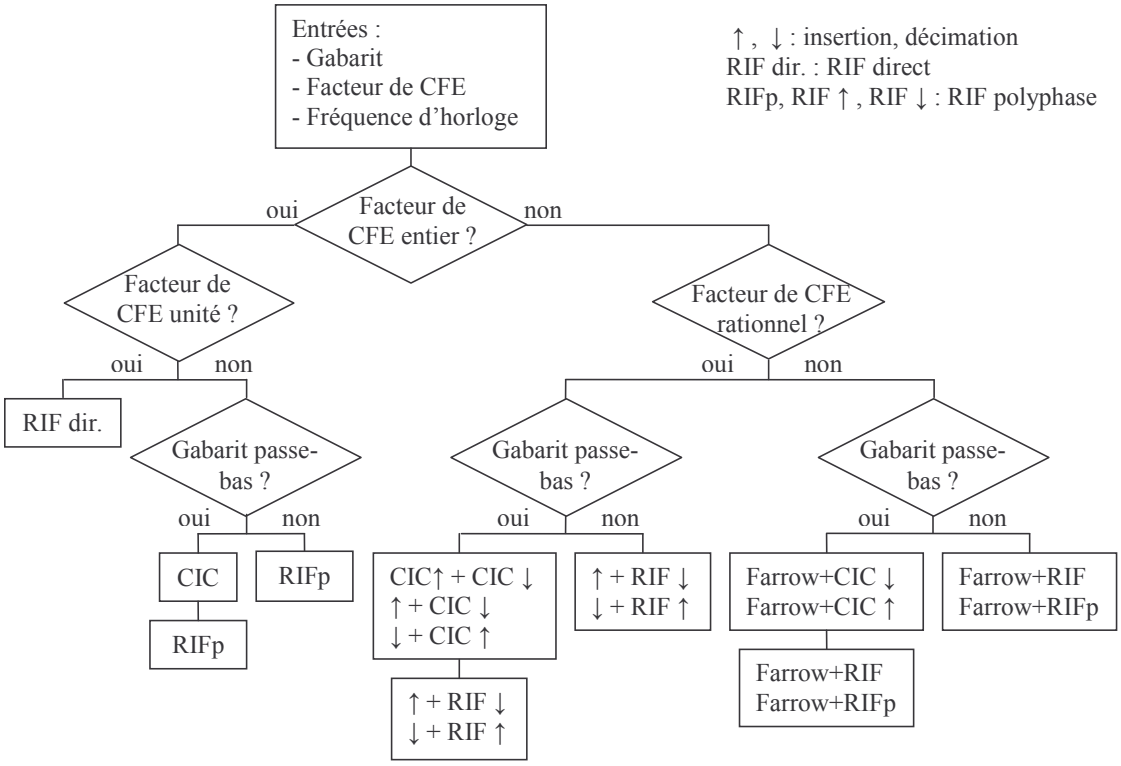


Figure 3.26. Heuristique de choix d'architectures de filtrage.

Un travail en amont sur l'architecture des filtres numériques pris en compte dans cette heuristique a pour but d'identifier les points communs entre ces structures. L'objectif est de faciliter et d'optimiser l'exploration architecturale sur la base d'un découpage modulaire des systèmes considérés.

3.2.2 Définition d'un module de base

3.2.2.1 Objectifs et enjeux

L'étude des architectures de filtres proposées dans le §3.2.1.2 (filtres utilisés), la comparaison des structures RIF direct (Figure 3.14), polyphase (Figure 3.16 et Figure 3.17) et Farrow (Figure 3.19) permet de mettre en évidence la possibilité d'utiliser un bloc générique pour les différentes architectures : la branche de filtre RIF. L'objectif est donc de développer une entité commune à plusieurs architectures de filtres ayant différentes caractéristiques et ce dans une perspective de réutilisabilité. La contrainte sur les ressources

doit guider l'optimisation de ce module tout en conservant à celui-ci un caractère générique et une fréquence d'utilisation compatible avec les applications RLR visées.
Cette démarche est parallèle à celle qui a été développée et appliquée à l'exploration architecturale modulaire des CAN pipeline exposée dans le chapitre 2. Elle se rapproche de travaux concernant la même thématique comme [Pal03] où l'opérateur FFT est utilisé comme bloc de base.

3.2.2.2 Granularité

Il convient de fixer un niveau de spécification pour développer et optimiser le bloc de filtrage élémentaire. Trois niveaux sont à distinguer : fonction, opérateur et module.

Le niveau **fonction** est le niveau le plus haut. Il consiste à optimiser la chaîne de filtrage dans sa globalité (Figure 3.27). La structure interne du système est décrite de façon comportementale.

Figure 3.27. Granularité : niveau fonction.

Ce niveau propose une description simple et peu fragmentée des fonctions de filtrage. Les degrés de liberté d'optimisation sont larges et la synthèse architecturale peut conduire à des solutions variées, très dépendantes des outils utilisés.

Le niveau **opérateur** est le niveau de granularité le plus fin (Figure 3.28). Il revient à fractionner au maximum la fonction en tâches élémentaires (addition, multiplication, retard…).

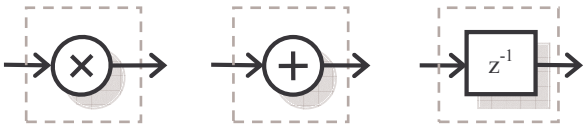


Figure 3.28. Granularité : niveau opérateur.

L'exploration architecturale à ce niveau offre des perspectives d'optimisation élevées mais ne permet pas de définir d'IP générique à proprement parler.

Le niveau **module** est un niveau intermédiaire pour lequel la fonction est découpée en entités génériques (Figure 3.28). Chaque bloc est un ensemble de fonctions élémentaires réutilisable.

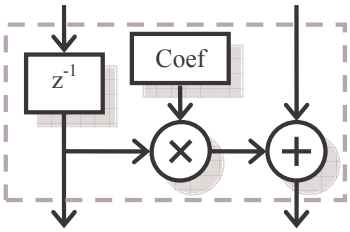


Figure 3.29. Granularité : niveau module.

Le niveau module est retenu, la synthèse matérielle pouvant être maîtrisée à l'aide des outils logiciels disponibles. Il présente un bon compromis entre une description haut niveau offrant un grand degré de liberté et une exploration de bas niveau permettant une optimisation plus fine.

3.2.2.3 Mise au point du module de base

Le fonctionnement du module doit être détaillé afin d'aboutir à une description comportementale optimale. L'étude au niveau opérateur du fonctionnement des filtres RIF direct et MAC (Multiply Accumulator), permettent de converger vers la mise au point du module de base.

3.2.2.3.1 Architecture d'un RIF direct

Un filtre RIF direct doit pouvoir se construire en cascadant plusieurs modules de base.
L'implantation de cette fonction peut être effectuée selon la structure classique (Figure 3.14). Elle peut également être construite selon le schéma bloc transposé (Figure 3.30), pour parvenir au même résultat.

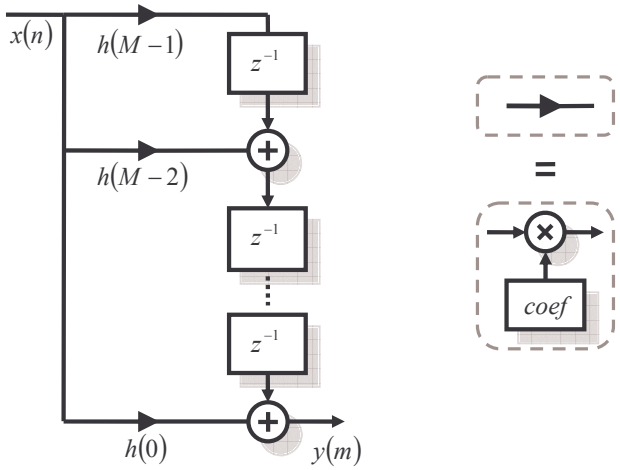


Figure 3.30. Schéma bloc transposé d'un filtre RIF direct.

Du point de vue matériel, cette topologie est plus simple à mettre en œuvre : les retards z^{-1} peuvent être inclus dans les opérateurs d'addition. En effet, l'obtention des résultats d'addition avec une latence d'un coup d'horloge rend inutile l'implémentation de points mémoires. La structure transposée facilite donc la synchronisation des différentes opérations de la chaîne.

Cette structure, entièrement parallèle, présente des performances fréquentielles optimales, mais nécessite un grand nombre de ressources matérielles. Pour palier à cet inconvénient, une sérialisation de l'architecture peut être envisagée.

3.2.2.3.2 Fonctionnement d'un RIF MAC

Les structures répétitives (Figure 3.30) impliquent naturellement l'idée de partage de ressources en rebouclant la fonction sur elle-même. Les filtres RIF peuvent utiliser la topologie MAC. Elle consiste à sérialiser totalement la structure RIF classique (Figure 3.31), l'objectif visé étant une économie de ressources matérielles.

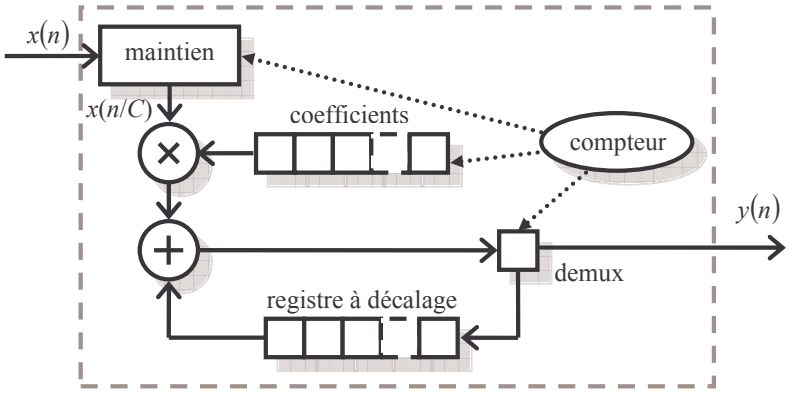


Figure 3.31. Schéma bloc d'une structure RIF MAC.

Toutes les opérations sont simultanées et reflètent directement un calcul de convolution tel qu'il peut être programmé ou effectué manuellement. L'entrée est maintenue tant qu'elle n'a pas été multipliée successivement par tous les coefficients. Le premier résultat de multiplication $x(n).h(0)$ est additionné avec la valeur présente dans la dernière case mémoire du registre. Ce résultat est stocké dans la première case de ce registre. Il en est de même successivement pour chaque coefficient. Lorsque tous les coefficients ont été utilisés, la sortie du module prend la valeur présente dans la dernière case du registre.
Le maintien de l'entrée et la récupération du mot de sortie sont synchrones à l'horloge de fréquence la plus basse (débit d'entrée et de sortie). Les opérations internes au filtre sont synchrones à l'horloge de fréquence la plus haute, égale à la fréquence d'entrée multipliée par le nombre de coefficients C.

Relativement à la structure série, cette technique ne permet pas d'atteindre des cadences de fonctionnement élevées. En admettant que la multiplication soit l'opération la plus longue à effectuer et qu'elle ne puisse assurer une cadence supérieure à f_{max}, le filtre RIF MAC ne pourra en théorie fonctionner à une fréquence utile supérieure à f_{max}/C, la multiplication étant utilisée C fois pour fournir chaque échantillon de sortie.

3.2.2.3.3 Fonctionnement du module de base

Le fonctionnement de l'architecture RIF série et celui de l'architecture série MAC ont été détaillés. Le module de filtrage développé est un compromis entre ces deux structures en

terme de rapidité et d'encombrement avec un principe de fonctionnement intermédiaire (Figure 3.32). Ce compromis est comparable à celui qui conduit au développement de CAN pipeline en tant que structure intermédiaire entre le CAN flash et le CAN à approximations successives (cf. chapitre 2). Ce module est construit sur la base de la structure MAC.

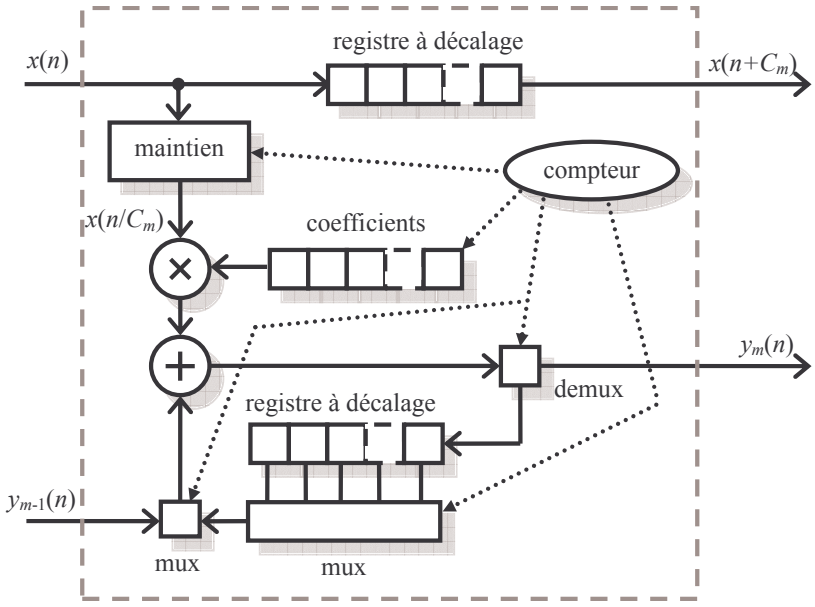


Figure 3.32. Schéma bloc d'un module de base de filtre RIF.

Par rapport au filtre MAC, ce module dispose de deux accès supplémentaires : une entrée (y_{m-1}) pour obtenir le résultat des opérations du module qui le précède et une sortie (y_m) fournissant son résultat au module qui lui succède. Il effectue un calcul utilisant C_m coefficients.

Son implémentation requiert quelques précautions. Il faut que le résultat précédent $y_{m-1}(n)$ soit appliqué en entrée du bloc en même temps que la valeur $y_m(n)$ correspondant à la sortie. Le compteur a pour rôle de cadencer chacune de ces opérations. Plus généralement, la construction du modèle fait l'objet d'une attention particulière concernant la synchronisation. Elle tient compte des latences de chaque opération ce qui implique de fortes contraintes de conception : les latences cumulées de l'additionneur et du multiplieur imposent un nombre minimum de coefficients par module.
Au niveau fonction, les modules doivent également être synchronisés entre eux. Pour ce faire, l'entrée $x(n)$ est retardée par un registre à décalage afin que les signaux y_m et x soient appliqués en même temps au bloc de rang m+1.

La mise en équation du système fait apparaître la possibilité de cascader ce module afin de constituer une branche de filtrage RIF. Les relations (3.22) et (3.23) correspondent respectivement aux réponses temporelles d'un filtre RIF direct et d'un module de filtrage. La relation (3.24) reprend ces deux résultats pour montrer la possibilité de décomposer un filtre RIF à l'aide de modules de filtrage.

$$y(n) = \sum_{i} x(i)h(n-i) = \sum_{i=0}^{N-1} h(i)x(n)z^{-i} \tag{3.22}$$

$$y_m(n) = \sum_{i=0}^{\frac{N}{B}-1} h(i)x(n)z^{-i} \tag{3.23}$$

$$y(n) = \sum_{m=0}^{B-1} \sum_{i=\frac{N}{B}(b-1)}^{\frac{N}{B}b} h(i)x(n)z^{-i} = \sum_{m=0}^{B-1} y_m(n) \tag{3.24}$$

Les termes $h(i)$ représentent les coefficients du filtre global et les valeurs $y_m(n)$ correspondent aux sorties des modules de filtrage, c'est-à-dire aux résultats intermédiaires dans la chaîne de filtrage.

Le nombre C_m de coefficients par module doit être un multiple entier B de l'ordre N du filtre à implémenter ce qui implique l'égalité (3.25).

$$N = B.C_m \tag{3.25}$$

C_m correspond au facteur de rebouclage : le module RIF est rebouclé C_m fois sur lui-même avant de répéter sa séquence de coefficients. Il est donc cadencé à l'aide d'une horloge dont la cadence atteint C_m fois le débit des données présentées en entrée. Il apparaît donc une notion de fréquence de fonctionnement (f_{clk}) et de fréquence utile (f_{utile}) qui lui est inférieure :

$$f_{utile} = f_{clk} / C_m \tag{3.26}$$

Il faut donc calculer le nombre de coefficients pris en charge par chaque module en fonction des besoins de débit imposés par l'application.

La structure obtenue en développant la boucle de chaque module de filtrage est détaillée sur la Figure 3.33. Ce schéma permet de préciser les notations et d'illustrer les équations (3.22), (3.23) et (3.24).

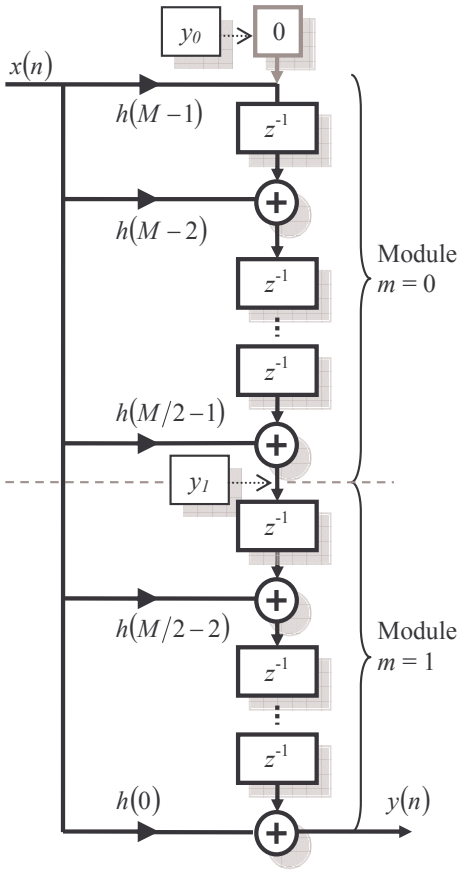


Figure 3.33. Mise en cascade de deux modules constituant une branche de filtre RIF direct.

Ce module est décrit de façon générique et constitue un élément de la bibliothèque proposée sur la Figure 3.9.

3.2.2.3.4 Développement VHDL

Le langage utilisé pour développer les IP relatives à ce travail est le VHDL. Le module de filtrage devant être réutilisé, il doit être le plus générique possible. A cet effet, les tailles des vecteurs de bits ainsi que le nombre de coefficients sont définis dans un *package*.

L'architecture globale est paramétrable pour permettre la modification de son fonctionnement en cours d'utilisation. Cela nécessite la mise en place d'accès spécifiques relatifs aux valeurs et aux adresses des nouveaux paramètres.

L'ensemble des accès du module est répertorié sur la Figure 3.34.

```
ENTITY bloc_RIF IS
GENERIC(     c_init : bloc_coe ) ;
PORT(
  CLK, EN, DIR, Wrcoef : IN std_logic ;
  Entree, bloc_prec : IN std_logic_vector(Wdata-1 DOWNTO 0) ;
  ADDRwr, varNcoef : IN std_logic_vector(Waddr-1 DOWNTO 0) ;
  COEFin : IN std_logic_vector(Wcoef-1 DOWNTO 0) ;
  DOR : OUT std_logic ;
  sortie : OUT std_logic_vector(Wdata-1 DOWNTO 0) ;
  entree_dly : OUT std_logic_vector(Wdata-1 DOWNTO 0)   ) ;
END ;
```

Figure 3.34. Code VHDL de l'entité du module de filtrage.

Les définitions des différents accès présentés dans cette entité sont présentées dans le Tableau 3.2.

Port	Direction	Usage
`CLK`	in	Horloge pilotant le module RIF
`EN`	in	Enable : le module fonctionne si EN = 1
`DIR`	in	Data In Ready : validation de la donnée `entree`
`DOR`	out	Data Out Ready : donnée `sortie` valide
`entree`	in	Bus de données (en entrée)
`bloc_prec`	in	Résultat de filtrage du bloc précédent
`sortie`	out	Bus de données (en sortie)
`Wrcoef`	in	Permission d'écrire un nouveau coefficient lorsque Wrcoef=1
`ADDRwr`	in	Adresse du coefficient à remplacer
`COEFin`	in	Valeur du coefficient à remplacer
`varNcoef`	in	Coefficient de rebouclage du bloc = nombre de coefficients utilisés par cycle de filtrage
`entree_dly`	out	L'entrée « `entree` » est retardée

Tableau 3.2. Entrées/sorties du blocs de filtrage.

Le mot *ADDRwr* permet d'indiquer l'adresse mémoire du coefficient à remplacer. La nouvelle valeur est portée par *COEFin* et validée par *Wrcoef*. Le mot *varNcoef* impose au compteur sa valeur de remise à zéro. Il a donc pour rôle de contrôler le nombre de coefficients de filtrage utilisés dans le module. Ce paramètre correspond à la valeur de C_m utilisée dans les relations (3.25) et (3.26). Les accès DIR et DOR sont utilisés pour la synchronisation entre blocs lorsque ceux-ci sont cascadés.

La présence du multiplieur et de l'additionneur implique un accroissement du nombre de bits. En effet, en considérant des opérateurs disposant de deux entrées sur W bits, la sortie doit être sur W+1 bits pour un additionneur et sur $2W$ bits pour un multiplieur. Ce phénomène devient problématique lorsque le nombre d'opérations successives est important, puisqu'il peut conduire à la manipulation de vecteurs de données beaucoup plus longs que le vecteur d'entrée. Cet inconvénient est contourné grâce à l'utilisation d'arrondis dans le module. L'impact de ces arrondis sur le bruit numérique n'est pas étudié [Con01, Men02]. Des outils spécifiques dédiés à cette problématique pourraient alors être utilisés (*Fixed-Point Designer* [Syn]).
Ce principe permet d'obtenir un vecteur d'entrée et de sortie de même taille, ce qui simplifie l'instanciation en chaîne de différentes entités (Figure 3.35).

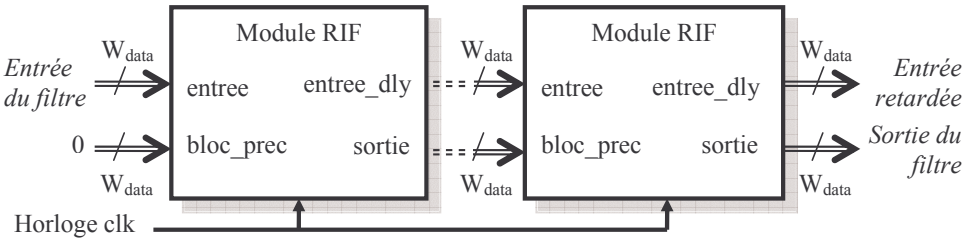


Figure 3.35. Mise en cascade de deux modules de filtrage.

L'entité a été développée de sorte à pouvoir utiliser aisément la fonction *generate* du langage VHDL, fonction permettant entre autres d'instancier en boucle des entités devant être cascadées ou disposées en matrice.

Associée à l'entité, l'architecture définit le fonctionnement du module de filtrage selon une description hybride. Elle est à la fois de nature comportementale et structurelle. L'additionneur et le multiplieur sont instanciés à partir de la bibliothèque d'IP fournie par le constructeur du FPGA utilisé [Xil], assurant ainsi l'optimisation fréquentielle et matérielle de ces opérateurs. De plus, cette méthode permet d'ajuster le fonctionnement des opérateurs aux besoins (entrées signées ou non, nombre de bits en sortie, latence) et de contrôler l'architecture finale (pipelinée ou non). Les autres fonctions du module (multiplexage, démultiplexage, mémoire, compteur) sont décrites à l'aide de codes comportementaux propriétaires. En effet, la paramétrisation dynamique spécifique à l'entité développée ne permet pas l'utilisation directe des IP disponibles.

3.2.2.3.5 Synthèse

Les résultats de synthèse relatifs au module développé sont fonctions de certains choix de développement tels que l'architecture et les caractéristiques du multiplieur et de l'additionneur utilisés.

L'additionneur, de même que le multiplieur, imposent un compromis ressources utilisées/fréquence maximale/latence.
D'un côté, si l'opérateur est optimisé en fréquence, sa structure est parallélisée au détriment de la quantité de ressources nécessaires. De l'autre, si l'opérateur est optimisé en ressources, sa structure est sérialisée au détriment de la fréquence maximale et de la latence. Or, le nombre de coefficients minimum est directement fonction de la latence des opérateurs. Le concepteur doit donc être attentif aux ressources utilisées et au nombre de coefficients dont il a besoin par module. Partant de ces observations, une étude paramétrique doit être menée pour quantifier l'influence de chaque choix.

Trois architectures (au sens VHDL) ont été développées pour répondre au mieux aux différents besoins.

La première offre la possibilité de changer le nombre de coefficients actifs et donc le rebouclage en cours de fonctionnement. Cette fonctionnalité implique de gérer

dynamiquement la taille des résultats d'addition intermédiaires et la synchronisation des différentes opérations.

La deuxième utilise un rebouclage fixe, c'est-à-dire un nombre constant de coefficients. Cette architecture est donc plus simple et utilise moins de ressources que la première.

La troisième est un module ne contenant qu'un seul coefficient. Elle donne un point de comparaison avec une architecture de filtre RIF classique et n'utilise pas les principes développés pour le module de filtrage de base.

Une étude paramétrique est menée sur le premier module (architecture entièrement reconfigurable). Ce module est étudié selon deux configurations : une première optimisée en fréquence (Figure 3.36) et une autre optimisée en ressources (Figure 3.37).
Pour toutes les configurations, la résolution de l'entrée est de 12 bits.
Pour la première configuration, l'additionneur et le multiplieur ont une latence de deux périodes d'horloge, ce qui permet un nombre minimum de coefficients de quatre. L'étude de cas est effectuée en fonction de la résolution des coefficients, du nombre de coefficients (4 ou 20) et du type de ressources utilisées dans le FPGA pour l'opération de multiplication : logique dédiée (multiplieurs câblés 18x18 bits) ou ressources distribuées (unités logiques programmables) (cf. Figure 3.10). Les résultats indiqués concernent la fréquence maximale permise pour l'utilisation du bloc, et la quantité de ressources logiques nécessaires (l'unité est le *slice*, c'est-à-dire l'unité logique de base). Ces performances sont évaluées après synthèse (outils Leonardo [Men]) et *mapping* (outils ISE [Xil]) sur un composant cible Xilinx Virtex II-4 d'un million de portes.

Les résultats correspondant au graphique de la Figure 3.36 ainsi qu'aux Figure 3.37 et Figure 3.38 sont détaillés dans des tableaux de valeur en Annexe C.

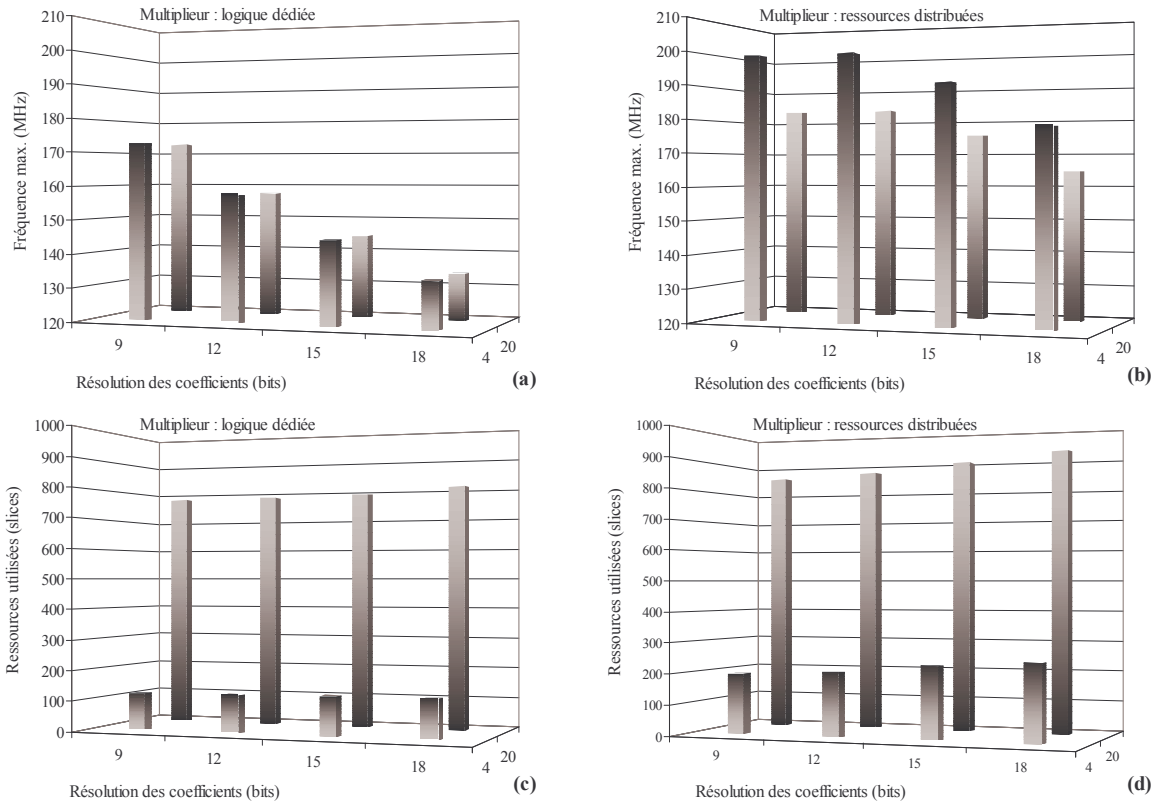


Figure 3.36. Etude paramétrique sur le module de filtrage reconfigurable optimisé en fréquence.

La Figure 3.36 démontre l'importance du dimensionnement des coefficients du filtre pour optimiser la fréquence d'utilisation du module. L'utilisation de la logique distribuée pour le multiplieur permet d'atteindre des fréquences plus élevées qu'avec les opérateurs dédiés (a. et b.), mais le premier cas implique l'utilisation d'environ 50 à 100 *slices* supplémentaires par rapport au deuxième (c. et d.). Le multiplieur occupe donc environ 50 à 100 *slices* selon la résolution des coefficients lorsqu'il n'est pas implémenté par un bloc arithmétique dédié.
Le graphique c. indique que l'utilisation de 20 coefficients fait plus que quintupler les ressources nécessaires par rapport à un module n'en contenant que 4. Ce phénomène s'explique par le besoin d'utiliser un additionneur fournissant un résultat sur un plus grand nombre de bits, le nombre d'accumulations étant supérieur dans ce cas. Chaque addition rajoutant un bit au résultat, l'additionneur occupe en proportion un nombre de *slices* plus élevé.
L'élément critique limitant la fréquence maximale d'utilisation est le multiplieur.

Ces résultats sont extraits dans les mêmes conditions pour un module optimisé en terme de ressources avec une latence identique d'une période d'horloge pour le multiplieur et l'additionneur (Figure 3.37). Le nombre minimum de coefficients permis est de 3, l'étude est donc effectuée pour un module comprenant soit 3 coefficients, soit 20 comme dans le cas précédent.

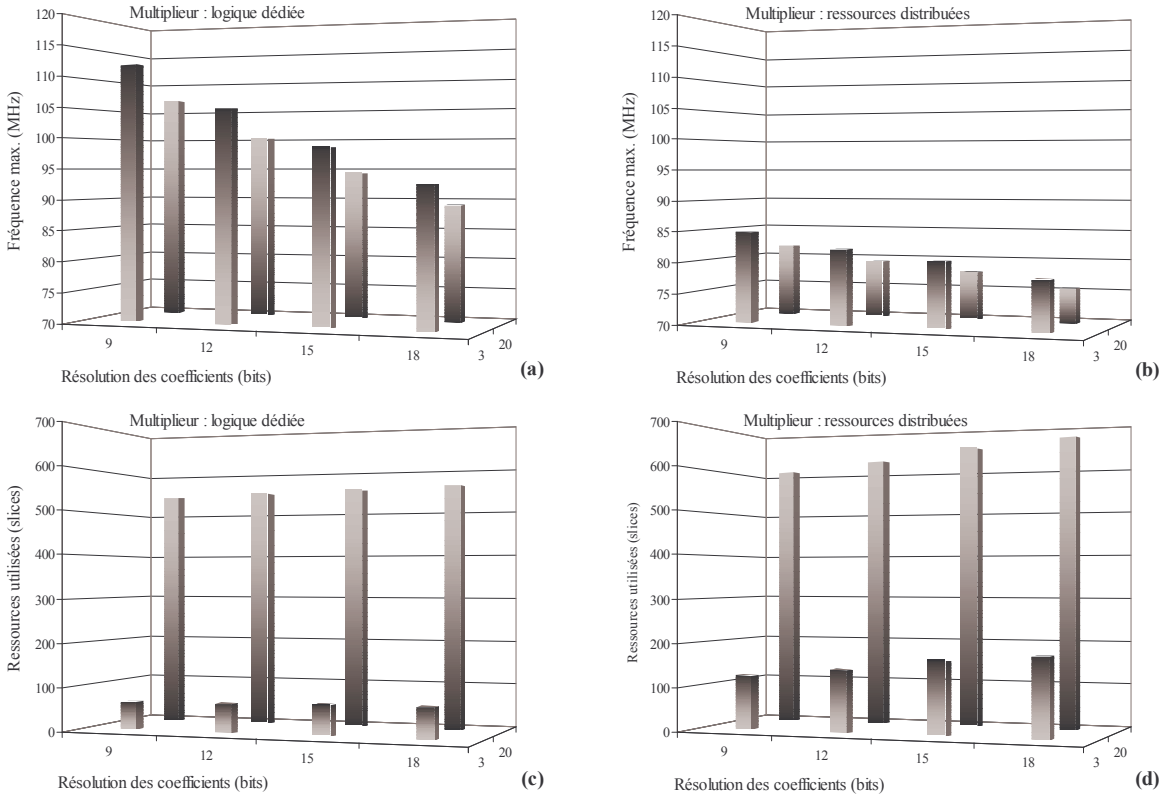


Figure 3.37. Etude paramétrique sur le module de filtrage reconfigurable optimisé en ressources.

Dans ce deuxième cas, la fréquence de fonctionnement permise est plus élevée lorsque les blocs de multiplication dédiés sont utilisés (a. et b.). Cette fréquence est en moyenne de 50 MHz à 100 MHz plus basse lorsque le module est optimisé en ressources comparativement aux résultats de la Figure 3.36 (a. et b.). Cependant, cette version économise en moyenne 30 à 35% de ressources par rapport à la précédente. Ce gain peut être non négligeable lorsque le composant cible est limité en nombre de *slices*.

Afin de donner une référence aux deux études précédentes, des résultats sont obtenus pour des blocs de filtrage n'offrant pas la possibilité de modifier le nombre de coefficients utilisés en cours de fonctionnement (Figure 3.38). La valeur des coefficients quant à elle reste programmable. Cette troisième série de résultats vise à évaluer l'impact des possibilités de reconfiguration sur les performances. Les conditions d'obtention des résultats sont identiques aux précédentes. Les blocs synthétisés contiennent un à quatre coefficients. Le module à coefficient unique est un cas particulier utilisant une architecture distincte des trois autres (ni registre à décalage ni compteur).

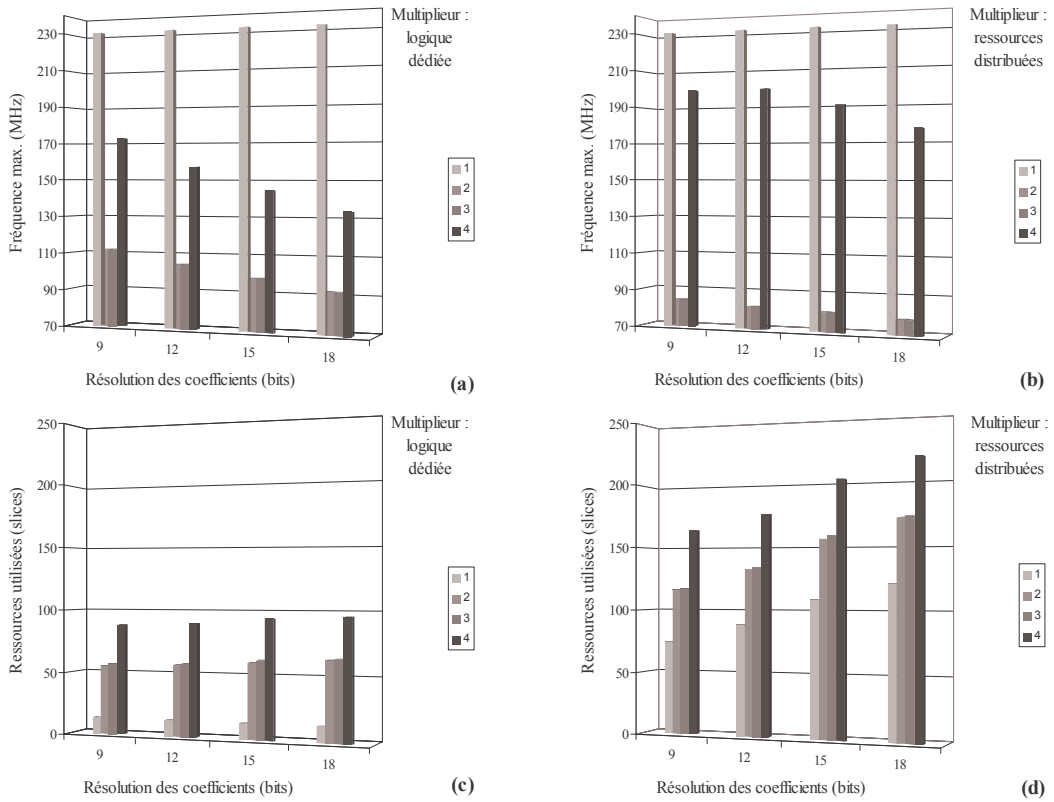


Figure 3.38. Etude paramétrique pour des modules de filtrage à rebouclage fixe.

Les fréquences maximales d'utilisation (a et b), toujours limitées par le multiplieur dans les cas à 3 et 4 coefficients, sont les mêmes que dans les cas où la reconfiguration du nombre de coefficients est permise. Pour un module à un seul coefficient, la fréquence maximale est d'environ 230 MHz. Dans ce cas, le multiplieur n'est plus associé à une mémoire lisant cycliquement les différents coefficients mais à une constante unique. L'analyse temporelle du module montre que l'additionneur est l'élément critique, le multiplieur ne limite pas la fréquence dans ce cas.

Un bloc ne contenant qu'un seul coefficient permet de fixer un point de comparaison pour évaluer l'intérêt du module de filtrage. Il n'utilise que 13 *slices* et un multiplieur (c) dans une première version et environ 50 à 100 *slices* supplémentaires lorsque le multiplieur utilise la logique distribuée (d), confirmant la conclusion donnée à partir de la Figure 3.36. Les mêmes graphiques c. et d. ne montrent qu'une légère différence pour l'implémentation d'un bloc contenant 2 ou 3 coefficients. En effet, ces deux architectures utilisent les mêmes options de synthèse pour le multiplieur et pour l'additionneur.

Ces performances doivent être réévaluées pour tout nouveau composant cible. L'utilisation d'un FPGA Virtex II-6 (plus rapide que le -4 considéré jusqu'ici) conduit à des résultats de synthèse indiquant des fréquences d'utilisation de l'ordre du double de celles qui sont présentées.

Une étude spécifique de la consommation électrique pourrait être menée afin de donner une information supplémentaire sur les performances. Une série de mesures automatisées sur

FPGA permettrait d'extraire un modèle paramétrique haut niveau de la consommation [Ell05].
Ces différents résultats donnent au concepteur des éléments lui permettant de choisir une architecture de module de filtrage et les options de synthèse associées. A partir de ces choix, il est possible d'évaluer *a priori* les ressources et les performances d'un filtre RIF connaissant la fréquence utile du système ainsi que le nombre de coefficients et leur résolution. Ces derniers points sont détaillés dans la suite de ce chapitre.

3.3 Exploration architecturale

La partie précédente expose les différentes architectures de filtres utilisées pour l'exploration architecturale appliquée à la tête numérique RLR. Les similitudes de ces filtres conduisent au développement d'un module de filtrage de base. A partir de cette étude, les méthodes d'optimisation proposées dans ce paragraphe vont permettre d'effectuer le lien entre les spécifications et la synthèse matérielle en s'intéressant en premier lieu à la fonction de filtrage. L'objectif final est d'obtenir un flot de conception systématique de la TRN globale.

3.3.1 Synthèse et optimisation des filtres

L'obtention du modèle synthétisable de chaque filtre numérique à mettre en œuvre nécessite trois étapes. Dans un premier temps, les **spécifications** permettent d'identifier les paramètres à optimiser. Cette opération correspond à la première étape du flot présenté sur la Figure 3.11. Ensuite, l'**optimisation** de la fonction de filtrage permet d'obtenir l'architecture de filtre à implémenter (nombre et types de filtres) ainsi que ses dimensions (ordre de filtrage, coefficients…). La **résolution des coefficients** est déterminante pour les performances du filtre, son optimisation est donc l'objet d'une attention particulière. Ces deux dernières étapes sont représentées par le deuxième bloc de la Figure 3.11.
Ce flot aboutit à la mise en place d'un outil logiciel permettant d'automatiser ces différents processus.

3.3.1.1 Spécifications et objectifs

Le point de départ d'une exploration architecturale est la définition d'objectifs à atteindre. Concernant les fonctions de filtrage, ces objectifs sont classiquement un gabarit. Il peut être défini de deux manières distinctes : soit il correspond à la description des bandes passantes et stoppées et de leurs atténuations respectives, soit à une liste de points dont les coordonnées (*fréquence* ; *atténuation*) sont indiquées. Des tolérances sur ces gabarits donnent une certaine liberté aux fonctions d'optimisation et permettent, dans la mesure du possible, de réduire significativement les dimensions du filtre résultant.
Dans le cadre de la RLR, les filtres utilisés peuvent être multifréquences. Il faut donc définir le facteur de conversion de fréquence d'échantillonnage, l'architecture du système de filtrage dépendant fortement de ce paramètre (cf. §3.2.1).
Dans une perspective de réalisation matérielle, la taille des coefficients doit également être optimisée. La méthode présentée plus loin s'appuie sur les contraintes de gabarit (forme et tolérances) pour trouver la résolution minimum satisfaisant aux spécifications.

3.3.1.2 Optimisation des filtres

3.3.1.2.1 Filtre CIC

Les relations (3.7) et (3.8) présentent la réponse en fréquence des filtres CIC dans le cas général. Les variables de ces équations sont *R*, le rapport entier de conversion de fréquence d'échantillonnage, *N*, le nombre d'étages du filtre et *M*, le délai différentiel des blocs *comb* (cf. §3.2.1.2.1). Chacun de ces paramètres étant entier, la réponse d'un filtre CIC ne peut pas être optimisée à proprement parler. *A priori*, *R* est connu puisque c'est le facteur de CFE que le filtre doit assurer ; seuls *M* et *N* doivent donc être dimensionnés.

Les algorithmes utilisés ici sont itératifs (Figure 3.39). Ils consistent à imbriquer deux boucles : l'une incrémente *M* successivement pour chaque valeur de *N*, l'autre incrémente *N*. Le programme d'optimisation sélectionne l'architecture la plus proche des spécifications.

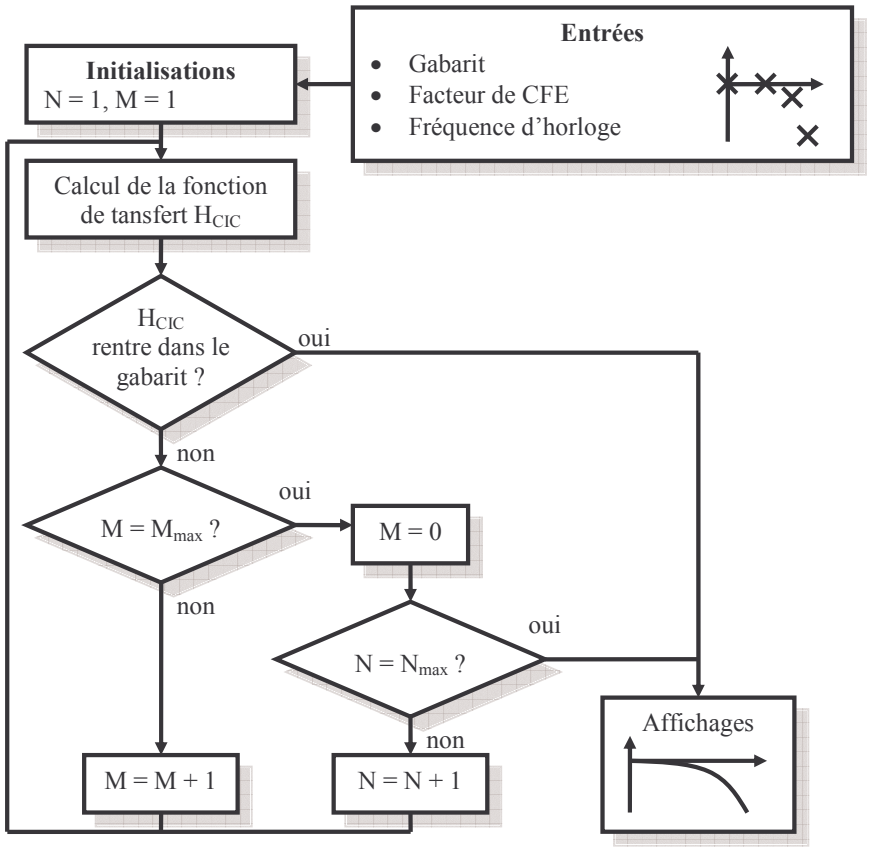


Figure 3.39. Organigramme d'optimisation de filtre CIC.

Pour trouver le filtre optimal, le programme mémorise au fur et à mesure l'architecture la plus proche du gabarit initial. Il indique le meilleur résultat après avoir exploré les différentes possibilités ce qui implique que *M* et *N* doivent être bornés (M_{max} et N_{max}).

3.3.1.2.2 Filtre RIF direct

Le paragraphe 3.2.1.2.2 présente l'architecture et le fonctionnement du filtre RIF direct. Trois méthodes mathématiques permettent d'optimiser ce système : la méthode de la fenêtre, la méthode de l'échantillonnage en fréquence et l'algorithme Parks-McClellan.

Méthode de la fenêtre

Cette méthode consiste en premier lieu à identifier la fonction de transfert $H(f)$ à l'expression d'une transformée de Fourier des signaux discrets (TFSD) tronquée sur N termes. Pour cela, il faut définir le gain complexe $H(f)$ à synthétiser selon la parité de N ((3.27) et (3.28)).

N impair : $$h(n) = \int_{-1/2}^{1/2} H(f) e^{j2\pi n f} df \tag{3.27}$$

N pair : $$h(n) = \int_{-1/2}^{1/2} H(f) e^{j\pi f(2n+1)} df \tag{3.28}$$

Le fait d'identifier $H(f)$ à une TFSD tronquée revient à multiplier la réponse impulsionnelle correspondante $h(n)$ par une fonction rectangle ou encore à convoluer $H(f)$ par $W_h(f)$, transformée de Fourier d'une fonction rectangle (3.29).

$$W_h(f) = \frac{\sin(N\pi f)}{\sin(\pi f)} e^{j\pi(N-1)f} \tag{3.29}$$

La fonction synthétisée présente alors une ondulation dans la bande et des lobes secondaires. Des fenêtres de pondération remplacent alors la fonction rectangle afin de compenser ces défauts. Les propriétés de ces fenêtres permettent de répondre à différents compromis entre précision spectrale et atténuation : Hanning, Bartlett, Blackmann, Kaiser, Chebichev… Ces fenêtres induisent également différents comportements dans les bandes passantes de la fonction de transfert : ondulation (ripple) ou constante.

Méthode de l'échantillonnage en fréquence (TFD)

La méthode de l'échantillonnage en fréquence consiste à partir des objectifs de filtrage pour aboutir à la réponse impulsionnelle correspondante à l'aide d'une transformée de Fourier discrète (TFD). Un échantillonnage en fréquence de la fonction de filtrage est effectué avec un pas de F_c / N, puis la TFD inverse est appliquée à ces N échantillons pour calculer les coefficients (3.30).

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) e^{j2\pi \frac{nk}{N}} \tag{3.30}$$

$H(k)$ est la suite des échantillons fréquentiels avec $k = 0, \ldots, N\text{-}1$
$h(n)$ est la suite des échantillons de la réponse impulsionnelle avec $n = 0, \ldots, N\text{-}1$

A partir de cette fonction, une fonction d'optimisation telle que la méthode des moindres carrés est appliquée avec pour objectif de trouver l'ordre *N* minimum satisfaisant aux contraintes de filtrage.

Algorithme Parks-McClellan

L'algorithme Parks-McClellan utilise les théories d'approximation de Chebichev. Les filtres ainsi conçus présentent une ondulation dans les bandes passantes ce qui induit des discontinuités aux extrémités de la réponse impulsionnelle.

Implémentation logicielle

Un programme a été développé afin d'optimiser l'ordre de filtres RIF directs selon un gabarit indiqué par l'utilisateur et de calculer les coefficients correspondants (Figure 3.40). Il met en œuvre des fonctions Matlab prédéfines [Mat]. Il se décompose en trois étapes. L'ordre *N* du filtre RIF est tout d'abord évalué à l'aide des méthodes décrites précédemment, la technique conduisant à l'ordre le plus faible est sélectionnée puis utilisée pour calculer les coefficients finaux.

Dans ce programme, le gabarit peut être décrit soit par fronts (les frontières des bandes passantes et des bandes stoppées sont indiquées) soit point par point (le filtre optimisé doit s'approcher au maximum de ces différents points).

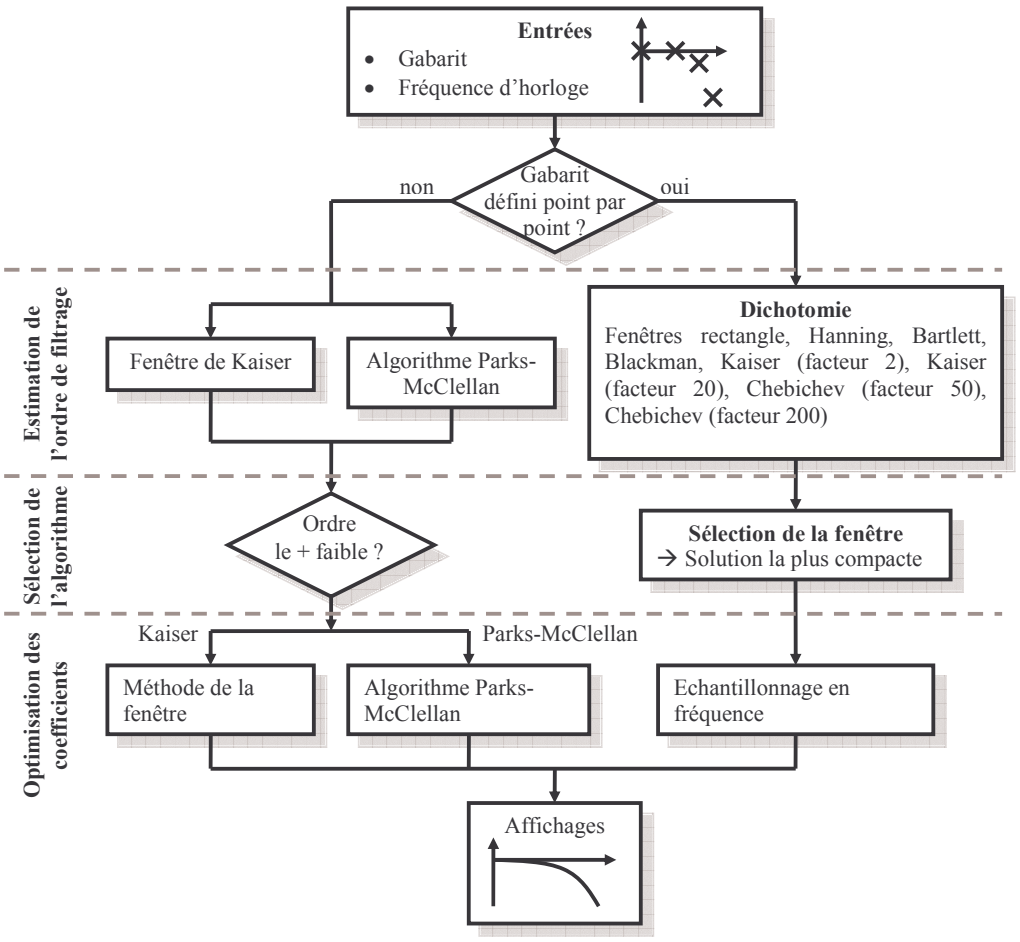


Figure 3.40. Organigramme du programme d'optimisation de filtres RIF directs.

Lorsque le gabarit est défini par fronts, la solution la plus compacte est choisie en comparant les résultats indiqués par l'utilisation de fonctions prédéfinies utilisant la fenêtre de Kaiser ou l'algorithme Parks-McClellan.
Pour un gabarit défini point par point, la sélection se base sur l'utilisation de la méthode de la fenêtre.
Le programme compare les résultats obtenus pour les 8 fenêtres de pondération suivantes :

- Rectangle
- Hanning
- Bartlett
- Blackman
- Kaiser, facteur de forme 2
- Kaiser, facteur de forme 20
- Chebichev, facteur de forme 50
- Chebichev, facteur de forme 200

Ces fenêtres sont choisies (et dimensionnées pour les fenêtres de Kaiser et Chebichev) afin de présenter une certaine diversité de caractéristiques. Une dichotomie permet de trouver l'ordre correspondant à la synthèse d'un filtre RIF pour chaque fenêtre. La fenêtre sélectionnée correspond à celle aboutissant à l'ordre le plus faible. Enfin, l'utilisation de fonctions prédéfinies permet d'obtenir les coefficients du filtre.

3.3.1.2.3 Filtre RIF polyphase

L'optimisation d'un filtre polyphase s'appuie directement sur l'optimisation du filtre RIF direct équivalent (cf. §3.2.1.2.3). Dans une première étape, le facteur de CFE R du filtre n'est pas pris en compte, et un filtre RIF direct est optimisé. La deuxième étape consiste à répartir ces coefficients entre R branches suivant les relations (3.10) et (3.11) et la Figure 3.41.

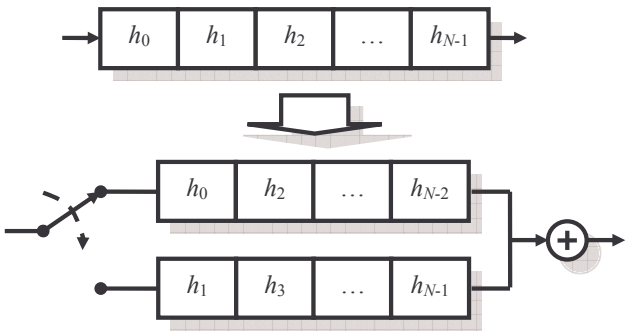


Figure 3.41. Répartition des coefficients dans un RIF polyphase décimateur.

3.3.1.2.4 Filtre de Farrow

L'optimisation d'un filtre de Farrow se base également sur la synthèse d'un RIF direct équivalent. La réponse impulsionnelle ainsi obtenue est alors décomposée en segments de fonctions polynomiales (cf. équations du §3.2.1.2.4 et Figure 3.20). Un filtre de Farrow est défini par deux dimensions : l'ordre de filtrage N issu de l'optimisation du filtre RIF direct équivalent et l'ordre polynomial M optimisé de façon itérative (cf. Figure 3.42). Plus

l'ordre polynomial est élevé, plus la fonction de transfert du filtre polynomial sera proche de celle du filtre RIF direct de départ. En contrepartie, plus *M* est élevé, plus le filtre synthétisé utilise de coefficients.
Dans un souci d'économie de ressources, seuls les filtres de Farrow modifiés sont considérés.

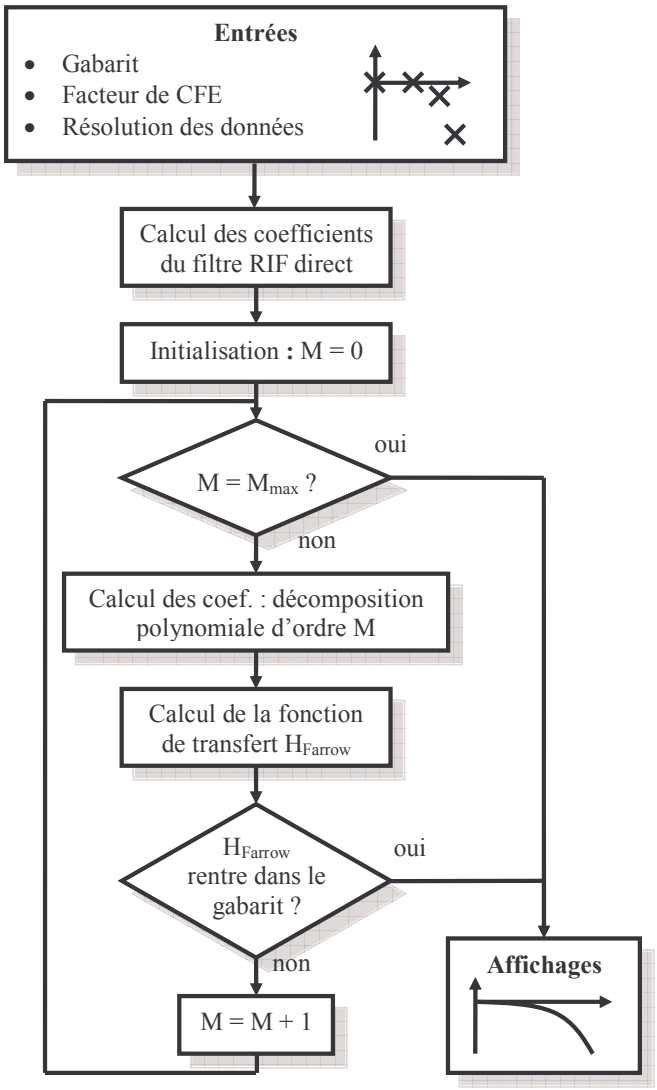


Figure 3.42. Organigramme du programme d'optimisation de filtres de Farrow.

L'itération est stoppée à partir du moment ou les spécifications de départ sont respectées. La Figure 3.43 et la Figure 3.44 illustrent l'influence de l'ordre polynomial sur la réponse en fréquence d'un filtre de Farrow modifié.

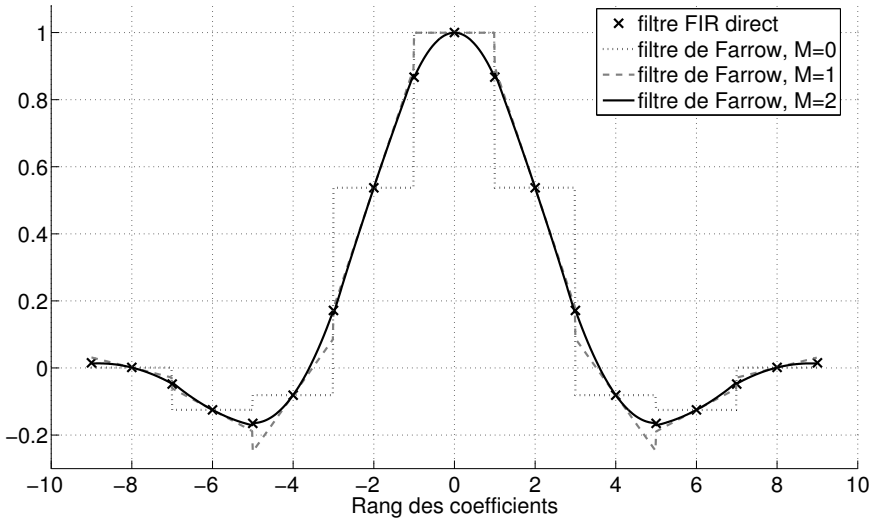


Figure 3.43. Réponse impulsionnelle d'un filtre de Farrow en fonction de l'ordre polynomial M.

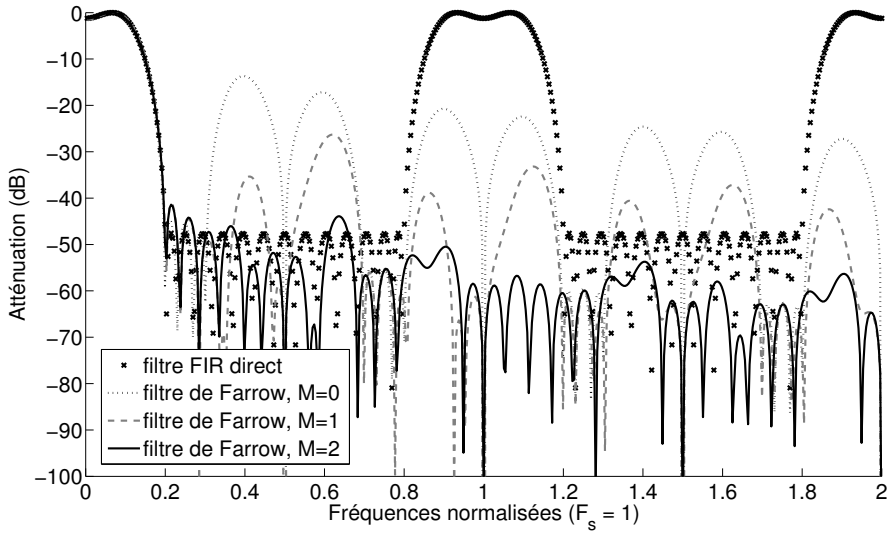


Figure 3.44. Fonction de transfert de filtres de Farrow en fonction de l'ordre polynomial M.

Les réponses illustrées sur les deux figures précédentes rendent compte de plusieurs phénomènes.

La Figure 3.43 est une application de la Figure 3.21 concernant les filtres de Farrow modifiés. La longueur des segments de réponse impulsionnelle recouvre exactement deux coefficients de filtre RIF direct initial et non pas un comme dans le cas du filtre de Farrow direct. La réponse impulsionnelle d'ordre 2 est presque confondue avec les points de la réponse impulsionnelle discrète servant de point de départ à l'optimisation.

Ce constat se retrouve sur la Figure 3.44 indiquant une bonne cohérence entre le filtre RIF direct et le filtre de Farrow modifié d'ordre deux. La fonction de transfert étant une transformée de Fourier discrète de la réponse impulsionnelle, les discontinuités des réponses impulsionnelles d'ordre 0 et 1 induisent des harmoniques dans la bande stoppée bien au-dessus de l'atténuation désirée.

La fonction de transfert des filtres de Farrow n'est pas périodique contrairement aux filtres numériques classiques. Ce phénomène met en évidence le caractère continu de ces systèmes.
L'ordre 2 semble suffire à l'obtention d'une réponse satisfaisante. Cependant, lorsque les contraintes sont plus strictes (atténuation plus forte dans les bandes stoppées, fronts de gabarit plus abrupts) des ordres polynomiaux plus élevés sont requis.

3.3.1.3 Optimisation de la résolution des coefficients

L'optimisation de la valeur des coefficients des filtres RIF est menée classiquement de façon numérique en double précision (les filtres CIC n'étant pas concernés par cet aspect). Les valeurs obtenues ne sont donc pas matériellement réalistes pour une implémentation matérielle. Une méthode simple a donc été mise au point afin d'éviter d'effectuer un choix arbitraire de la résolution des coefficients (par exemple, choix d'une résolution identique pour les données à traiter et pour les coefficients). Deux cas peuvent alors se produire : soit la résolution des coefficients n'est pas suffisante pour respecter le gabarit, soit il est possible qu'une résolution inférieure puisse être satisfaisante et conduise à un filtrage vérifiant les contraintes. Or, au vu des résultats du §3.2.2.3.5 (synthèse du module de filtrage), il semble important d'ajuster au mieux ce paramètre. La Figure 3.45 illustre ces observations pour un filtre RIF direct passe-bas.

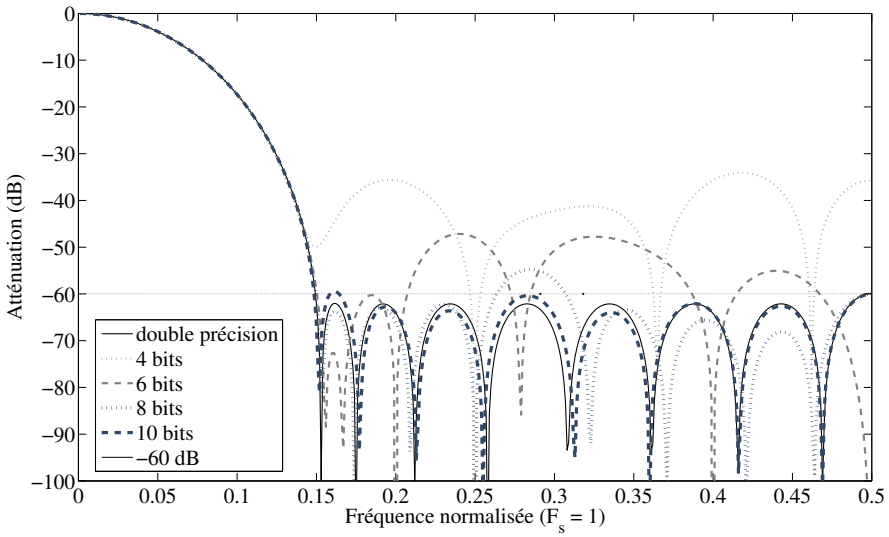


Figure 3.45. Influence de la résolution des coefficients sur la fonction de transfert d'un filtre RIF.

Sur l'exemple ci-dessus, une résolution de 10 bits suffit à obtenir un filtre satisfaisant aux contraintes d'atténuation de -60 dB. Les filtres ayant des coefficients de résolution supérieure ont une fonction de transfert se rapprochant naturellement de celle tracée pour des coefficients en double précision.
L'optimisation de la résolution des coefficients est effectuée de façon itérative comme le montre la Figure 3.46.

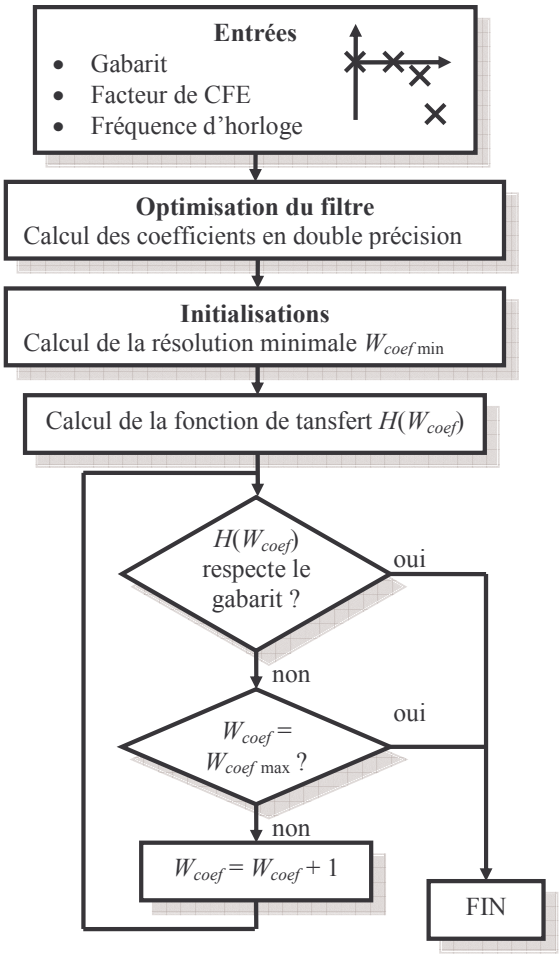


Figure 3.46. Organigramme d'optimisation de la résolution des coefficients.

La résolution minimale des coefficients correspond au pas nécessaire pour décrire le coefficient le plus petit en valeur absolue. Elle se calcule comme suit :

$$W_{coef\,min} = \left\lfloor \log_2\left(\frac{1}{\min(coef)}\right) \right\rfloor \tag{3.31}$$

Coef est un vecteur de coefficients normalisés de telle sorte que le plus grand soit égal à 1.

La rapidité d'exécution de cette méthode conduit à intégrer cette optimisation au flot de conception de la Figure 3.11.

Une approche analytique pourrait être adoptée, à l'exemple de [Lop02] pour la résolution de la variable μ des filtres de Farrow. La méthode décrite dans cette référence pourrait également être utilisée dans le programme d'optimisation de filtres Farrow. A ce jour, la résolution de μ est arbitraire : elle correspond à la taille du bus d'entrée du filtre.

3.3.1.4 Programme d'optimisation des filtres

Un outil logiciel (Figure 3.47) a été développé pour intégrer les différentes techniques d'optimisation décrites dans cette partie. Il utilise l'heuristique présentée sur la Figure 3.26.

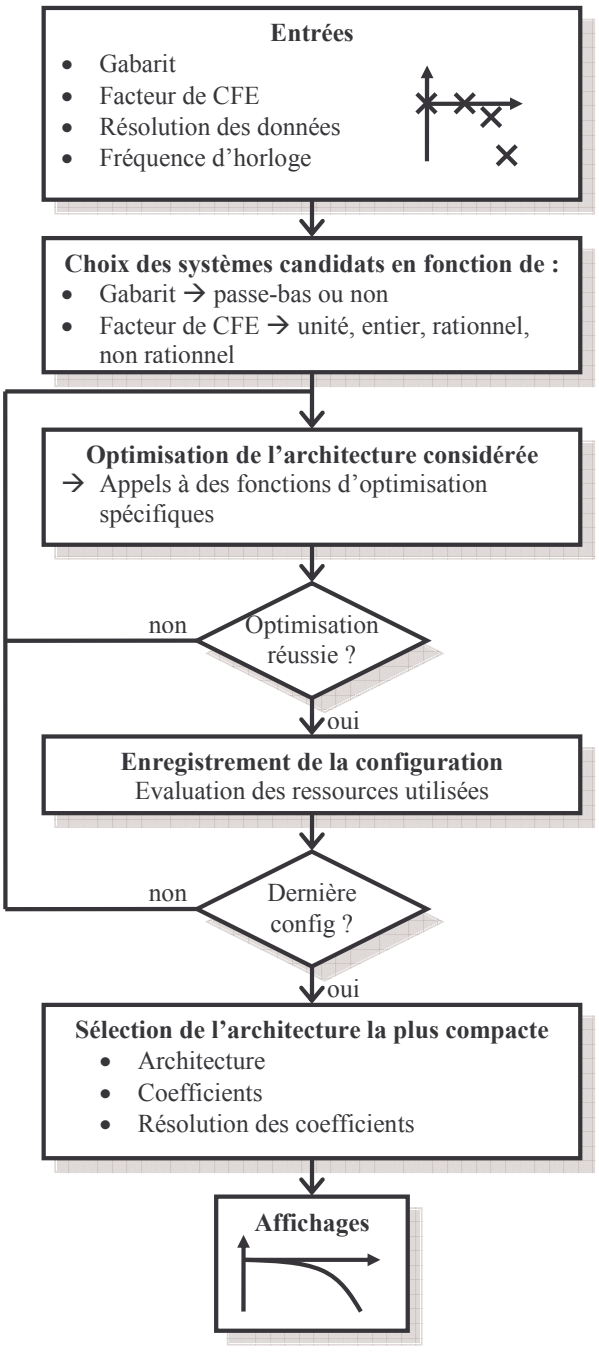


Figure 3.47. Organigramme du programme d'optimisation des filtres.

Cet outil est l'un des éléments du flot de conception proposé sur la Figure 3.11. L'étape suivante est la génération de code VHDL synthétisable à partir des résultats d'optimisation de ce programme.

3.3.2 Outil de génération de code VHDL

Le passage des résultats d'optimisation de filtres vers la génération de code VHDL doit être automatique afin d'éviter toute intervention de l'utilisateur à ce niveau. Cet outil peut être utilisé soit à la suite du programme d'optimisation, soit de façon autonome si l'utilisateur dispose d'une liste de coefficients ainsi que du dimensionnement de l'architecture correspondante.
La génération du code des filtres est détaillée. Ces filtres sont utilisés dans un second temps en tant que blocs constitutifs de la TRN.

3.3.2.1 Fonctionnement de l'outil de synthèse de code VHDL

La Figure 3.48 introduit les différents aspects de cette partie et résume les points concernant les outils méthodologiques développés tout au long de ce chapitre.

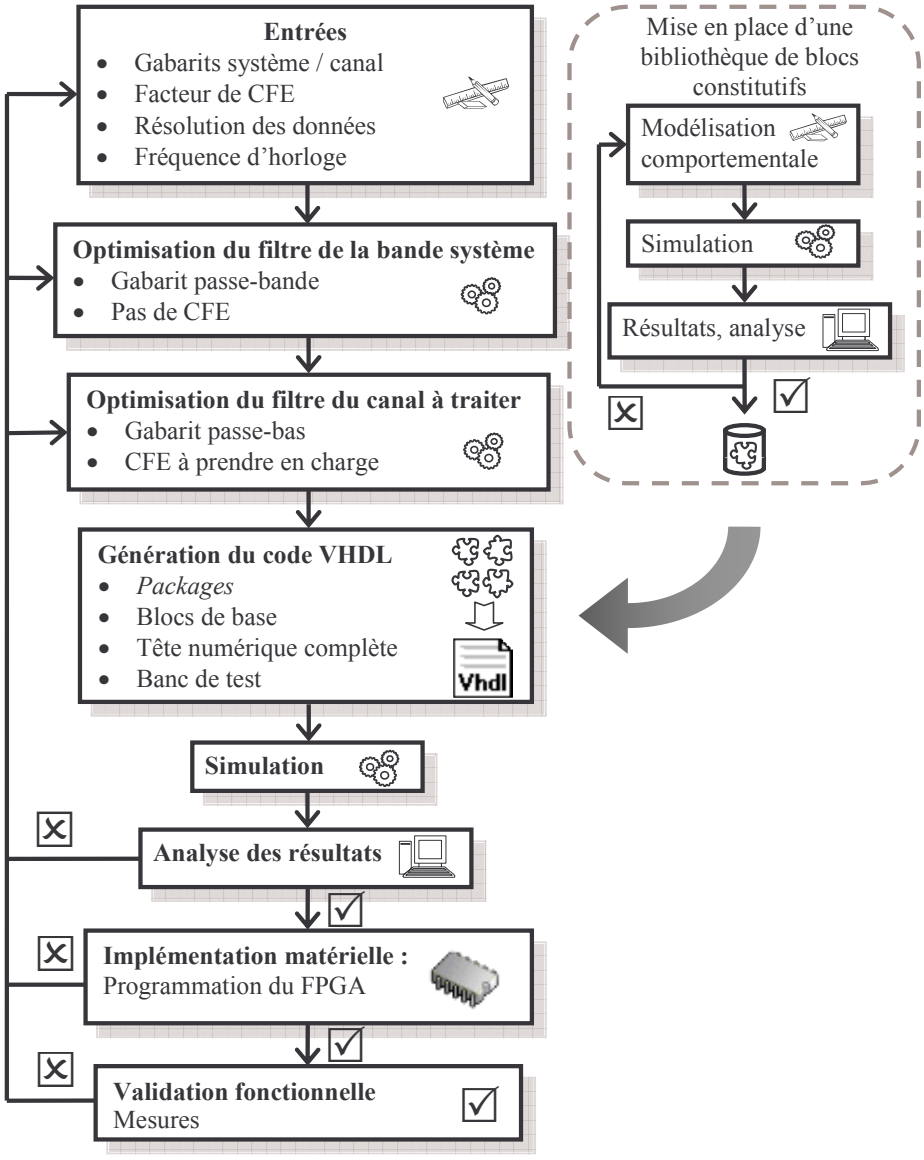


Figure 3.48. Organigramme du programme de génération du code VHDL de la TRN.

3.3.2.2 Génération du code des filtres

3.3.2.2.1 Organisation

Cet outil utilise le module de filtrage, il doit donc prévoir le nombre et la répartition des coefficients de chaque module.

Il est organisé de façon hiérarchique. A un premier niveau, il génère le code VHDL des blocs constitutifs des filtres, le code des packages de constantes et de types, et les fichiers permettant de décrire les IP fournies par le constructeur du FPGA (multiplieurs et additionneurs essentiellement). Le niveau supérieur consiste à regrouper ces entités constituant le filtre et à gérer la paramétrisation. Il comporte donc deux parties : une zone d'instanciations (code structurel) et un zone de contrôle (code comportemental). Le plus haut niveau hiérarchique est la génération éventuelle d'un banc de test. Ce dernier point est développé pour la TRN complète au §3.3.2.4.

3.3.2.2.2 Package

Les *packages* contiennent toutes les données génériques des filtres. Il existe un *package* par filtre. Ces données peuvent être séparées en deux catégories : les constantes et les types. Les constantes représentent la taille des différents vecteurs de données et des valeurs d'initialisation. Les types sont des vecteurs ou des matrices permettant d'interconnecter les entités constitutives entre elles.

3.3.2.2.3 Entité commune à tous les filtres

Tous les filtres générés, y compris les filtres CIC, présentent la même entité VHDL (Figure 3.49). Elle est similaire à celle d'un module de filtrage (Figure 3.34).

```
ENTITY fen_test_canal1 IS
PORT(    clk : in std_logic ;
   en : in std_logic ;
   entree : in std_logic_vector(Wdata-1 DOWNTO 0) ;
   wr : in std_logic ;
   addr : in std_logic_vector(1 DOWNTO 0) ;
   param : in std_logic_vector(Wparam-1 DOWNTO 0) ;
   clk_out : out std_logic ;
   sortie : out std_logic_vector(Wdata-1 DOWNTO 0)) ;
```

Figure 3.49. Code VHDL de l'entité du module de filtrage.

Les définitions des différents accès de cette entité sont présentées dans le Tableau 3.3.

Port	Direction	Usage
clk	in	Horloge pilotant le bloc RIF
en	in	Enable : le bloc fonction si EN = 1
entree	in	Bus de données (en entrée)
sortie	out	Bus de données (en sortie)
wr	in	Permission d'écrire une nouvelle donnée de configuration lorsque wr = 1
addr	in	Adresse du paramètre à remplacer
param	in	Valeur du paramètre à remplacer
clk_out	out	Horloge après changement de fréquence d'échantillonnage (inopérant pour les RIF directs)

Tableau 3.3. Entrées/sorties d'un filtre RIF quelconque.

Les ports *entree* et *sortie* sont des vecteurs de taille identique.

Le couple de valeurs *addr* et *param* a pour rôle de gérer la paramétrisation du filtre.
Les deux bits de poids faible de *addr* permettent de sélectionner le type de paramétrisation souhaité. Ce peut être le remplacement d'un coefficient, le changement du facteur de rebouclage C_m (cf. §3.2.2.3.4, (3.25) et (3.26)), d'une des valeurs R, M ou N pour un filtre CIC et μ pour un filtre de Farrow.
Les autres bits de *addr* sont utilisés pour repérer les coefficients dans la matrice de modules de filtrage. Ils représentent donc trois coordonnées : la position du coefficient dans le module, la position du module dans la branche de filtrage et éventuellement la position de la branche de filtrage pour les filtres polyphases et Farrow.
Le vecteur *addr* a donc une taille déterminée pour chaque système et la taille de chaque section est calculée puis enregistrée dans un *package*. Pour les filtres CIC, le vecteur *addr* est uniquement défini sur 2 bits puisque ces structures ne comportent pas de coefficients. Seuls les 2 bits correspondant au choix du type de paramétrisation sont conservés.

Le signal *clk_out* indique la fréquence utile du filtre cadencé à l'aide du signal *clk* (cf. (3.26)). Il est utilisé en tant qu'horloge pour piloter les systèmes en amont du filtre multifréquence en question.

3.3.2.2.4 Blocs constitutifs

Le Tableau 3.4 présente la liste des blocs nécessaires à chaque filtre [Bar05b].

RIF direct	RIF polyphase interpolateur	RIF polyphase décimateur	Farrow	CIC
Modules de filtrage				Bloc *comb*
	Interpolateur Multiplexeur	Décimateur Additionneur Démultiplexeur	Bloc de calcul de μ Chaîne de multiplieurs /additionneurs Décimateur, Délai	Bloc intégrateur Interpolateur ou décimateur

Tableau 3.4. Liste des blocs à synthétiser pour chaque architecture de filtre.

Les besoins spécifiques de paramétrisation dynamique nécessitent la mise en place d'une bibliothèque dédiée. Ainsi, à l'image du module de filtrage générique, tous les blocs ont été décrits pour l'application. Selon les cas, ils sont définis de façon comportementale uniquement, ou de façon comportementale et structurelle. Cette dernière méthode de description inclut généralement des opérateurs de multiplication ou d'addition.

3.3.2.3 Génération automatique du code de la TRN

La mise à disposition des filtres optimisés permet d'envisager la construction de la TRN et le développement d'outils correspondant aux dernières étapes de la Figure 3.11.

La fréquence d'horloge et la résolution des données en entrée de la TRN, sont soit estimées et optimisées avant le travail d'optimisation de la TRN, soit imposées par des contraintes matérielles (fréquence d'un quartz et CAN disponibles). Les gabarits de filtrage et le rapport de CFE sont imposés par le standard de communication choisi. La synthèse et l'optimisation des filtres doivent répondre aux contraintes de départ et s'effectuent à l'intérieur de l'espace de conception détaillé au §3.2.

La TRN synthétisée doit répondre à l'architecture donnée sur la Figure 3.1. Au même titre que pour un filtre seul, sa description comprend une zone structurelle d'instanciations et une zone comportementale de contrôle.
La génération du code de la TRN se déroule en quatre étapes. Tout d'abord, le filtre de la bande système est optimisé selon la méthode présentée au §3.2 et son code VHDL est généré. Ensuite, cette même opération est effectuée pour le filtrage du canal à isoler dans la bande système. Le code du système de démodulation I / Q est alors généré et consiste en l'écriture de codes correspondant aux multiplieurs et au NCO de la Figure 3.1. A partir des fichiers qui ont été créés jusqu'alors, ces trois sous systèmes sont interconnectés pour créer le code VHDL de la TRN, c'est-à-dire l'entité la plus élevée dans la hiérarchie.

3.3.2.4 Génération guidée du code du banc de test

Dans le but d'améliorer la rapidité et l'efficacité du test et du développement de la TRN générée, un outil assiste l'écriture du code VHDL d'un banc de test. Il permet de sélectionner la fréquence de l'horloge globale, le signal d'entrée (Tableau 3.5), le délai

avant déclenchement de l'*enable* (entrée *EN*) et la paramétrisation d'un filtre ou de la TRN complète. Cette opération correspond à la description des entrées *param* et *addr* vues précédemment, à la différence près que deux bits supplémentaires sur *addr* désignent le filtre concerné par le nouveau paramètre dans la TRN.

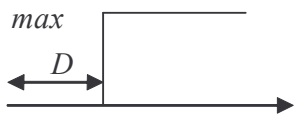
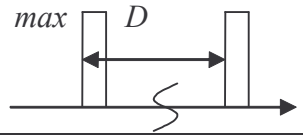
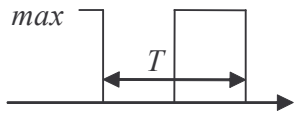
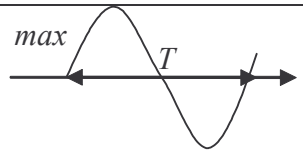
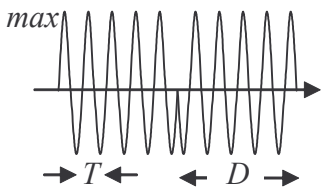
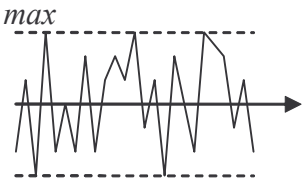
Type de signal	Paramètres	Tracé
Réponse indicielle	Délai de déclenchement du front (D)	
Série de réponses impulsionnelles	Délai (D) entre deux impulsions de durée $1/f_{clk}$	
Signal carré (de rapport cyclique ½)	Période (T)	
Signal sinusoïdal	Période (T)	
Signal QPSK	Période de la porteuse (T) Durée d'un mot (D) Les mots sont soit définis manuellement soit choisis en mode aléatoire	
Bruit blanc	Nombre total de points	

Tableau 3.5. Formes de signaux utilisés dans le banc de test VHDL.

Max est la plus grande valeur positive que peut prendre le signal d'entrée, ce qui correspond à 011…111 en binaire complément à deux. Il est à noter que dans les trois derniers cas, le signal correspond à une liste de points enregistrés dans un fichier texte et lus lors de la simulation. Les autres signaux, plus simples, sont directement calculés en cours de simulation.

3.3.3 Synthèse matérielle de filtres : étude de cas

La dernière étape de ce chapitre met en œuvre la partie optimisation et génération du code VHDL des filtres de la figure précédente. Le but de ce travail est d'illustrer l'étude préliminaire concernant l'heuristique de sélection de filtre en fonction de la nature du facteur de CFE (Figure 3.26). Elle donne également des ordres de grandeurs de

performances de différents filtres synthétisés et s'avère être une étape indispensable vers la synthèse de la TRN complète.
Afin de ne pas exclure les filtres CIC, le gabarit de filtrage donné en exemple est de type passe-bas (Tableau 3.6).

	Fréquences normalisées	Atténuation (dB)	Tolérance (dB)
Bande passante	0 – 0,02	0< A <-1	1
Bande de transition	0,02 – 0,12	-1< A <-60	-
Bande stoppée	0,12-0,5	A <-60	2

Tableau 3.6. Gabarit de l'étude comparative.

Ces contraintes sont peu restrictives, leur rôle étant de donner une base d'étude.
Dans tous les cas présentés, les données sont sur 12 bits, de même que les coefficients. Ces derniers ne sont pas optimisés en résolution afin d'établir un comparatif utilisant le même bloc de filtrage élémentaire.
Des résultats bancs de tests simulés sont proposés en Annexe D. Ils rendent compte du fonctionnement interne des filtres développés et illustrent les capacités de reconfiguration de ceux-ci.

3.3.3.1 Filtre RIF direct

Le filtre RIF direct est optimisé en utilisant la méthode de la fenêtre associée à une fenêtre de Kaiser (choix automatiques du programme d'optimisation). La fonction de transfert résultante présentée sur la Figure 3.50 indique une bonne adéquation entre les contraintes de départ et le résultat d'optimisation.

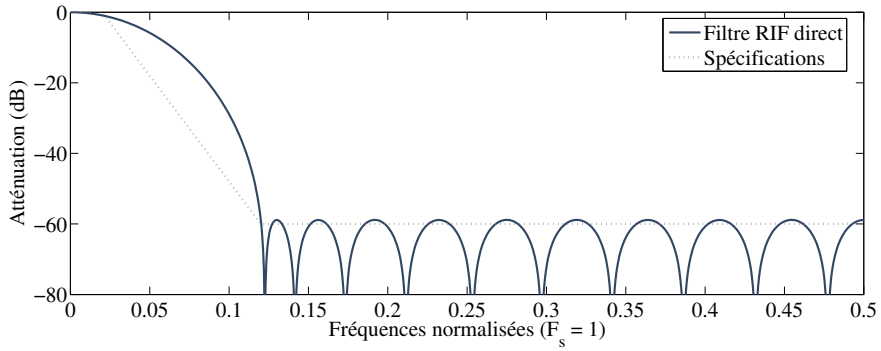


Figure 3.50. Fonction de transfert du filtre RIF direct.

23 coefficients sont nécessaires pour obtenir ce résultat. Ils sont répartis dans une chaîne de 4 modules de filtrage optimisés en fréquence et contenant chacun 6 coefficients (il existe donc une case mémoire de coefficient inutilisée, elle est donc initialisée à zéro). Reprenant la démarche de caractérisation du module de base, les performances sont évaluées après synthèse (outils Leonardo [Men]) et placement routage (outils ISE [Xil]) sur un composant cible Xilinx Virtex II comptant un million de portes (Tableau 3.7).

Multiplieurs câblés	Slices	Fréquence max
0	1012 (19%)	163 MHz
4	669 (13%)	157 MHz

Tableau 3.7. Ressources utilisées par le filtre RIF direct synthétisé.

Ces résultats doivent être comparés aux performances d'un module de filtrage (Figure 3.36). La quantité de ressources utilisées est environ le quadruple de celles d'un module seul (sachant que la Figure 3.36 donne des caractéristiques pour 4 coefficients). Dans le cas où les multiplieurs câblés sont utilisés, la fréquence d'utilisation maximale permise est identique à celle d'un module seul. Si seule la ressource distribuée est mise en œuvre, la fréquence permise diminue d'environ 36 MHz, toujours par rapport à un module isolé. Ce constat peut s'expliquer par la quantité de ressources utilisées et l'effort de routage que cela implique : plus le taux d'occupation du FPGA est élevé, moins l'outil d'optimisation a de liberté pour proposer une fréquence élevée.
Dans les deux cas, la fréquence utile est d'environ 27 MHz (~160 MHz / 6 coefficients).

3.3.3.2 Filtre CIC décimateur

La même démarche est appliquée à la synthèse d'un filtre CIC décimateur de facteur 4. L'optimisation conduit à une structure comprenant 5 étages et un délai différentiel de 2 périodes d'horloge. La réponse en fréquence obtenue est différente de celle d'un filtre RIF direct. La Figure 3.51 illustre une réponse en adéquation avec les contraintes de départ et rend compte de la difficulté à répondre à un gabarit quelconque, surtout si les tolérances sont réduites.

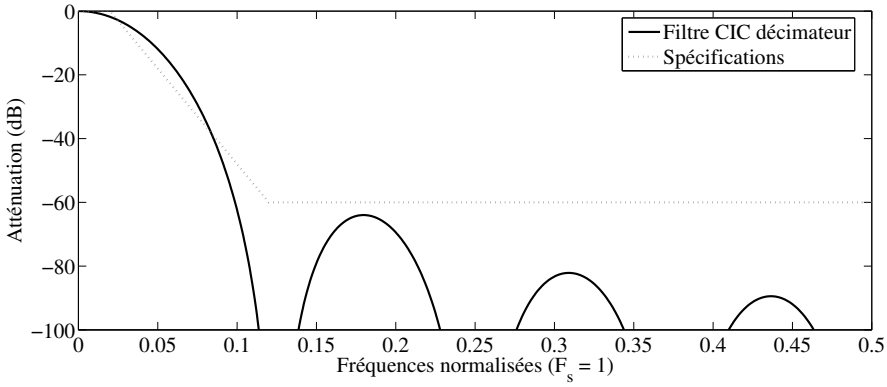


Figure 3.51. Fonction de transfert du filtre CIC.

Le filtre est synthétisé dans les mêmes conditions que le filtre RIF direct. Il requiert l'utilisation de 224 *slices* (soit 4% du FPGA), et peut être piloté avec une horloge à 180 MHz (ce qui correspond directement à la fréquence utile).
Pour développer ces observations, ces caractéristiques doivent être comparées avec celles d'un filtre polyphase décimateur équivalent.

3.3.3.3 Filtre RIF polyphase décimateur

Un filtre polyphase décimateur (décimation de facteur 4) est créé à partir des coefficients du filtre RIF direct précédent. Les résultats de synthèse sont donnés dans le Tableau 3.8.

Multiplieurs câblés	Slices	Fréquence max.
0	941 (18%)	266 MHz
4	599 (11%)	266 MHz

Tableau 3.8. Ressources utilisées par le filtre RIF polyphase décimateur.

Comparés au filtre RIF direct (Tableau 3.7), ces résultats peuvent paraître surprenants. En effet, alors que le filtre polyphase utilise le même nombre de modules de filtrage auxquels se rajoutent un additionneur et un démultiplexeur, la quantité de ressources utilisées est plus faible alors que la fréquence permise et plus élevée. Deux phénomènes entrent en jeux (la Figure 3.17 ainsi que la Figure 3.35 peuvent aider à leur compréhension).
D'une part, les modules instanciés ne sont pas cascadés ce qui suppose que l'entrée récupérant le résultat d'un potentiel bloc précédent est toujours nulle pour les quatre branches puisqu'un seul module est instancié par branche. De plus, le signal correspondant à l'entrée retardée n'est jamais utilisé. Le synthétiseur logique peut donc simplifier les modules de filtrage ce qui induit le gain de ressources constaté.
D'autre part, la partie filtrage du système est située en amont du démultiplexeur. Les modules ne sont donc pas utilisés à la fréquence d'horloge initiale, mais ils sont cadencés au quart de celle-ci. La fréquence n'est pas limitée par les blocs de filtrage, mais par le démultiplexeur ce qui justifie les fréquences maximales constatées.

Pour des performances comparables, le filtre polyphase utilise plus de quatre fois les ressources logiques nécessaires à un filtre CIC (dans le cas où les multiplieurs sont distribués). Lorsque les contraintes le permettent, le filtre CIC apparaît donc plus avantageux, mais il souffre d'un manque de flexibilité par rapport aux filtres polyphases.

3.3.3.4 Filtre de Farrow modifié

Pour aboutir à une fonction de filtrage similaire à celle d'un filtre RIF direct (Figure 3.50), l'outil d'optimisation recommande la mise en œuvre d'un filtre de Farrow modifié d'ordre 4 comprenant 44 coefficients. La structure à synthétiser [Bar05b] comporte donc 4 branches de deux modules de filtrages utilisant chacun 6 coefficients comme dans les exemples précédents. Chaque branche constitue donc un filtre RIF de 11 coefficients ce qui se rapproche de la moitié des 23 coefficients nécessaires pour un filtre RIF direct. Cette remarque concorde avec l'utilisation d'une architecture de Farrow modifiée. Les résultats de synthèse sont donnés dans le Tableau 3.9.

Multiplieurs câblés	Slices	Fréquence max.
0	2291 (44%)	163 MHz
4*2+3=11	1347 (26%)	157 MHz

Tableau 3.9. Ressources utilisées par le filtre de Farrow.

Ces résultats conduisent à des constations équivalentes à celles d'un filtre RIF direct, c'est-à-dire que la quantité de ressources utilisées concorde avec l'utilisation de 8 modules de filtrage auxquels se joignent trois multiplieurs et un bloc de calcul servant à la gestion de la variable μ. Les performances temporelles sont sensiblement les mêmes que celle du filtre RIF de base.
Pour une même capacité de filtrage, le filtre de Farrow modifié utilise plus du double des ressources nécessaires à un filtre RIF polyphase. Le filtre polyphase est donc préféré dans la mesure du possible, mais le filtre de Farrow répond à tous les facteurs de conversion de fréquence d'échantillonnage, dans la limite de la résolution du paramètre μ [Lop02].

3.3.3.5 Bilan sur la synthèse de filtres

L'exemple de filtrage développé confirme l'étude préliminaire proposée au §3.2.1. Il met en valeur l'importance d'économiser les ressources au vu des taux de remplissage du FPGA pour une application élémentaire. L'utilisation du module de filtrage s'avère utile voire indispensable à la réalisation des cas d'étude présentés. En effet, si le filtre RIF direct semble utiliser relativement peu de coefficients (23), une architecture non optimisée requerrait 23 multiplieurs ce qui peut remplir plus de la moitié du FPGA considéré si ces derniers utilisent la logique distribuée (cf. Figure 3.38). Dans de telles conditions, le filtre de Farrow modifié dépasserait largement les ressources disponibles.

3.4 Conclusion

Suivant le principe méthodologique du deuxième chapitre concernant l'exploration architecturale de CAN pipeline, l'étude proposée s'appuie sur un découpage modulaire de la TRN.

L'espace de conception concernant la TRN a été défini en deux temps.
Quatre architectures de filtre permettant de répondre aux spécifications de la tête numérique ont été décrites (FIR directs, CIC, polyphases et architecture Farrow modifiée).
L'exploitation des similitudes de trois de ces architectures (CIC exclu) conduit au développement d'un module de filtrage de base. Son intérêt est de faciliter la conception et la reconfigurabilité des systèmes, son optimisation étant effectuée en amont de la construction des filtres. La génération automatique de codes VHDL synthétisables est utilisée pour sa modélisation en vue de son implémentation sur FPGA.

D'autres architectures de filtres multifréquences telles que des variantes de la structure de Farrow [Bab02] pourraient être considérées dans l'espace de conception et venir enrichir le spectre des possibilités d'optimisation [Hen02, Hen00c, Bab01b].

Sur la base du module de filtrage de base, les filtres CIC, de structure régulière, pourraient bénéficier du même principe de conception que les autres filtres présentés. Une description modulaire mutualisant les ressources est envisageable.

Se basant sur les techniques d'optimisation de filtres multifréquences, un outil a été développé afin d'automatiser la synthèse matérielle de la TRN à partir des spécifications initiales. Il aboutit à la mise en place d'un flot de conception systématique complet.

Une étude de cas sur la synthèse de filtres retrace les grandes lignes de ce chapitre. L'analyse des résultats obtenus permet d'illustrer différents points abordés antérieurement : performances du module de base, heuristique de sélection de filtres, capacités d'intégration. Ce travail pose les bases du dimensionnement d'une TRN.

Associé aux méthodologies de dimensionnement de CAN pipeline, l'outil d'exploration systématique de la TRN est mis à contribution dans le chapitre suivant dans le cadre du dimensionnement et de la réalisation d'une TRM.

Chapitre 4

Mise en œuvre

Ce dernier chapitre présente la mutualisation des études exposées dans les chapitres 2 et 3. Les méthodes et les outils développés sont utilisés dans le but de dimensionner une tête de réception mixte radio logicielle restreinte. Les standards choisis pour cet exemple sont le GSM et l'UMTS. La simulation globale de ce système, ainsi que son implémentation matérielle sont proposées.

4.1 Introduction

4.1.1 Méthodologie d'exploration de la TRM

La convergence des études menées dans les deux chapitres précédents s'appuie sur la mutualisation des outils et méthodologies développés dans les chapitres 2 (CAN pipeline) et 3 (filtres numériques). L'exploration des solutions concernant la TRM met à contributions les techniques décrites sur la Figure 2.29 et la Figure 3.11. Elle conserve les principes de modularité et de démarche de conception descendante.
L'objectif est d'aboutir au dimensionnement, à la modélisation, à la spécification et à la réalisation d'une TRM reconfigurable.

4.1.1.1 Dimensionnement

La démarche de dimensionnement d'une TRM comprend deux étapes (Figure 4.1).

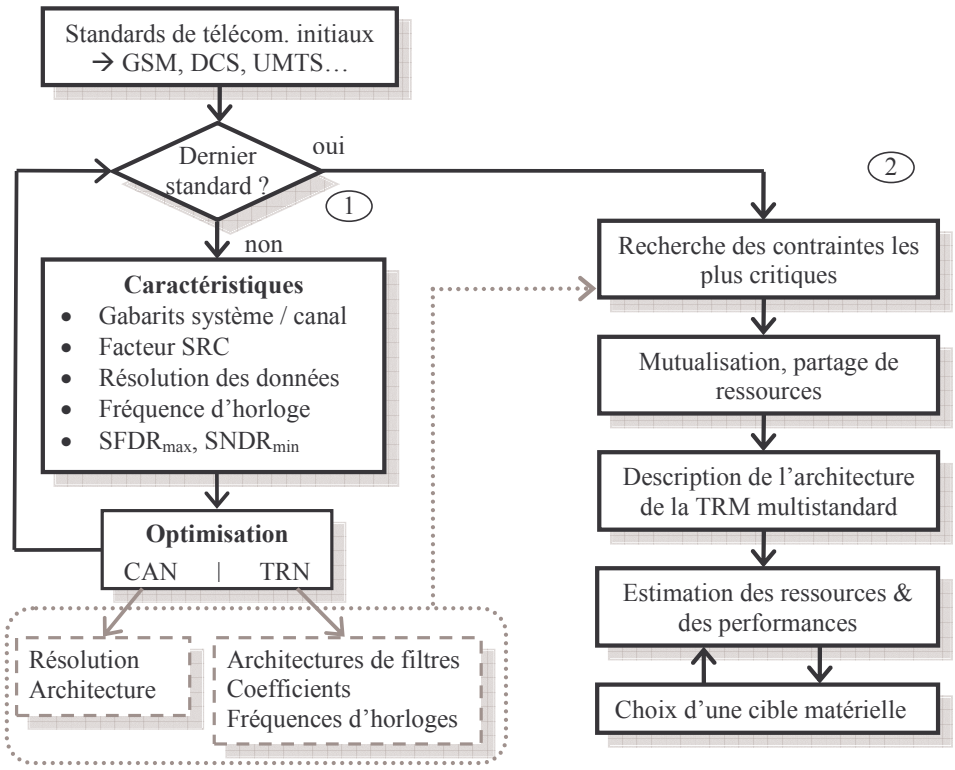


Figure 4.1. Dimensionnement d'une tête de réception numérique multistandard.

Dans un premier temps, le CAN et la TRN sont dimensionnés pour chaque standard de télécommunication considéré (branche (1)). Cette démarche met en œuvre les outils logiciels présentés dans les deux chapitres précédents.
La deuxième étape (branche (2)) prend en compte les contraintes matérielles les plus exigeantes issues de chaque optimisation afin de dimensionner la TRM reconfigurable, conçue pour répondre aux différents standards initiaux. Le découpage modulaire des filtres

et des CAN facilite ce travail et permet également une évaluation des ressources nécessaires à la mise en œuvre de la TRM.

La partie (2) du flot proposé (mutualisation des ressources, architecture finale) ne fait pas l'objet d'un développement d'outil spécifique. Il appartient à l'utilisateur d'étudier les résultats d'optimisation et de définir l'architecture de la TRM à partir de celle-ci. Le dimensionnement d'une TRM multistandard est proposé dans le §4.2.
Partant de cette architecture, les étapes suivantes consistent à générer les codes VHDL et VHDL-AMS permettant d'envisager la modélisation (simulation) et l'implantation matérielle sur carte de prototypage.

4.1.1.2 Modélisation

La mise en place d'un modèle de TRM utilise les outils de génération de codes VHDL-AMS pour le CAN et VHDL synthétisable pour la TRN. Le développement global d'un banc de test a été défini dans le premier chapitre autour d'une étude préliminaire (cf. §1.3).

4.1.1.3 Réalisation matérielle

Suivant la démarche proposée dans le premier chapitre, la réalisation matérielle s'opère autour d'une plateforme de prototypage mixte comprenant deux CAN, deux CNA et un FPGA d'un million de portes (Figure 1.19).
Ce bloc mixte peut être considéré comme un élément de base pour la conception d'un SoC-AMS [Hou02] utilisant des technologies éventuellement différentes et des protocoles de communication variés [Oua02]. Afin de s'inscrire dans cette problématique, une carte d'interface a été développée afin d'intégrer la plateforme de prototypage mixte dans un NoC (réseau sur puce ou *Network on Chip*) (Figure 4.2) [Bar04a].

Cette carte d'interface permet l'alimentation de la carte de prototypage et la gestion des protocoles de communication. La mise au point et le fonctionnement de ce système sont présentés au §4.3.

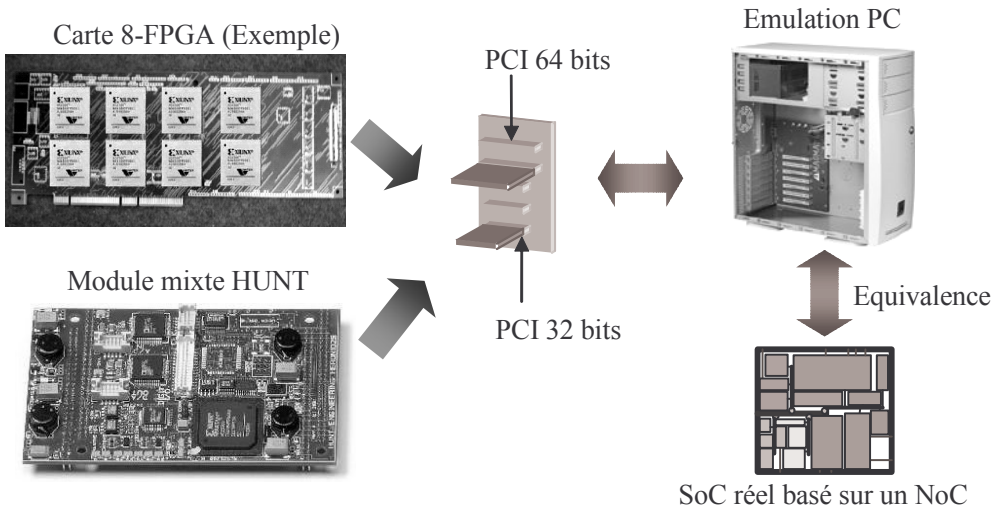


Figure 4.2. Prototypage par association de modules de traitement programmables.

Les différents modules de traitement offrent la possibilité d'effectuer un prototypage multi-composant autour d'un NoC (Network-on-Chip) utilisant un protocole de communication standard (exemple : le PCI – *Peripheral Component Interconnect*).
Cette démarche s'inscrit dans un contexte de conception basée sur des plateformes (PBD, *Platform Based Design*). La PBD correspond à une méthodologie hiérarchique englobant les potentialités de la BBD (conception par blocs, *Block-Based Design*) et offrant de larges perspectives en matière de conception de systèmes (réutilisabilité, test).

4.1.2 Cadre de l'étude

Les outils présentés sont appliqués au développement d'une TRM prenant en charge deux standards de communication : le GSM et l'UMTS [ETSI]. Les caractéristiques techniques utiles au développement de la TRM sont répertoriées dans le Tableau 4.1 [Mal01]. Elles constituent le contenu du bloc de caractéristiques de la Figure 4.1.

	GSM	UMTS (W-CDMA)
Bande système (MHz)	↑ 880 – 915 / ↓ 925 – 960 → *BW* = 35	↑ 1850 – 1910 / ↓ 1930 – 1990 → *BW* = 70
Débit	270,833 ksps	3,84 Msps
Largeur d'un canal	200 kHz	5 MHz
SNDR	> 79 dB → ENOB ~ 12 bits	> 36 dB → ENOB ~ 6 à 8 bits
SFDR	> 90 dB	> 60dB
A_{min}	88 dB	55 dB

Tableau 4.1. Spécifications des standards de télécommunication traités.

Les largeurs des bandes de réception (*BW*) permettent de calculer les fréquences de Nyquist et donc les fréquences d'échantillonnage minimales correspondantes. Elles indiquent également le gabarit des filtres passe-bande en entrée de la TRN. Les facteurs de CFE sont déduits à partir des cadences de conversion A/N et des débits en bande de base. La TRN est conçue pour fournir à sa sortie un débit double du standard [Hen99b]. Les SNDR et SFDR

minimums conduisent à évaluer le nombre de bits des CAN ainsi que leur précision aux fréquences d'utilisation prédéfinies. La largeur des canaux à traiter ainsi que l'atténuation des canaux adjacents (A_{min}) guident l'optimisation des filtres de canal.

La réalisation de la partie analogique de la tête de réception ne rentre pas dans le cadre de cette étude. La transposition des bandes systèmes reçues (en RF) fournit des bandes de fréquences intermédiaires (FI) précédant l'opération de numérisation. Le choix de ces bandes en FI doit respecter les performances offertes par les CAN actuels et en particulier les deux CAN présents sur la carte de prototypage évoquée précédemment. Il doit également prendre en compte les capacités de traitement du FPGA utilisé. L'étude préliminaire menée au chapitre 3 indique des limitations en fréquence (bloc de filtrage, filtres synthétisés). L'exemple retenu est présenté sur la Figure 4.3.

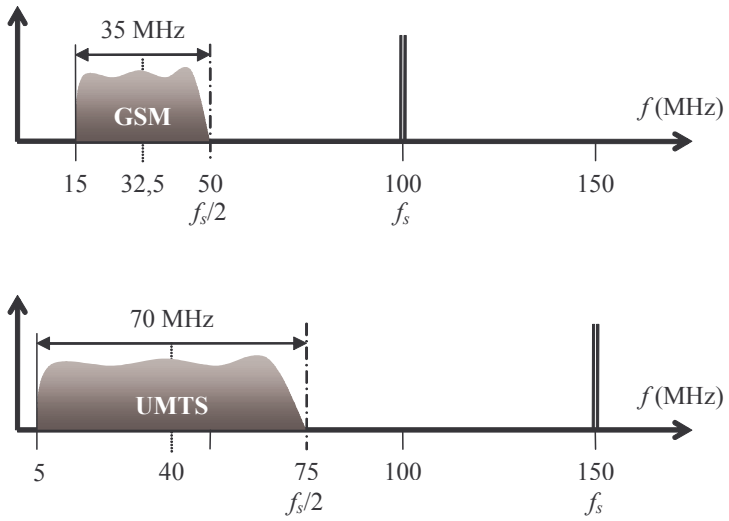


Figure 4.3. Bandes système GSM et UMTS en fréquence intermédiaire (entrée de la TRM).

Concernant les traitements UMTS, la bande système est de 70 MHz, l'échantillonnage doit s'opérer à une cadence minimum de 140 MSPS. Seule une horloge à 100 MHz est disponible sur la carte, mais elle peut être traitée par le FPGA pour fournir une fréquence différente à l'aide d'un DCM (*Digital Clock Manager*). Une fréquence d'échantillonnage de 150 MSPS utilisant un rapport 3/2 est retenue pour l'UMTS. Le GSM quant à lui utilise une bande de 35 MHz ce qui implique une fréquence d'échantillonnage minimum de 70 MSPS. L'horloge de la carte (100 MHz) répond directement à cette contrainte.
L'utilisation de fréquences d'horloge multiples des débits en bande de base est envisageable mais les rapports ne sont pas réalisables avec les DCM. Afin de montrer la généricité de la méthode, des fréquences quelconques (au regard des débits bande de base) sont choisies. Cette démarche permet de proposer une architecture de TRN générique pouvant répondre à des standards différents de ceux mis en œuvre pour ces travaux.

4.2 Dimensionnement de la tête de réception mixte

4.2.1 CAN pour la réception GSM et UMTS

4.2.1.1 Démarche

Le Tableau 4.1 expose les caractéristiques attendues pour la conversion AN. En mode GSM, le CAN doit fonctionner sur 12 bits à 70 MSPS minimum. En mode UMTS, sa résolution est fixée à 8 bits avec une cadence de 140 MSPS.

Le dimensionnement de la fonction CAN s'oriente vers deux directions.
La première consiste à utiliser les outils et les résultats du deuxième chapitre afin d'obtenir une architecture pipeline optimisée pour chaque standard en considérant la faisabilité d'un tel composant. Des solutions techniques en relation directe avec le cadre de cette étude de conception sont proposées dans [Sum02b]. Le modèle de ce système de conversion AN reconfigurable au niveau module (cf. §2.6) peut être mis en place et utilisé pour la simulation d'une TRM.
Le second volet vise à la réalisation de la TRM. Le module de prototypage disponible comporte deux CAN pipeline 12 bits fonctionnant à 105 MSPS maximum pouvant être modélisés et utilisés pour la simulation d'une TRM. Ces deux CAN peuvent être entrelacés à l'aide du FPGA associé afin de proposer une cadence de fonctionnement pouvant aller jusqu'au double de la fréquence individuelle nominale. Ce principe consiste en une reconfiguration au niveau fonction si le passage d'une configuration entrelacée vers une utilisation individuelle est prévu.

4.2.1.2 Optimisation niveau module de conversion

L'optimisation d'un CAN au niveau module de conversion utilise les méthodes et outils développés au chapitre 2. Les performances des CAN guident l'exploration architecturale de la TRM. En effet, le partitionnement analogique–numérique est intimement lié aux caractéristiques fréquentielles (f_{max}) et spectrales (SNDR, SFDR) de ces composants.
Ce travail, nécessitant des données techniques (limitations technologiques, imperfections…), est une perspective ouverte par cette étude.

4.2.1.3 Réalisation niveau système de conversion

4.2.1.3.1 Dimensionnement

La numérisation de la bande GSM avec l'un des CAN disponibles sur la carte est envisageable en termes de fréquence d'échantillonnage (jusqu'à 105 MSPS) et de bande passante (500 MHz) (cf. Tableau 4.1). En revanche, la cadence de 140 MSPS requise pour l'UMTS dépasse les capacités d'un CAN seul. Il faut donc envisager l'entrelacement des

deux CAN et la mise en œuvre d'un DCM de facteur 0,75 pour parvenir à une fréquence de conversion de 75 MSPS (Figure 4.4).

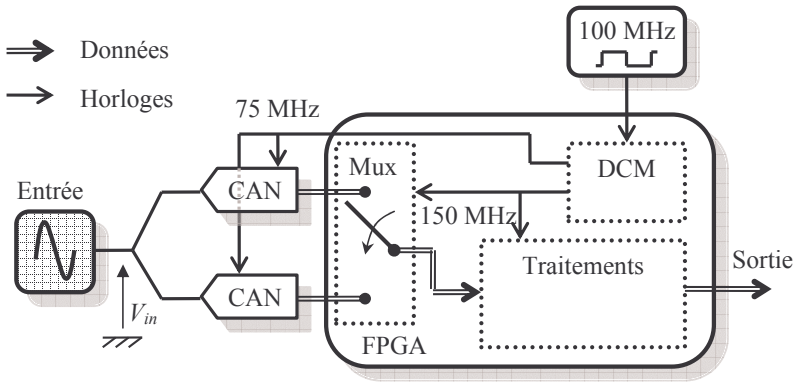


Figure 4.4. Gestion des horloges et des CAN en mode UMTS.

En amont du multiplexeur, un module de correction d'erreurs permettrait d'équilibrer les deux voies de conversion.

4.2.1.3.2 Modélisation

Le modèle VHDL-AMS générique de CAN pipeline reconfigurable est mis à contribution à cette étape. L'architecture interne du convertisseur de la carte de prototypage (AD9432 [An1]) n'est pas fournie par le constructeur. Une démarche peut être adoptée afin d'extraire une structure « hypothétique » à partir de sa documentation technique et d'obtenir un comportement se rapprochant de ses caractéristiques.

L'observation de sa NLI montre une série de « cassures » régulièrement espacées (Figure 4.5 (a)). Leur nombre peut être estimé à 16. La Figure 2.31 montre le lien entre les imperfections des amplificateurs inter-étages et la NLI. Un étage de 2 bits présentant une erreur de gain d'amplification, induit une NLI sous la forme de 4 rampes (soit 2^N). Sachant que le premier étage est le plus critique et qu'il est testé sur sa pleine échelle, une caractéristique comptant 16 « pseudo-périodes » implique l'utilisation d'un étage de tête de résolution 4 bits. Cette observation est plus aisée sur la NLI d'un AD9224 (Figure 4.5 (b)) dont le premier étage est de résolution 5 bits (données constructeur).

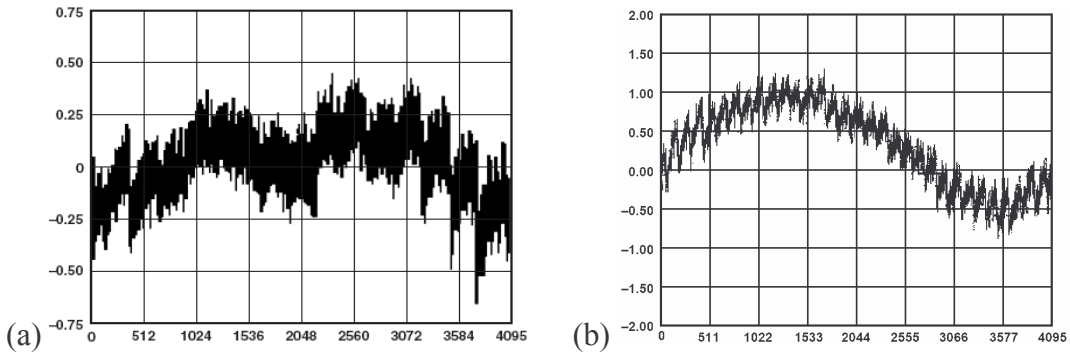


Figure 4.5. NLI des CAN AD9432 (a) et AD9224 (b) (source [An1])

Ce CAN, de résolution 12 bits, a une latence de 10 cycles d'horloge. Il est donc assez improbable qu'une synchronisation des étages par des fronts d'horloge alternés soit mise en œuvre. En estimant a un ou deux cycles d'horloge la durée nécessaire à la reconstitution du mot binaire, ce CAN doit compter 8 à 9 étages de conversion. L'essentiel du CAN serait donc principalement constitué d'une chaîne d'étages 1,5 bits voire 2 bits. Une comparaison avec d'autres circuits du même constructeur (AD9245 [An1]) conduit à penser que le premier étage est suivi d'une chaîne de 6 étages 1,5 bit et se termine par un étage de 3 bits. Il resterait donc deux périodes d'horloge pour la partie numérique du convertisseur.
Une erreur de gain ε_{gain} de -5% identique sur tous les amplificateurs du CAN modélisé permet d'obtenir un ENOB de 11,3 bits et une SFDR de 83,1 dB. Ces caractéristiques sont relativement proches des données constructeur (ENOB = 11,1 bits et SFDR = 85 dB).
Ce modèle, jugé suffisamment proche des objectifs, est conservé pour la suite de l'étude.

4.2.2 TRN pour la réception GSM et UMTS

Le dimensionnement de la TRN pour les deux standards retenus repose sur les choix de la Figure 4.3. L'utilisation du bloc de filtrage générique doit permettre d'optimiser l'implémentation des filtres. Un des objectifs de ce travail est l'obtention d'une TRN matérielle à l'aide de codes VHDL synthétisables générés automatiquement à partir des objectifs.
La conception d'une TRN répondant conjointement aux contraintes du GSM et de l'UMTS conduit à une solution sous-optimale par rapport à un standard isolé (cf. Figure 4.1) : à partir des résultats d'optimisation de chaque standard (§4.2.2.1 et §4.2.2.2), il appartient au concepteur d'étudier la mutualisation des différents systèmes (§4.2.2.3).

4.2.2.1 Optimisation de la TRN pour le GSM

Le filtre de la bande système n'effectuant pas de CFE, l'architecture optimale est le filtre FIR direct. Les objectifs indiqués dans le Tableau 4.1 sont atteints avec 55 coefficients de 16 bits.

Pour isoler le canal à traiter, différentes solutions de mise en cascade ont été présentées au chapitre 3. Les architectures à retenir sont celles pouvant assurer une CFE de facteur quelconque. Ces solutions, explorées suivant l'heuristique proposée (Figure 3.26), incluent donc un filtre de Farrow (sous sa forme modifiée afin de réduire le nombre de coefficients). Les résultats du programme d'optimisation sont classés dans le Tableau 4.2 et répondent aux objectifs fixés dans le Tableau 4.1. La fréquence d'horloge considérée est de 100 MHz.

	Filtre 1	Filtre 2	Nb coef 1	Nb coef 2	Remarques
A	Farrow		~ (M+1).6831 / 2		Encombrement rédhibitoire
B	Farrow	direct	12	77	Repliement sur le 1er filtre
C	polyphase	Farrow	184x38 = 6992	12	Filtrage sur le 1er filtre
D	polyphase	Farrow	92x2	47x5 = 235	Filtrage sur le 2ème filtre
E	Farrow	CIC			Résultats hors gabarit
F	CIC	Farrow	R=92, M=1, N=1	47x5 = 235	Solution optimale

Tableau 4.2. Résultats d'optimisation de filtres de sélection de canal GSM.

La solution A utilisant un filtre de Farrow modifié seul conduit à un nombre de coefficients trop important pour la fonction d'optimisation. Un filtre FIR direct présentant le gabarit de filtrage voulu utilise 6831 coefficients. Un filtre de Farrow modifié doit donc comporter environ 6831/2 coefficients par branche sur M+1 branches. Cette architecture étant trop encombrante, d'autres solutions doivent être explorées.

Si un filtre de Farrow est cascadé avec un filtre FIR direct (B), l'effort de filtrage doit être assuré par le deuxième bloc, le premier effectuant la CFE. Le filtre de Farrow présente donc un facteur de décimation d'environ 185 (~ 100/0,54). Un tel facteur implique un repliement du spectre dans la bande avant le deuxième filtre à moins de porter l'effort de filtrage sur le premier filtre : cette solution se rapproche alors de la première.

L'architecture C est composée d'un filtre polyphase prenant en charge le filtrage et la partie entière de le CFE, le coefficient de CFE étant ajusté par le filtre de Farrow. La structure résultante nécessite un grand nombre de coefficients et n'est pas envisagée pour une réalisation matérielle.
Si, dans la même structure, l'effort de filtrage est porté sur le deuxième filtre (D), le premier bloc évite le repliement et réduit le débit de données au minimum. Le filtrage ainsi que le facteur de CFE sont ensuite ajustés par le filtre de Farrow.

La structure Farrow – CIC (E) ne vérifie pas les contraintes.

La mise en cascade inverse (F) repose sur le même principe que l'enchaînement polyphase – Farrow lorsque l'effort de filtrage est porté sur le deuxième filtre. Pour éviter le repliement tout en gardant une caractéristique relativement constante dans la bande passante du système global, le coefficient de décimation du filtre CIC est de 92.

D et F sont les solutions les plus envisageables. Le filtre CIC nécessitant moins de ressources que le filtre polyphase, la structure F est retenue. De plus, le premier filtre devant fonctionner à 100 MHz, les perspectives de rebouclage du module de filtrages sont très restreintes (D). Le filtre de Farrow, dans les deux derniers cas fonctionne à une cadence réduite de l'ordre du MHz. Malgré son nombre de coefficients plus élevé dans l'association avec un filtre CIC, le dernier résultat est retenu (Figure 4.6).

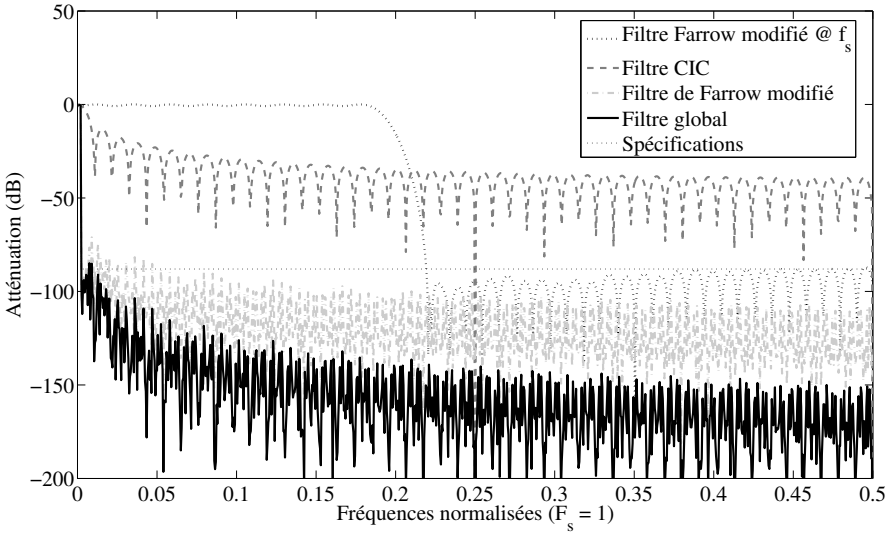


Figure 4.6. Réponse en fréquence du système de filtrage de canal GSM.

4.2.2.2 Optimisation de la TRN pour l'UMTS

L'optimisation du filtrage de la bande système conduit à une structure FIR directe contenant 53 coefficients de résolution 12 bits (Tableau 4.1).

De la même façon qu'avec le standard GSM, les solutions de filtrage impliquent un filtre de Farrow. Les résultats sont répertoriés dans le Tableau 4.3 et répondent aux objectifs du Tableau 4.1. La fréquence d'horloge considérée est de 150 MHz.

Filtre 1	Filtre 2	Nb coef 1	Nb coef 2	Remarques
Farrow		138x4		Encombrement rédhibitoire
Farrow	direct	12	15	Repliement sur le 1er filtre
polyphase	Farrow	30x10	12	Filtrage sur le 1er filtre
polyphase	Farrow	7x15	24x4	Filtrage sur le 2ème filtre
Farrow	CIC			Résultats hors gabarit
CIC	Farrow	R=7, M=1, N=1	24x4	Solution optimale

Tableau 4.3. Résultats d'optimisation de filtres de sélection de canal UMTS.

L'analyse de ces résultats conduit aux mêmes remarques que pour les filtres de sélection de canal en GSM. L'enchaînement filtre CIC – filtre de Farrow modifié est retenu (Figure 4.7).

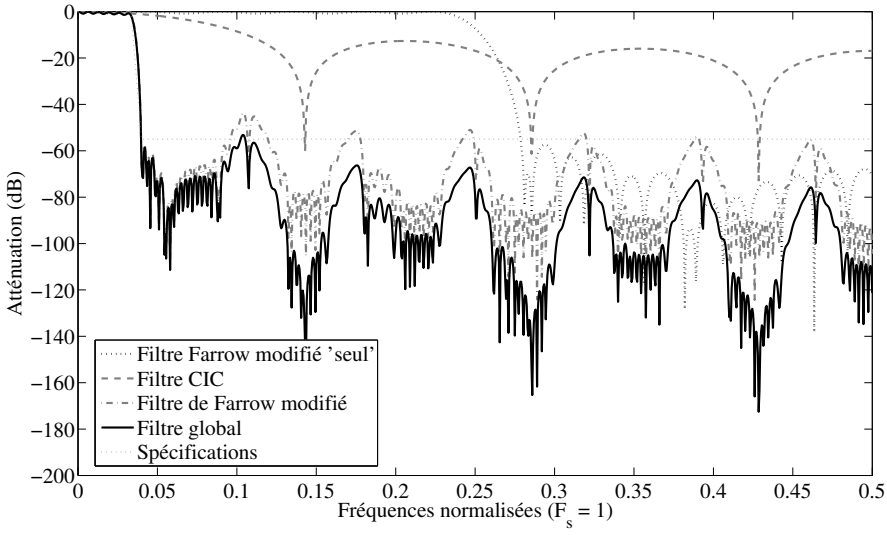


Figure 4.7. Réponse en fréquence du système de filtrage de canal UMTS.

4.2.2.3 Mutualisation

Pour répondre au caractère multistandard de la RLR, une mise en commun des ressources utilisées par chaque standard est nécessaire. Ce travail consiste à gérer les différentes horloges pilotant les blocs de traitement et à répartir les coefficients dans les modules de filtrage. Cette opération représente la seule partie non automatisée de la Figure 4.1 (partie (2)).

4.2.2.3.1 Optimisation conjointe

Les filtres de la bande système fonctionnent à 100 voire 150 MSPS selon l'application (fréquence utile). Pour bénéficier du rebouclage des modules de conversion, ces derniers doivent pouvoir être cadencés à un multiple entier de ces fréquences. Or un module contenant deux, trois ou quatre coefficients ne peut proposer qu'une fréquence utile de l'ordre de 50 MSPS. Cette contrainte impose l'utilisation de modules de filtrage non rebouclés (un coefficient) permettant d'atteindre environ 230 MSPS.

Pour les filtres multifréquences, c'est-à-dire les filtres de Farrow dans cette étude, la difficulté réside dans la nécessité de faire coïncider les fréquences utiles de chaque standard avec les fréquences d'horloge disponibles dans l'application.
La Figure 4.8 indique la technique de gestion des horloges et illustre le raisonnement suivant.
La structure globale des filtres CIC reste fixe (nombre d'étages et délai différentiel), seul le facteur de décimation est variable (7 ou 92). Sa mise en œuvre ne présente *a priori* pas de difficulté majeure au vu du résultat de synthèse (cf. §3.3.3.2). L'effort de dimensionnement doit donc se porter principalement sur le filtre de Farrow modifié.
En UMTS, sa fréquence utile est de 21,4 MSPS : l'échantillonnage à 150 MSPS est suivi d'une décimation de facteur 7. En se reportant au chapitre 3, le module de filtrage optimisé

en fréquence peut atteindre une cadence supérieure à 175 MSPS. Afin d'assurer le fonctionnement de l'application, une marge de sécurité doit être prise et les blocs seront cadencés à 150 MSPS. En UMTS, chaque module peut donc contenir 7 coefficients. Le filtre à implémenter issu de ces constatations comprend donc 4 branches de 4 modules.
Le filtre correspondant en GSM comprend un nombre important de coefficients, mais sa fréquence utile est bien inférieure (1,087 MSPS soit un facteur 20 avec l'UMTS). Cette différence permet donc le rebouclage d'un plus grand nombre de coefficients. Sachant que le filtre UMTS nécessite 4 modules par branche et que 47 coefficients sont utilisés par branche en GSM, chaque module doit inclure 12 coefficients.

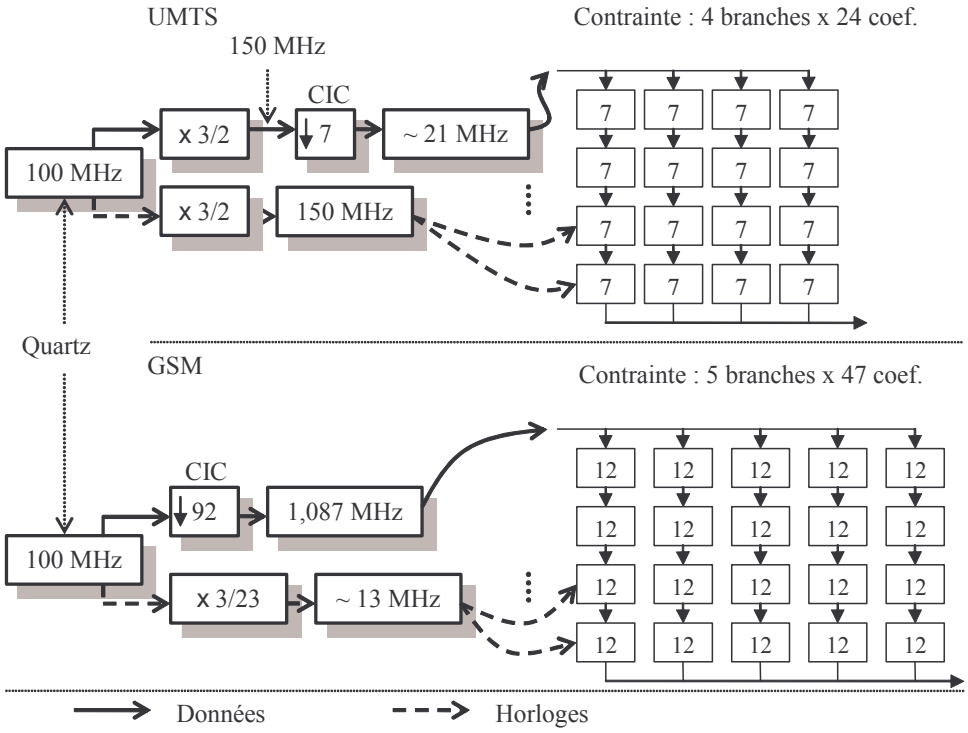


Figure 4.8. Gestion des horloges et des modules de filtrage pour le GSM et l'UMTS.

Cette figure répertorie les différents systèmes de gestion d'horloge à implémenter ainsi que le nombre de modules de filtrages nécessaires à une chaîne commune paramétrable prenant en charge les deux standards. Ces modules sont au nombre de 20 et contiennent soit 7 soit 12 coefficients.
La méthode utilisée permet d'économiser des ressources à deux niveaux. D'une part, la mise en cascade d'un filtre CIC avec le filtre de Farrow réduit le nombre de coefficients utiles de plusieurs milliers à 235 coefficients. D'autre part, chaque module de filtrage prend 12 coefficients en charge ce qui implique que 20 multiplieurs sont nécessaires au lieu de 235 pour une réalisation série.

L'optimisation de la résolution des coefficients sélectionne une taille de 12 bits pour le filtre de Farrow en UMTS et 18 bits en GSM. Une taille de 18 bits doit être choisie pour les cas, le module de filtrage ne permettant pas de reconfigurer la taille des coefficients.

4.2.2.3.2 Résultats de synthèse

Les ressources disponibles dans le FPGA de la carte de traitement sont indiquées dans le Tableau 4.4. Bien que la TRN soit optimisée, une évaluation des ressources qui lui sont nécessaires (étude paramétrique §3.2.2.3.5) prévoit un dépassement des capacités du composant. En effet, dans l'exemple proposé, la TRN met en œuvre deux filtres de Farrow contenant 20 modules chacun, deux filtres CIC et un filtre FIR direct de 55 coefficients. L'application requiert donc 105 multiplieurs.

Les résultats de synthèse de l'application, à partir des 33 fichiers générés par l'outil sont proposés dans le Tableau 4.4. La TRN est complète dans le premier cas, le filtre de la bande système est exclu dans le deuxième.

FEN	Slices	Slice Flip Flops	4 input LUTs	DCM	Multiplieurs câblés
Disponible	5120 (100%)	10240 (100%)	10240 (100%)	8 (100%)	40 (100%)
Avec filtre système	16033 (**313%**)	23857 (**232%**)	19891 (**194%**)	2 (25%)	105 (**262%**)
Sans filtre système	14185 (**277%**)	20112 (**196%**)	19094 (**186%**)	2 (25%)	50 (**125%**)

Tableau 4.4. Ressources utilisées par la TRM avec une cible Virtex II Xc2v1000.

Le Tableau 4.5 présente les ressources d'un FPGA comprenant 4 millions de portes appartenant à la famille du composant de la carte de traitement. Celui-ci pourrait accueillir l'application complète [Xil]. Avec ce FPGA comme cible matérielle, l'outil de placement routage indique une fréquence maximale d'horloge de 102 MHz, satisfaisante pour l'application dimensionnée.

	Portes	Slices	Multiplieurs	RAM	DCM	IO
Xc2V1000	10^6	5120	40	40*18k	8	432
Xc2V4000	4.10^6	23040	120	120*18k	12	912

Tableau 4.5. Ressources disponibles dans les FPGA Xilinx Virtex II Xc2V1000 et Xc2V4000.

4.2.3 Validation comportementale : simulation mixte

La simulation mixte de la TRM complète doit permettre une validation fonctionnelle du système. Le banc de test de cette application peut, dans ce cadre, prendre en compte la paramétrisation de la TRN ainsi que de la reconfiguration du CAN.

Ce travail se base sur les outils Modelsim et ADVanceMS [Men] utilisés pour l'étude préliminaire (chapitre 1). Le premier prend en charge les modèles purement numériques, alors que le second permet l'utilisation de modèles analogiques et mixtes (Figure 1.1).

La partie numérique utilise trois bibliothèques différentes. Les bibliothèques XilinxCoreLib et Unisim [Xil] contiennent les IP constructeur utilisées dans la TRN. La troisième regroupe les IP VHDL développées pour cette étude.

La partie analogique utilise uniquement une bibliothèque d'IP VHDL-AMS génériques relatives à la conversion A/N, également développées pour cette étude.

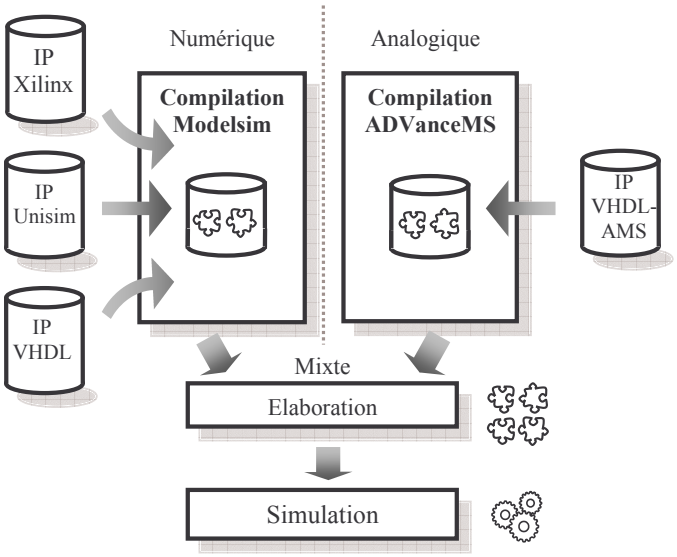


Figure 4.9. Vue logicielle de la simulation mixte.

Les fonctions analogiques/mixtes et numériques sont compilées séparément. La mise en commun des modèles est effectuée à l'étape d'élaboration. Le système peut alors être simulé et vérifié dans un même environnement logiciel.

4.3 Réalisation matérielle

La dernière étape du flot proposé sur la Figure 1.15 (principe général du flot de conception) est l'implémentation de l'application RLR. Celle-ci repose principalement sur l'utilisation de la carte de traitements FPGA dont il a été question dans les précédents chapitres (carte Hunt [Hun]). Elle est complétée par une carte d'interface spécialement développée à l'IETR et dont le fonctionnement fait l'objet de la première partie. La seconde donne et commente les résultats de cette réalisation de TRM.

4.3.1 Plateforme PCI mixte

4.3.1.1 Intérêt, objectifs

Le module mixte de prototypage, constitué de convertisseurs et d'un FPGA, peut être considéré comme un élément de base pour la conception d'un SoC (cf. Figure 4.2). La carte Hunt ne possédant pas d'accès de communication vers l'extérieur, une carte d'interface a été développée. Son rôle est d'augmenter l'efficacité du module de prototypage en facilitant la paramétrisation de l'application testée et l'échange de données avec celle-ci.

4.3.1.2 Architecture de la plateforme

L'objectif de cette plateforme est de permettre une utilisation efficace de la carte de traitement grâce à une interface de communication standard. Pour ce faire, elle remplit trois fonctions : elle adapte le format du connecteur, elle alimente la carte de traitements et elle assure le lien entre deux protocoles de transfert. Le protocole PCI [Pci] est choisi pour son caractère à la fois universel et évolutif. La version 32 bits cadencée à 33 MHz est retenue.

La carte d'interface (Figure 4.10) est principalement constituée d'un FPGA de type Spartan II [Xil] et des éléments nécessaires à l'alimentation de la carte de prototypage. Le FPGA contient une IP PCI propriétaire [Oua02] développée au sein de l'IETR et un bloc de gestion des communications avec la carte de traitement présenté dans le paragraphe suivant.

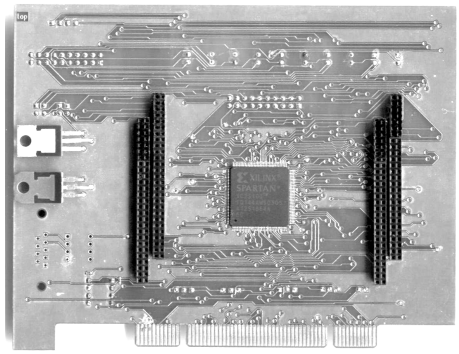


Figure 4.10. Photographie de la carte d'interface PCI.

4.3.1.3 Interfaçage des deux cartes

Le nombre de signaux partagés entre les deux cartes est limité à 38. Un bus bidirectionnel de 32 bits étant utilisé pour les données, 6 bits sont à disposition pour gérer la synchronisation (Figure 4.11).

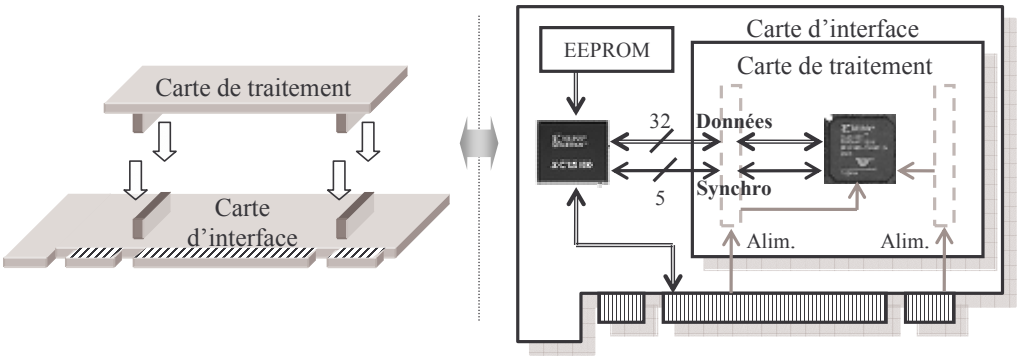


Figure 4.11. Interfaçage matériel des deux cartes.

Le protocole développé, entièrement synchrone à l'horloge du PCI, doit permettre d'effectuer deux types de transmission.
Le mode **flot de données** (appelé mode *flot* par la suite) permet d'échanger des données sur 32 bits (bidirectionnels) avec un PC via l'interface PCI. Selon l'application, il est utilisé soit pour envoyer des fichiers de points correspondant à des signaux de test, soit de récupérer des valeurs converties par les CAN puis traitées par le FPGA.
Le mode de **contrôle** (appelé *contrôle/état* ou *CE*) est utilisé pour la paramétrisation et le contrôle des différentes fonctions constituant l'application. Le mode *contrôle* permet de paramétrer les blocs constitutifs d'une application donnée (coefficients de filtres, fréquences d'horloge, …). Le mode *État* est utilisé pour récupérer des paramètres amenés à évoluer au cours du fonctionnement d'une application. Pour le mode *CE*, l'information provenant du PCI est dissociée en 24 bits bidirectionnels pour la donnée et en 8 bits pour l'adresse du paramètre à lire ou à modifier.

L'interface doit gérer deux modes ayant chacun deux directions, ce qui conduit à la définition de deux signaux : le signal *Flot/CE* (*Flot/Contrôle Etat*) et le signal *LE* (*Lecture Ecriture*). Trois autres signaux sont nécessaires au protocole développé. L'horloge provenant directement du PCI (33 MHz) peut piloter des traitements de l'application. Deux signaux *Plein* et *Vide* sont prévus en vue de l'utilisation de mémoires FIFO : ils peuvent interrompre la transmission lorsque la FIFO en écriture est pleine ou lorsque la FIFO en lecture est vide.

La Figure 4.12 est un synoptique de l'interface implantée dans le FPGA de la carte PCI. Elle consiste essentiellement à générer les bits de contrôle destinés à l'application implantée dans la carte de traitement à partir de l'IP PCI.

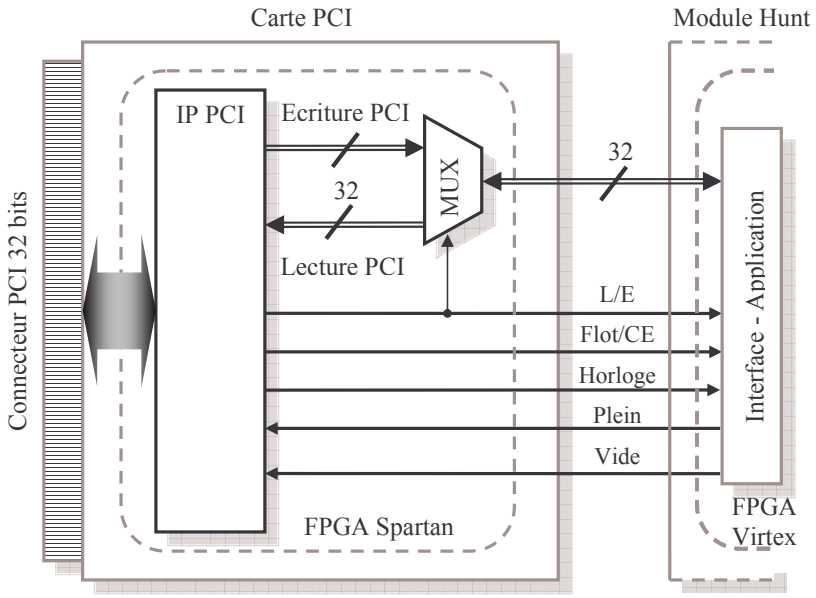


Figure 4.12. Interfaces de la carte PCI.

La Figure 4.13 représente l'interface implantée sur la carte de traitement.
Elle comporte deux FIFO jouant le rôle de tampon en mode *flot de données*. Elle génère les signaux *Plein* et *Vide* destinés à interrompre l'écriture ou la lecture du PCI. Un bloc de gestion d'horloge (*Digital Clock Manager*) associé aux FIFO permet en mode *flot* de fournir à l'application des données sur 16 bits à 66 MHz à partir de données 32 bits à 33 MHz (bus PCI). Certaines fonctions (FIFO, DCM) sont issues de la bibliothèque XilinxCoreLib [Xil]. Les FIFO pourraient également être instanciées dans la carte d'interface, mais celle-ci présente l'inconvénient de contenir moins de mémoire dédiée.
Le bloc *MUX* a pour rôle de diriger les différentes données en fonction du mode de communication sélectionné.

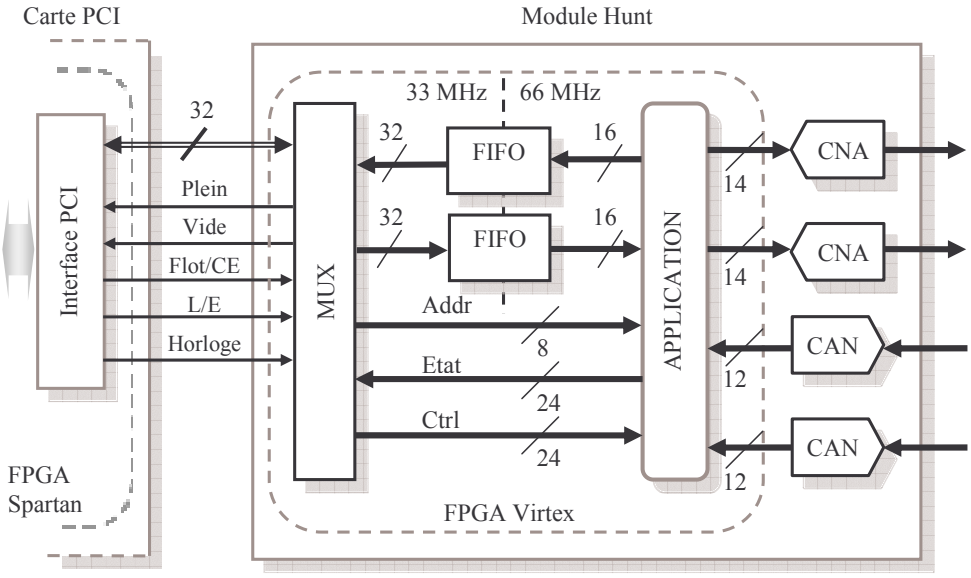


Figure 4.13. Interfaces de la carte de traitement.

4.3.1.4 Tests et applications

Les deux interfaces ont été simulées conjointement grâce au logiciel ModelSim [Men] dans le but d'optimiser puis de valider leur fonctionnement (Figure 4.14). La principale difficulté est la gestion des signaux de synchronisation en tenant compte des latences. En effet, plusieurs périodes d'horloge séparent le moment où un mode de communication est sélectionné et le moment où l'application reçoit ou envoie effectivement la première donnée.

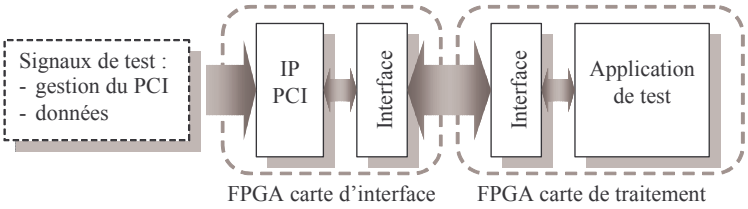


Figure 4.14. Banc de test des interfaces mises en œuvre.

Une application de test simple a été développée pour vérifier les protocoles de lecture et d'écriture de données pour chacun des deux modes (Figure 4.15). Cette application test contient deux mémoires correspondant chacune à un mode de communication. Ces mémoires stockent les valeurs écrites et ces mêmes valeurs sont renvoyées en lecture par la suite.
La Figure 4.15 illustre une succession de modes de communication et rend compte des délais nécessaires à l'IP PCI pour commuter d'un mode à l'autre.

(A) : initialisation de la carte PCI.
(B) : envoi de 6 mots de contrôle du PC vers l'application.
(C) : utilisation d'un des 6 mots de contrôle de B) en tant que donnée d'état (« *DA700000* »). *Etat* étant le mode par défaut (*L*/*E* et *Flot*/*CE* à '0'), la dernière donnée d'état est présente sur le bus 32 bits lorsque celui-ci n'est pas utilisé par un autre mode.
(D) : envoi d'un paquet de données du PC vers l'application en mode *Flot*.
(E) : envoi des données de l'application vers le PC en mode *Flot*.

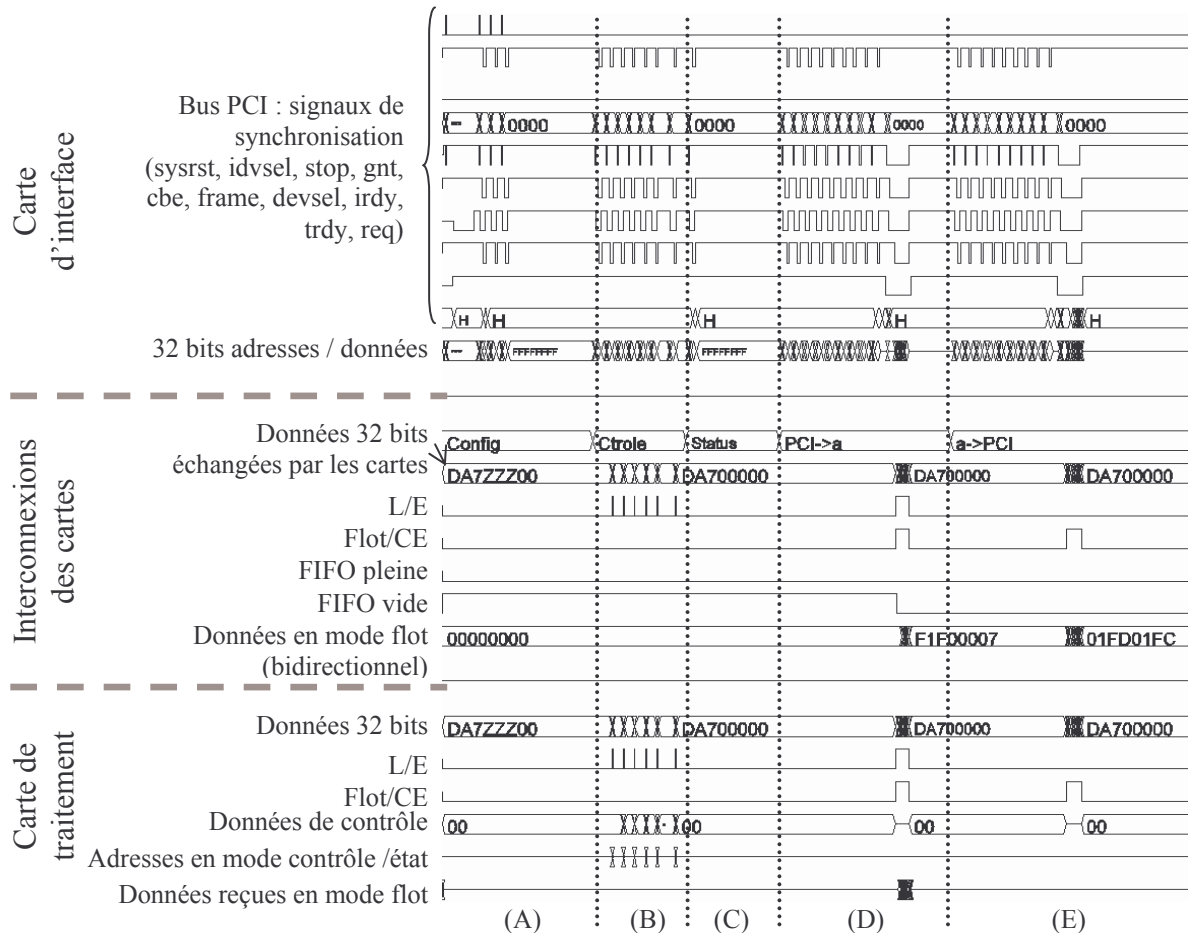


Figure 4.15. Chronogrammes du banc de test

Un utilitaire C a été développé afin de gérer l'interface avec l'utilisateur. Il permet de déclencher chaque de mode de communication. En mode *Flot*, il permet d'envoyer des fichiers de valeurs prédéfinies vers l'application ou d'enregistrer des données issues de traitements.

Ce programme propose un mode *script*. Un *script* définit la succession des modes utilisés et permet d'automatiser la gestion de l'application. Il facilite notamment l'utilisation du mode *Contrôle/Etat* et évitant à l'utilisateur d'entrer au clavier les paramètres en cours de fonctionnement (coefficients de filtre par exemple).

4.3.1.5 Perspectives

Cette plateforme met à disposition un module de prototypage mixte paramétrable. Dans la perspective d'implémenter une tête de réception mixte, les données peuvent être récupérées après chaque étage de la chaîne de traitement (FIR, multiplieurs, voies I et Q démodulées…) grâce au mode *flot*. Il est également possible de paramétrer le système ainsi que de le diagnostiquer en mode *CE*.

4.4 Conclusion

Ce chapitre a présenté des méthodes et des moyens techniques nécessaires à la mise en œuvre d'une TRM. Celle-ci utilise les différents aspects méthodologiques développés dans les chapitres 2 et 3, cœur de ce mémoire.

Une étude de cas autour des standards GSM et UMTS propose un exemple de dimensionnement des différentes fonctions de l'application.
Concernant la conversion A/N, les deux chaînes de réception ont des besoins très différents. Alors que le GSM requiert une fréquence d'échantillonnage minimum de 70 MHz, une résolution de 12 bits est nécessaire. Pour l'UMTS, l'effort se porte plus sur le débit de données (140 MHz) que sur la résolution (6 bits minimum). Dans le but d'optimiser la puissance consommée, les CAN reconfigurables offrent l'opportunité de s'adapter aux besoins en temps réel. Le modèle VHDL-AMS de CAN pipeline proposé dans le chapitre 2 peut être utilisé dans ce cadre.

L'utilisation d'une plateforme de prototypage mixte visant à la réalisation et à la validation de la TRM a été étudiée. Cette plateforme, constituée d'une carte d'interface et d'une carte de traitement, s'insère dans une démarche de *Platform Based Design*.
Afin d'assurer la communication entre l'application et un PC, elle utilise d'une part le standard PCI et d'autre part un protocole dédié synchronisant les échanges entre les deux cartes.
Un banc de test permet de valider le fonctionnement des modes de gestion de l'application.

Cette plateforme, communicante et flexible, permet d'envisager l'implémentation d'une partie de la TRN. Ce travail doit permettre la validation fonctionnelle de la chaîne de traitement globale.

Ce matériel peut être utilisé pour des applications différentes de la radio logicielle et peut s'insérer dans le prototypage de SoC intégrant un NoC.

Conclusion générale et perspectives

La mise à disposition de technologies électroniques toujours plus performantes offre la possibilité matérielle de réaliser des systèmes sur puce. Les méthodologies doivent s'adapter à ce nouveau cadre de conception afin de tirer profit des capacités que présentent les SoC tant au niveau de la densité d'intégration que de l'hétérogénéité des fonctions.

La radio logicielle, thème émergent de la recherche en télécommunication, s'inscrit dans ce contexte afin de bénéficier du champ d'exploration ouvert par les SoC. Elle a pour rôle de répondre à une diversité de fonctionnalités et de standards de télécommunication sans fil à l'aide d'une interface matérielle unique. Dans ce cadre, elle offre l'opportunité d'introduire les aspects d'universalité, d'adaptabilité et d'évolutivité dans les objets communicants.

Cette étude a permis d'élaborer un outil contribuant au dimensionnement et à la synthèse systématique d'un module analogique/numérique. Celui-ci, dédié à des applications radio logicielle, se caractérise par des traitements rapides, de faible complexité, mais nécessitant un haut degré de flexibilité. Ce dernier critère induit un partitionnement analogique/numérique favorisant les traitements numériques et par conséquent l'utilisation de circuits tels que DSP, FPGA.

Les aspects modélisation et prototypage d'une tête de réception mixte ont été développés et constituent les principaux travaux exposés dans ce mémoire.

Le premier chapitre permet de délimiter l'espace de conception. La réalisation d'un terminal radio logicielle idéal est inenvisageable à l'heure actuelle vu les performances des convertisseurs, et notamment leur fréquence d'échantillonnage. L'architecture retenue, comprenant une transposition du signal RF en FI analogique (structure superhétérodyne), correspond à une chaîne de réception de type radio logicielle restreinte.

Le CAN, placé au cœur de la problématique du partitionnement, fait l'objet du deuxième chapitre. Ce travail s'intègre dans le contexte de la mise en place de techniques de conception descendantes appliquées aux systèmes mixtes (SoC-AMS).

Un aperçu de techniques de conversion A/N courantes ainsi qu'un bref état de l'art conduit à retenir l'architecture pipeline pour l'étude de la tête de réception mixte.
Un modèle VHDL-AMS générique de ce système a été proposé et validé à partir de différentes techniques de caractérisation (méthode de l'histogramme, FFT, ajustement 4 paramètres). Il permet d'observer l'influence de défauts internes sur son comportement global et constitue une brique de base du modèle de la tête de réception mixte.
L'exploration de la structure pipeline est basée sur une fonction de mérite. Les coefficients de pondération qui constituent cette FDM sont évalués en fonction des contraintes de l'application.
Le calcul de la FDM concernant des CAN pipeline de résolution 10 bits a été présenté. La méthode employée n'utilise ni simulation temporelle, ni FFT. Les différents arrangements d'étages de conversion sont testés, partant du principe que les amplificateurs inter-étages présentent tous les mêmes défauts.
Ce travail permet de dégager un type d'architecture pipeline reflétant le meilleur compromis entre encombrement et précision spectrale (SNDR et SFDR). Celui-ci est composé d'un premier étage de conversion avec une résolution fonction de la qualité des amplificateurs de résidu, suivi par une chaîne d'étages de caractéristique 1,5 bits.
Trois niveaux de granularité de reconfiguration des CAN ont été définis (niveaux circuit, module et fonction) et offrent des perspectives techniques à ces travaux. Cet état de l'art permet d'introduire la reconfigurabilité dans le modèle VHDL-AMS proposé.

Dans la chaîne de réception RLR, la conversion A/N est suivie de traitements numériques rapides destinés à extraire les données en bande de base du signal en fréquence intermédiaire. L'étude s'est principalement orientée vers l'optimisation du filtrage multifréquence, permettant d'effectuer la sélection d'un canal et la conversion de la fréquence d'échantillonnage. Les filtres RIF utilisés appartiennent à quatre familles différentes : RIF direct, CIC, RIF polyphase et filtre de Farrow. Les possibilités de mise en cascade de ces filtres ont été discutées afin d'orienter les choix architecturaux. Ce travail a conduit à la mise en place d'une heuristique au sein d'un programme d'optimisation. Ce dernier est associé à un outil logiciel générant du code VHDL synthétisable à partir des spécifications des filtres. Il utilise un module de filtrage paramétrable commun, dimensionné en fonction des contraintes fréquentielles de l'application et visant à optimiser la quantité de ressources logiques utilisées.

La dernière partie mutualise les résultats des deux chapitres centraux de cette thèse.
Elle propose d'une part le dimensionnement de la TRN autour des standards GSM et UMTS. L'optimisation du filtrage du canal à traiter conduit dans les deux cas à une structure comprenant un filtre CIC et un filtre de Farrow modifié.
A partir de cette étude, la simulation d'un banc de test de la TRM est proposée. La construction du modèle de TRM consiste à connecter les entités VHDL de la TRN et VHDL-AMS de la conversion A/N.

D'autre part, l'implémentation matérielle et le test de l'application sur un module de prototypage est envisagée. Elle met en œuvre une carte de traitement mixte (FPGA, CAN, CNA) et une carte d'interface PCI.

A l'issue des travaux menés dans le cadre de cette thèse, les différents axes d'étude abordés ouvrent des perspectives nombreuses et variées.

Du point de vue de l'exploration architecturale des CAN, les différentes imperfections de la structure pipeline doivent être évaluées, en accord avec une technologie de réalisation. Cette étude permettrait d'approfondir la méthode proposée et de valider les résultats sur un cas concret.
L'application de cette méthodologie à d'autres architectures de CAN (ΣΔ par exemple) dans le cadre de la RLR ou avec d'autres objectifs d'application peut être envisagée.

Concernant la TRN, l'automatisation de la conception de filtres permet un gain de temps considérable. L'intervention de l'utilisateur est réduite puisque la génération des codes VHDL ainsi que les liens hiérarchiques entre entités sont transparents.
L'introduction de filtres multifréquences différents est un développement intéressant. En particulier, les filtres de Farrow présentent de nombreuses variantes (structure transposée, transposée/modifiée, prolongée) dont l'optimisation permettrait d'économiser des ressources matérielles.

Le module de filtrage peut être perfectionné notamment avec une étude de la gestion des arrondis des résultats de calcul et de son impact le long d'une chaîne de filtrage.

Une étude paramétrique de la consommation est également envisagée en partenariat avec des membres de notre laboratoire.

Annexes

Annexe A (Chapitre 2)

Calcul théorique du SFDR d'un CAN idéal

Annexe B (Chapitre 2)

Code VHDL-AMS d'un étage de conversion de CAN pipeline

Annexe C (Chapitre 3)

Etude paramétrique des modules de base

Annexe D (chapitre 3)

Simulation de filtres

Annexe A (Chapitre 2)
Calcul théorique du SFDR d'un CAN idéal

Soit $x(t) = \cos(\omega_{in} t)$ le signal d'entrée du CAN idéal.
Soit $y(t)$ la sortie du CAN.

$$\begin{aligned} y(t) &= q(x(t)) \\ &= \sum_{i=1}^{2^n} q_i(\cos(\omega_{in} t), x_i, x_{i+1}) \end{aligned} \tag{A.1}$$

La décomposition de $y(t)$ en série de Fourier répond à la relation générale :

$$y(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [a_k \cos(\omega_{in} t) + b_k \sin(k\omega_{in} t)] \tag{A.2}$$

La symétrie de $y(t)$ implique que $b_k = 0$. Nous obtenons alors :

$$\begin{aligned} a_k &= \frac{2}{T} \int_0^T y(t) \cos(k\omega_{in} t) dt \\ &= \frac{4}{T} \sum_{i=1}^{2^n} \int_0^{T/2} q_i \sin((\omega_{in} t), x_i, x_{i+1}) \cos(k\omega_{in} t) dt \\ &= \frac{4}{T} \sum_{i=1}^{2^n} \int_{t(x_{i+1})}^{t(x_i)} y_i \cos(k\omega_{in} t) dt \\ &= \frac{2}{\pi k} \sum_{i=1}^{2^n} y_i [\sin(k\omega_{in} t(x_i)) - \sin(k\omega_{in} t(x_{i+1}))] \\ &= \frac{2}{\pi k} \sum_{i=1}^{2^n} y_i [\sin(k.\cos^{-1}(x_i)) - \sin(k.\cos^{-1}(x_{i+1}))] \\ &= \left.\begin{aligned} &\frac{2LSB}{\pi k} \sum_{i=1}^{2^n} \sin(k.\cos^{-1}(x_i)) \\ &= \frac{2LSB}{\pi k} \sum_{i=1}^{2^n} U_k(x_i) \end{aligned}\right\} \Leftarrow INL = 0 \end{aligned} \tag{A.3}$$

$U_k(x)$ est la $k^{ième}$ fonction polynomiale de Chebyshev de second ordre

A partir de a_k, la relation (A.4) permet de calculer le niveau des harmoniques induites par le CAN idéal.

$$HD_k \equiv 20.\log(|a_k / a_1|) \qquad en\ dB_c \tag{A.4}$$

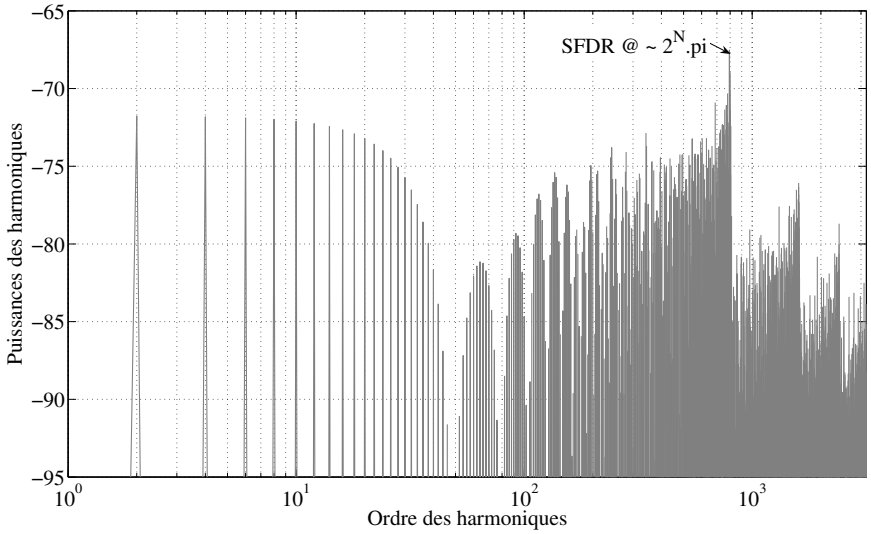


Figure A.1. Niveau des harmoniques d'un signal sinusoïdal idéalement quantifié.

Avec l'opération d'échantillonnage, ces harmoniques se trouvent repliées dans la bande $[0..f_s]$, la plus élevée constituant le SFDR. Cette démarche est complétée dans le chapitre 2 pour être appliquée à des CAN dont les paliers de conversion ne sont pas équirépartis.
La Figure A.2 indique les niveaux de SFDR calculés pour des CAN idéaux dont la résolution est de 3 à 14 bits.

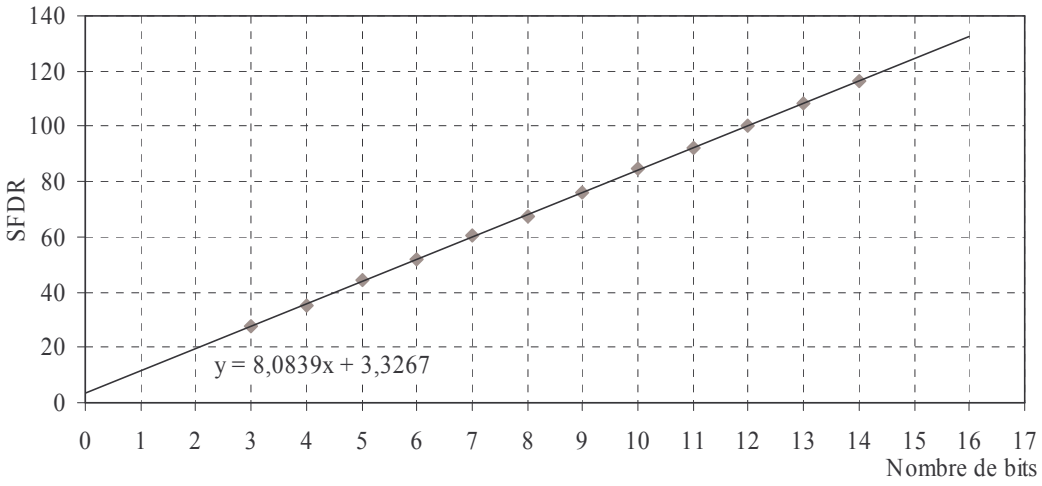


Figure A.2. SFDR théorique de CAN idéaux.

A partir de ces valeurs, une formule empirique simple est extraite afin d'évaluer le SFDR d'un CAN idéal. Une optimisation par la méthode des moindres carrés donne le résultat suivant :

$$SFDR_{id} \approx 8,1.N + 3.3 \qquad \text{(A.5)}$$

Les résultats de cette formule sont comparés aux résultats du calcul complet dans le Tableau A.1.

Résolution (bits)	SFDR analytique (dB)	SFDR éq. (A.5) (dB)	Différence (dB)
3	27,46	27,58	0,12
4	35,17	35,66	0,49
5	44,27	43,75	-0,53
6	51,76	51,83	0,07
7	60,54	59,91	-0,62
8	67,52	68,00	0,48
9	75,96	76,08	0,12
10	84,56	84,17	-0,39
11	91,98	92,25	0,27
12	100,47	100,33	-0,14
13	108,52	108,42	-0,10
14	116,27	116,50	0,23

Tableau A.1. SFDR théorique et empirique de CAN idéaux.

Une bonne adéquation entre les deux modes d'obtention des SFDR permet de valider la relation et de simplifier l'approximation (A.6) couramment utilisée.

$$SFDR_{id} \approx 9.N - n \text{ avec } 0<n<3 \text{ selon N} \qquad \text{(A.6)}$$

Annexe B (Chapitre 2)
Code VHDL-AMS d'un étage de conversion de CAN pipeline

Le code VHDL-AMS d'un étage de conversion de CAN pipeline est proposé afin d'illustrer les principes liés à son développement (§2.4.2.2). Il présente notamment l'ensemble des paramètres génériques et indique leur rôle dans le fonctionnement du système.

```
LIBRARY DISCIPLINES;
USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL ;
LIBRARY IEEE ;
USE IEEE.MATH_REAL.ALL ;

ENTITY etage_pipe_HN_nb_int_erreur_2 IS
GENERIC (   N : integer := 3 ; -- nombre de bits en sortie binaire
naturel
                  -- si N=1 => etage 1_5 bits
                  V_ref : real := 2.0 ;   -- tension de reference
                  f_c : real := 10000000000.0 ; -- frequence de coupure
            offset : real := -0.1 ; -- erreur absolue d'offset
                  erreur_gain : real := 1.1 ;   -- erreur relative de gain
                  offset_residu : real := 0.1 ;
                  NL : integer := 2 ; -- type de non linéarité /
-- 1 = polynomiale (coef1 ; coef2) ; 2 = tangente hyperbolique (coef1
slmt)
                  coef1 : real := 0.2 ; -- non linearite 2e ordre
                  coef2 : real := 0.2 ; -- non linearite 3e ordre
                  RC_SH : real := 0.000005 ; -- coef du filtrage RC du
S/H
                  retard_sortie : time := 1us ;
                  retard_ech : time := 1us) ;

PORT (TERMINAL
entree_plus,entree_moins,reste_plus,reste_moins :ELECTRICAL ;
            SIGNAL clk : in bit ;
            SIGNAL out_of_range : out bit := '0' ;
            SIGNAL sortie_num : out integer := 0 ;
            SIGNAL enable_in : in bit := '0' ;
            SIGNAL enable_out : out bit := '0') ;
END ;

ARCHITECTURE behav OF etage_pipe_HN_nb_int_erreur_2 IS
      QUANTITY V_in ACROSS I_in THROUGH entree_plus TO entree_moins ;
      QUANTITY V_out ACROSS I_out THROUGH reste_plus TO reste_moins;
      QUANTITY V_in_PB : real ;
      QUANTITY V_SH_p : real ;
      SIGNAL var2 , V_SH , LSB : real := 0.0 ;
      SIGNAL delai_ech : bit ;
      SIGNAL archi : integer RANGE 0 TO 1 := 1 ;
      SIGNAL enable_Vout : integer RANGE 0 TO 1 := 0 ;
      SIGNAL c : real := 0.0 ;

BEGIN

      V_in_PB == V_in - (1.0 / ( 2.0 * MATH_PI * f_c ) ) * V_in_PB'dot ;
      LSB <= V_ref / real(2**(N-1)) ;
      I_in == V_in_PB / 50.0 ;
```

```
      c <= tanh(V_ref*coef1)/V_ref ;

rst : process (clk)
      VARIABLE reset : bit := '0' ;
      BEGIN
            IF reset = '0' THEN
                  sortie_num <= 0 ;
                  V_SH <= 0.0 ;
                  var2 <= 0.0 ;
                        reset := '1' ;
                  enable_out <= '0' ;
                  out_of_range <= '0' ;
            END IF ;

END PROCESS rst ;

P1 : process (clk) -- process sample and hold
      BEGIN
            delai_ech <= clk AFTER retard_ech ;
END PROCESS P1 ;

P2 : process (delai_ech)
      BEGIN
      IF (delai_ech = '1') THEN
            V_SH <= V_in_PB ;
      END IF ;
END PROCESS P2 ;

V_SH_p == V_SH - RC_SH * V_SH_p'dot ;

P3 : PROCESS (clk)
      VARIABLE var_code : integer ;
BEGIN
IF enable_in = '1' THEN
      IF ( clk = '1' ) THEN
   -- etage N bits
            IF N > 1 THEN
                  archi <= 1 ;
   -- cas ou on est en dessous de la limite basse du convertisseur
                  IF ( V_in_PB < -V_ref + offset ) THEN
                        var_code := - 2**(N-1) ;
                        out_of_range <= '1' ;
                        var2 <= -V_ref + LSB / 2.0 AFTER retard_sortie ;
                  ELSE
   -- cas ou on est au dessus de la limite haute du convertisseur
                        IF ( V_in_PB > V_ref + offset ) THEN
                              var_code := 2**(N-1) - 1 ;
                              out_of_range <= '1' ;
                              var2 <= V_ref - LSB / 2.0 AFTER
retard_sortie ;
   -- cas ou entree entre -Vref et +Vref
                        ELSE
                              out_of_range <= '0' ;
                              FOR rg3 IN (-2**(N-1)+1) TO 2**(N-1) LOOP
                                    var_code := rg3-1 ;
                              EXIT WHEN (V_in_PB<(real(rg3)*LSB+offset)) ;
                              END LOOP ;
                              var2<=real(var_code)*LSB+LSB/2.0 AFTER
retard_sortie ;
                        END IF ;
                  END IF ;
```

```
-- etage 1_5 bit
            ELSE
                  archi <= 0 ;
                  IF ( v_in_PB < (-V_ref/4.0 + offset) ) THEN
                        var_code := -1 ;
                        var2 <= (-V_ref / 2.0) ;
                        IF ( v_in_PB < -V_ref + offset ) THEN
                              out_of_range <= '1' ;
                        ELSE
                              out_of_range <= '0';
                        END IF ;
                  ELSE
                        IF ( v_in_PB > (V_ref/4.0 + offset) ) THEN
                              var_code := 1 ;
                              var2 <= V_ref / 2.0 ;
                              IF ( v_in_PB > V_ref + offset ) THEN
                                    out_of_range <= '1' ;
                              ELSE
                                    out_of_range <= '0' ;
                              END IF ;
                        ELSE
                              var_code := 0 ;
                              var2 <= 0.0 ;
                        END IF ;
                  END IF ;
            END IF ;
      END IF ;
      sortie_num <= (var_code) AFTER retard_sortie ;
END IF ;
END PROCESS P3 ;

P4 : PROCESS (clk)
BEGIN
IF clk = '1' THEN
IF enable_in = '1' THEN
      enable_out <= '1' ;
      enable_Vout <= 1 ;
END IF ;
END IF ;
END PROCESS P4 ;

   -- sortie echantillonée
v_out == real(enable_Vout)*erreur_gain*(((real(abs(NL-2)) *
(real(archi*2**(N-1))+real(1-archi)*2.0)) * ((V_SH_p-var2+coef1*(V_SH_p-
var2)**2+coef2*(V_SH_p-var2)**3))+ real(abs(NL-1)) * ( real(archi)* 0.5
*(1.0/c)*tanh(coef1*(V_SH_p-var2)*real(2**N) ) + real(1-
archi)*(1.0/c)*tanh(coef1*2.0*(V_SH_p-var2) ) ) ))+ offset_residu ;

END behav ;
```

Annexe C (Chapitre 3)
Etude paramétrique des modules de base

Les tableaux présentés dans cette annexe regroupent les valeurs utilisées pour les Figures 3.36, 3.37 et 3.38.

		Multiplieur	Slices	Slice Registers	4 input LUTs	Fréq. Max (MHz)
N_{coef}	W_{coef}	40 (100%)	5120 (100%)	10240 (100%)	10240 (100%)	-
4	9	Câblé	121 (2%)	147 (1%)	151 (1%)	173
		Distribué	197 (3%)	283 (2%)	280 (2%)	199
	12	Câblé	123 (2%)	147 (1%)	157 (1%)	157
		Distribué	209 (4%)	305 (2%)	319 (3%)	199
	15	Câblé	128 (2%)	147 (1%)	163 (1%)	145
		Distribué	235 (4%)	338 (3%)	358 (3%)	190
	18	Câblé	129 (2%)	147 (1%)	169 (1%)	134
		Distribué	251 (4%)	371 (3%)	397 (3%)	177
20	9	Câblé	775 (15%)	901 (8%)	736 (7%)	173
		Distribué	851 (16%)	1037 (10%)	865 (8%)	183
	12	Câblé	779 (15%)	901 (8%)	745 (7%)	157
		Distribué	865 (16%)	1059 (10%)	907 (8%)	183
	15	Câblé	785 (15%)	901 (8%)	757 (7%)	145
		Distribué	893 (17%)	1092 (10%)	952 (9%)	176
	18	Câblé	804 (15%)	901 (8%)	782 (7%)	134
		Distribué	924 (18%)	1125 (10%)	1010 (9%)	165

Tableau C.1. Etude paramétrique appliquée au bloc optimisé en fréquence (cf. Figure 3.36).

		Multiplieur	Slices	Slice Registers	4 input LUTs	Fréq. Max (MHz)
N_{coef}	W_{coef}	40 (100%)	5120 (100%)	10240 (100%)	10240 (100%)	-
3	9	Câblé	63 (1%)	88 (1%)	99 (1%)	112
		Distribué	124 (2%)	88 (1%)	210 (2%)	85
	12	Câblé	65 (1%)	88 (1%)	105 (1%)	105
		Distribué	141 (2%)	88 (1%)	267 (2%)	82
	15	Câblé	69 (1%)	88 (1%)	111 (1%)	98
		Distribué	165 (3%)	88 (1%)	306 (2%)	81
	18	Câblé	71 (1%)	88 (1%)	117 (1%)	91
		Distribué	180 (3%)	88 (1%)	345 (3%)	78
20	9	Câblé	542 (10%)	623 (6%)	537 (5%)	107
		Distribué	603 (11%)	623 (6%)	666 (6%)	82
	12	Câblé	548 (10%)	623 (6%)	549 (5%)	100
		Distribué	624 (12%)	623 (6%)	711 (6%)	79
	15	Câblé	552 (10%)	623 (6%)	561 (5%)	94
		Distribué	649 (12%)	623 (6%)	756 (7%)	78
	18	Câblé	559 (10%)	623 (6%)	571 (5%)	89
		Distribué	669 (13%)	623 (6%)	799 (7%)	76

Tableau C.2. Etude paramétrique appliquée au bloc optimisé en ressources (cf. Figure 3.37).

N_{coef}	W_{coef}	Multiplieur	Slices	Slice Flip Flops	4 input LUTs	Fréquence max. (MHz)
		-	5120 (100%)	10240 (100%)	10240 (100%)	-
1	9bits	Câblé	13 (1%)	12 (1%)	13 (1%)	231
		Distribué	74 (1%)	12 (1%)	142 (1%)	231
	12bits	Câblé	13 (1%)	12 (1%)	13 (1%)	231
		Distribué	89 (1%)	12 (1%)	175 (1%)	231
	15bits	Câblé	13 (1%)	12 (1%)	13 (1%)	231
		Distribué	109 (%)	12 (1%)	208 (%)	231
	18bits	Câblé	13 (1%)	12 (1%)	13 (1%)	231
		Distribué	122 (2%)	12 (1%)	241 (2%)	231
2	9bits	Câblé	55 (1%)	84 (1%)	69 (1%)	112
		Distribué	116 (2%)	84 (2%)	198 (1%)	85
	12bits	Câblé	57 (1%)	84 (1%)	75 (1%)	105
		Distribué	133 (2%)	84 (2%)	237 (2%)	82
	15bits	Câblé	60 (1%)	84 (1%)	81 (1%)	98
		Distribué	156 (3%)	84 (2%)	276 (2%)	81
	18bits	Câblé	63 (1%)	84 (1%)	87 (1%)	93
		Distribué	172 (3%)	84 (3%)	315 (3%)	78
3	9bits	Câblé	56 (1%)	88 (1%)	68 (1%)	112
		Distribué	117 (2%)	88 (1%)	215 (2%)	85
	12bits	Câblé	58 (1%)	88 (1%)	92 (1%)	105
		Distribué	134 (2%)	88 (1%)	254 (2%)	82
	15bits	Câblé	62 (1%)	88 (1%)	98 (1%)	98
		Distribué	158 (3%)	88 (1%)	293 (2%)	81
	18bits	Câblé	64 (1%)	88 (1%)	104 (1%)	93
		Distribué	173 (3%)	88 (1%)	332 (3%)	78
4	9bits	Câblé	88 (1%)	122 (1%)	97 (1%)	173
		Distribué	164 (3%)	258 (2%)	226 (2%)	199
	12bits	Câblé	90 (1%)	122 (1%)	103 (1%)	157
		Distribué	176 (3%)	280 (2%)	265 (2%)	199
	15bits	Câblé	94 (1%)	122 (1%)	109 (1%)	145
		Distribué	202 (3%)	313 (3%)	304 (2%)	190
	18bits	Câblé	96 (1%)	122 (1%)	115 (1%)	134
		Distribué	218 (4%)	346 (3%)	343 (3%)	177

Tableau C.3. Etude paramétrique appliquée au bloc à rebouclage fixe (2-3-4 coefficients) et au bloc à coefficient unique (cf. Figure 3.38).

Annexe D (chapitre 3)
Simulation de filtres

Cette annexe présente des bancs de test associés aux filtres dont les résultats de synthèse sont indiqués à la fin du chapitre 3. Le gabarit de filtrage, commun à toutes les structures, est rappelé dans le Tableau D.1.

	Fréquences normalisées	Atténuation (dB)	Tolérance (dB)
Bande passante	0 – 0,02	0< A <-1	1
Bande de transition	0,02 – 0,12	-1< A <-60	-
Bande stoppée	0,12-0,5	A <-60	2

Tableau D.1. Gabarit de l'étude comparative.

Filtre FIR

Le Tableau D.2 liste les 23 coefficients relatifs au filtre FIR direct.

N°	1	2	3	4	5	6	7	8	9	10	11
coef	-0,004	0,013	0,042	0,096	0,182	0,299	0,443	0,600	0,754	0,884	0,970

12	13	14	15	16	17	18	19	20	21	22	23
1,000	0,970	0,884	0,754	0,600	0,443	0,299	0,182	0,096	0,042	0,013	-0,004

Tableau D.2. Coefficients du filtre direct.

La Figure D.1 illustre la construction de la réponse impulsionnelle correspondant à ces coefficients. Elle résulte de l'addition des résultats intermédiaires de quatre modules de filtrage contenant chacun 6 coefficients. Cette figure présente également un changement de coefficients et la réponse impulsionnelle correspondant à ces nouvelles valeurs. Dans ce deuxième cas de figure, seuls quatre coefficients par module sont utilisés.

Figure D.1. Simulation du filtre RIF direct : réponses impulsionnelles et paramétrisation.

La Figure D.2 présente le filtrage d'un signal carré à partir des deux configurations de la figure précédente.

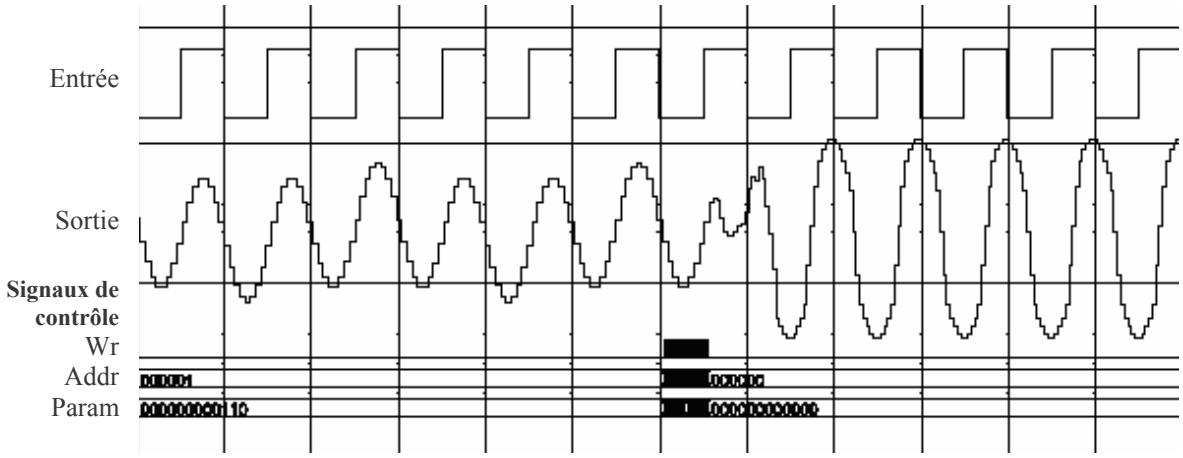


Figure D.2. Simulation du filtre RIF direct : réponse à un signal carré et paramétrisation.

Filtre CIC

Le filtre CIC décimateur comporte 3 étages intégrateurs et 3 étages *comb* (Figure D.3)

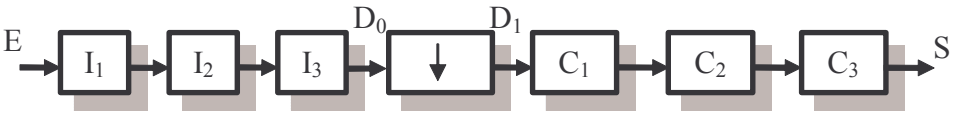


Figure D.3. Architecture CIC décimateur simulée.

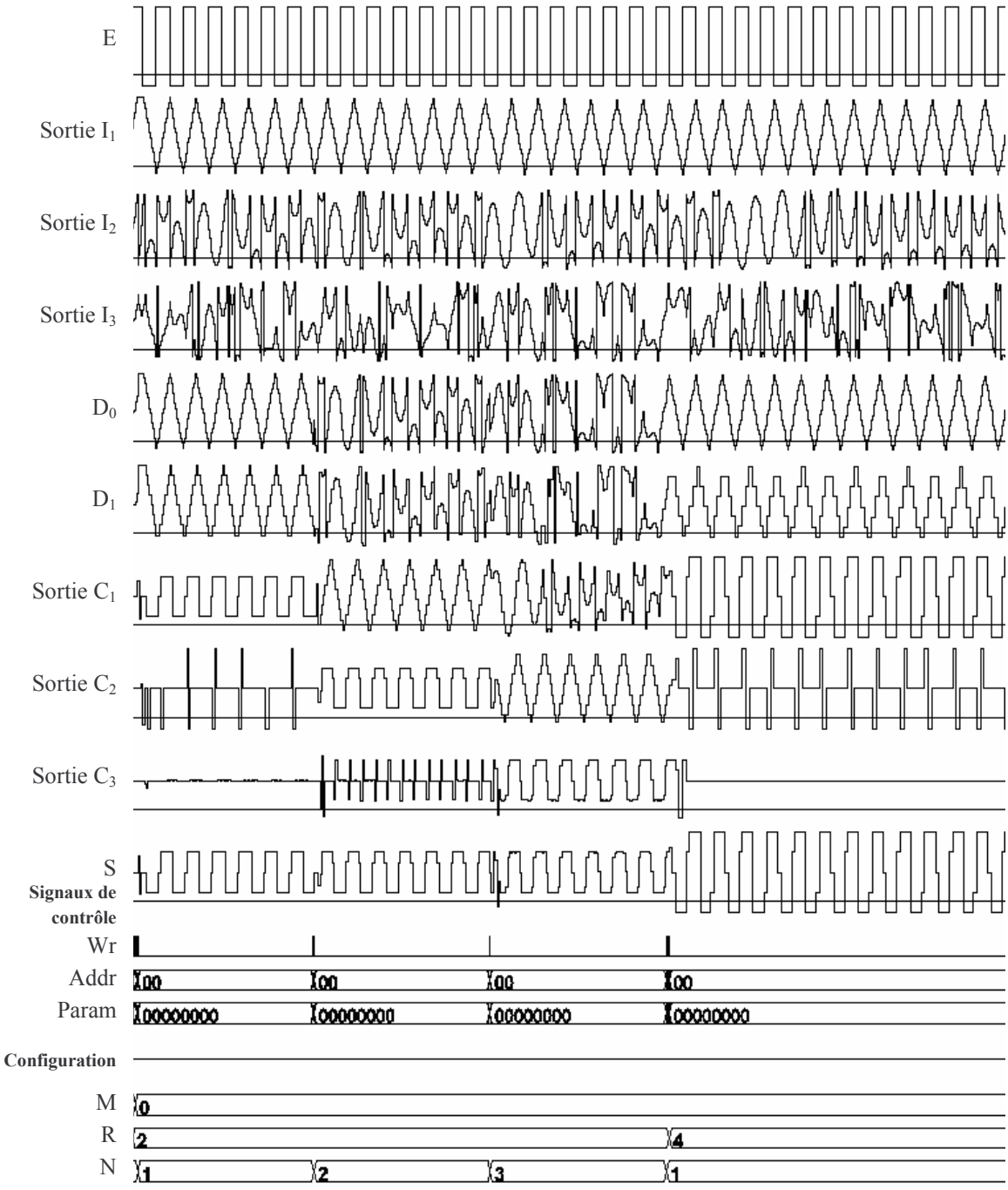


Figure D.4. Réponse d'un filtre CIC à un signal carré et paramétrisation.

La Figure D.4 indique quatre configurations différentes.

La première n'utilise que les blocs I_1, C_1 et le décimateur dont le facteur est de 2. Dans ce cas, le signal est intégré, décimé puis dérivé avant d'être présenté sur S.
La deuxième configuration utilise I_1, I_2, C_1 et C_2. La sortie de I_2 est le résultat de deux intégrations. A partir du signal carré d'entrée, elle représente donc une succession d'arcs de paraboles additionnés à une fonction de droite. Cela suppose que le résultat du premier étage contient une valeur constante.
La troisième configuration utilise tous les étages.

Entre ces trois cas, l'influence du nombre d'étages sur le filtrage de la sortie S est sensible.

La dernière configuration correspond à un CIC d'ordre 1 avec un facteur de décimation de 4.

FIR polynomial Décimateur
Le filtre FIR polynomial décimateur répartit 23 coefficients sur 4 branches.
La Figure D.5 présente le filtrage d'un signal carré par ce système. Les résultats de chaque branche de filtres sont indiqués.
La première configuration utilise les quatre branches alors que la deuxième n'en utilise que deux. Il en résulte un changement de fréquence de fonctionnement du démultiplexeur et des modules de filtrage.

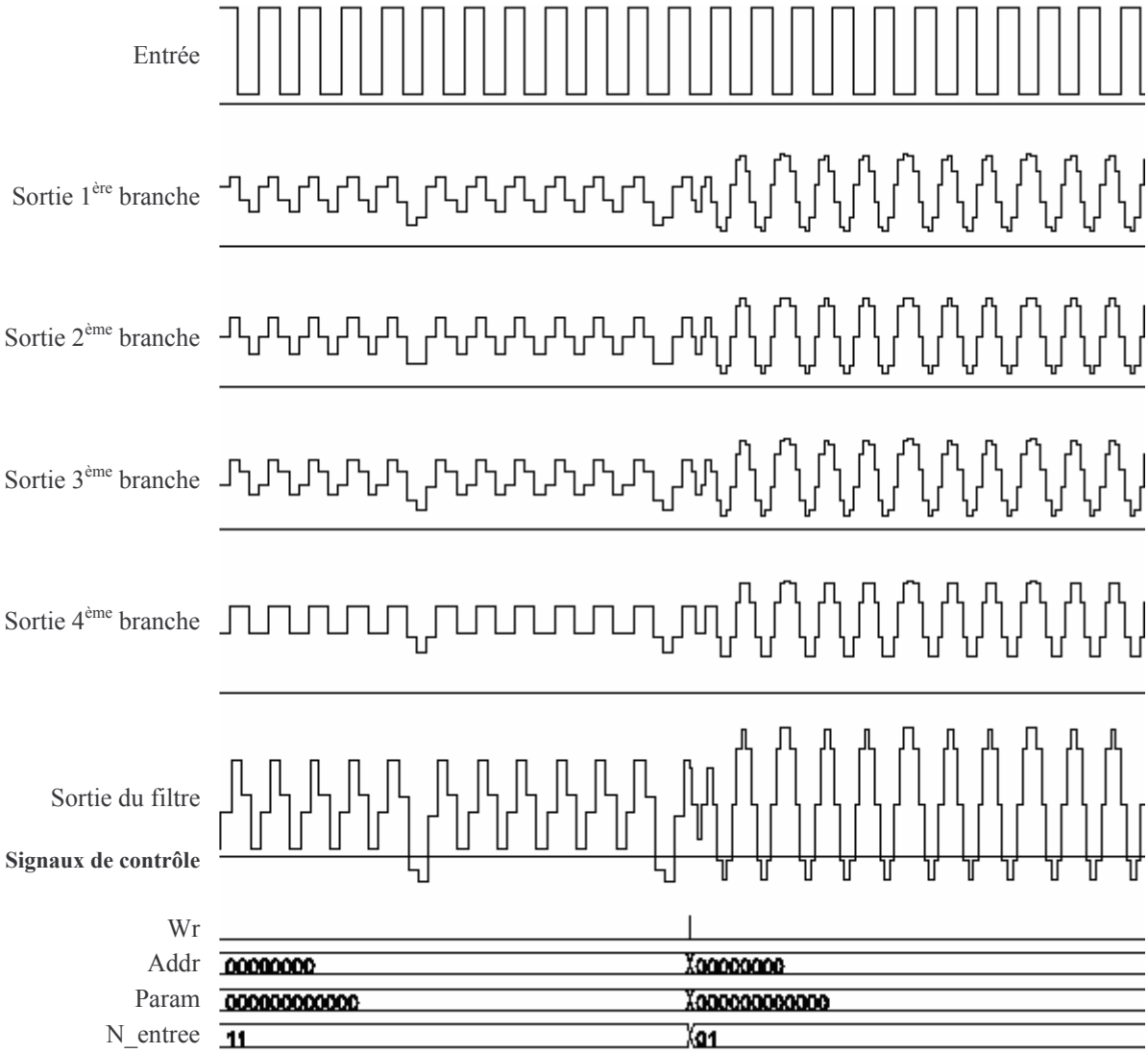


Figure D.5. Simulation du filtre RIF polyphase : réponse à un signal carré et paramétrisation.

La différence de fréquence entre les deux configurations est visible sur la Figure D.6.

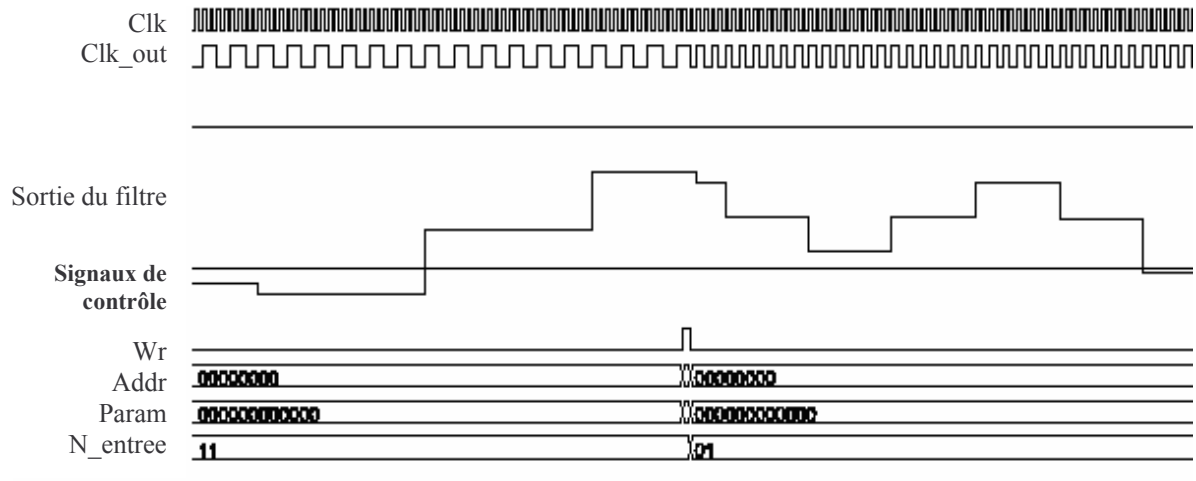


Figure D.6. Fréquence des traitements : zoom autour de l'instant de la paramétrisation.

Filtre de Farrow modifié

Le filtre de Farrow modifié simulé comporte 4 branches comptant 11 coefficients. Ceux-ci sont listés dans le Tableau D.3.

M \ N	0	1	2	3
-5	0,0172	0,0217	0,0065	0,0028
-4	0,1051	0,0717	0,0151	0,0003
-3	0,3099	0,1317	0,0125	-0,0015
-2	0,6079	0,1564	-0,0022	-0,0032
-1	0,8856	0,1084	-0,0211	-0,0027
0	1,0000	0,0000	-0,0298	0,0000
1	0,8856	-0,1084	-0,0211	0,0027
2	0,6079	-0,1564	-0,0022	0,0032
3	0,3099	-0,1317	0,0125	0,0015
4	0,1051	-0,0717	0,0151	-0,0003
5	0,0172	-0,0217	0,0065	-0,0028

Tableau D.3. Coefficients du filtre de Farrow modifié synthétisé.

La Figure D.7 illustre la réponse de ce filtre à un signal carré. Elle indique également la réponse en sortie de chaque branche polynomiale.
La sortie avant décimation correspond au résultat issu de la chaîne propageant la valeur μ. Elle rend compte de l'effet d'interpolation du système.

La simulation propose deux configurations. Elles différent par leur facteur de CFE, les coefficients de filtrage et le nombre d'étages utilisés. Dans la deuxième configuration, seules deux branches sont actives.

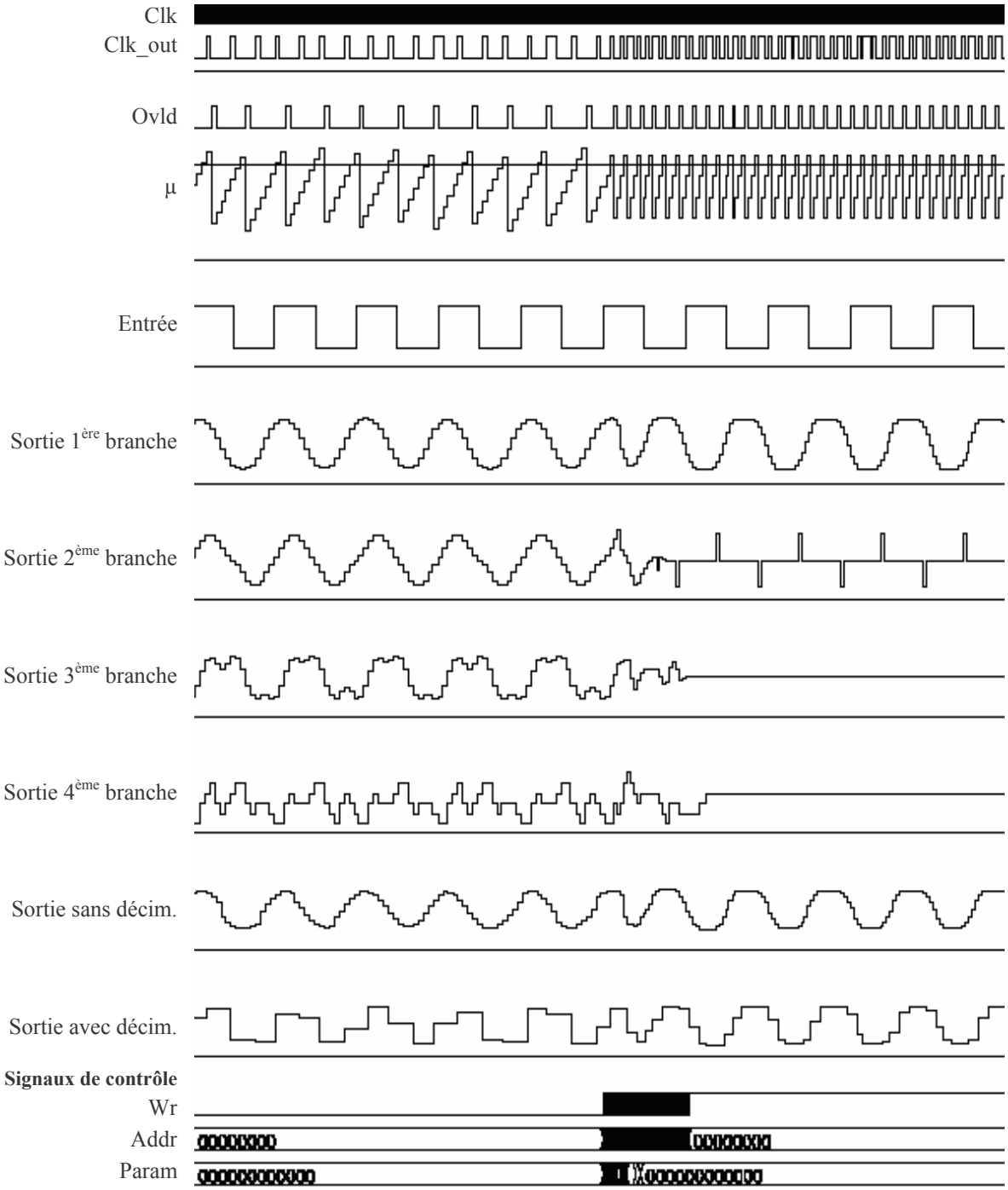


Figure D.7. Simulation du filtre de Farrow : réponse à un signal carré et paramétrisation.

Echelles :

- Entrée : 1 (référence)
- Sortie 1e branche : x 2
- Sortie 2e branche : x 10
- Sortie 3e branche : x 100
- Sortie 4e branche : x 1000
- Sortie avant décimation : x 6
- Sortie du filtre Farrow : x 6

Acronymes et abréviations

μP	Microprocesseur
ΣΔ	Sigma-Delta
AMS	Analog and Mixed Signals
ASIC	Application Specific Integrated Circuit
BB	Bande de Base
BBD	Block Based Design
BW	BandWidth
CAN	Convertisseur Analogique Numérique
CFE	Conversion de Fréquence d'Echantillonnage
CIC	Cascaded Integrator Comb
CLB	Configurable Logic Block
CNA	Convertisseur Numérique Analogique
CORDIC	Coordinate Rotation Digital Computation
DC	Direct Current
DCM	Digital Clock Manager
DRC	Design Rule Check
DSP	Digital Signal Processor
ENOB	Effective Number Of Bits
FFT	Fast Fourier Transform
FI	Fréquence Intermédiaire
FIFO	First In First Out
FoM	Figure of Merit
FPAA	Field Programmable Analog Array
FPGA	Field Programmable Gate Array
FS	Full Scale
GSM	Global System for Mobile communication
GSPS	Giga Samples Per Second
IP	Intellectual Property
kSPS	kilo-Samples Per Second
LMS	Least Mean Square
LNA	Low Noise Amplifier
LUT	Look-Up Table
LVS	Layout Versus Schematic

MAC	Multiply Acumulator
MDAC	Multiplying Digital-to-Analog Converter
MEMS	Micro-ElectroMechanical System
MSPS	Mega Samples Per Second
NCO	Numerically Controlled Oscillator
NLD	Non Linéarité Différentielle
NLI	Non Linéarité Intégrale
NoC	Network-on-Chip
OL	Oscillateur Libre / Oscillateur Local
PBD	Platform Based Design
PCI	Peripheral Component Interconnect
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RF	Radio Fréquence
RIF	Réponse Impulsionnelle Finie
ROM	Read Only Memory
RII	Réponse Impulsionnelle Infinie
RTL	Register Transfer Level
SAR	Successive Approximation Register
SFDR	Spurious Free Dynamic Range
SNDR SINAD SND	Signal to Noise and Distorsion Ratio
SNR	Signal to Noise Ratio
SoC	System-on-Chip
SoPC	System-on-Programmable-Chip
STA	Static Timing Analysis
TRM	Tête de Réception Mixte
TRN	Tête de Réception Numérique
UART	Universal Asynchronous Receiver Transmitter
UMTS	Universal Mobile Telecommunications System
VC	Virtual Component
VGA	Variable Gain Amplifier
VHDL	Very high speed integrated circuit Hardware Description Language
VHDL-AMS	VHDL for Analog and Mixed Signals

Liste des tableaux

Liste des figures

Bibliographie

[Abo04] H. Aboushady, L. de Lamarre, N. Beilleau M.-M. Louërat, « Automatic Synthesis and Simulation of Continuous-Time Sigma-Delta Modulators », Design Automation and Test in Europe, DATE 2004, 16-20 février 2004, pp. 674-675, Paris.

[Aco99] A.J. Acosta, E.J. Peralias, A. Rueda, J.L. Huertas, « VHDL Behavioural Modelling of Pipeline ADC », 4th Workshop on ACD Modeling and Testing (IWADC'99), septembre 1999, pp. 35-39, Bordeaux.

[Aco00] J. Acosta, E. J. Peralias, A. Rueda, J. L. Huertas, « A VHDL-based Methodology for the Design and Verification of Pipeline A/D Converters », Design, Automation and Test in Europe, DATE 2000, pp. 534-538, Paris.

[All03] E. Allier, « Interface Analogique Numérique Asynchrone : une Nouvelle Classe de Convertisseurs Basés sur la Quantification du Temps », thèse de doctorat soutenue le 27 novembre 2003 à l'Institut National Polytechnique de Grenoble (INPG).

[Alt] Site Internet de la compagnie Altera, www.altera.com

[An1] Site Internet de la compagnie Anadigm, www.anadigm.com

[An2] Site Internet de la compagnie Analog Devices, www.analog.com

[Arp02] P. Arpaia, P. Daponte, S. Rapuano, « A State-of-the-Art on ADC Modelling », invited paper to 4th IEE ADDA-7th IMEKO TC-4 Wokshop on Modelling and Testing, Prague, République Tchèque, juin 2002, pp.151-156. Computer Standards and Interfaces, vol.26, 2003, pp.31-42.

[Bab01a] D. Babic, M. Renfors, « Sampling Rate Conversion for Arbitrary Ratio Using Transposed Linear Interpoation and CIC Filter » 9th Telecommunications Forum, TELFOR 2001, 20–22 novembre 2001, Belgrade, Yougoslavie.

[Bab01b] D. Babic, J. Vessma, M. Renfors, « Decimation by Irrational Factor Using CIC-Filter and Linear Interpolation », ICASSP 2001, mai 2001, Salt Lake City, Utah.

[Bab02] D. Babic, J. Vesma, T. Saramäki, M. Renfors, « Polynomial-Based Interpolation Filters for DSP Applications : Design, Implementation, and Application », Receiver Architectures and Signal Processing, septembre 2002, Tampere University of Technology, Finlande.

[Bac96] D. Baccigulapi, M. D'Apuzzo, « Analog-to-Digital Converter Modelling : a Survey », IEEE Measurement, vol. 19, no.3/4, pp. 139-146, 1996

[Bar] Site Internet de la compagnie Barcelona Design, www.barcelonadesign.com

[Bar04a] L. Barrandon, W. Gouret, S. Crand et D. Houzet, « Développement d'une Plateforme PCI Mixte Analogique-Numérique Reconfigurable », 4èmes journées Optiques et Traitement de l'Information – Optique 2004, 15-16 avril 2004, Fès, Maroc.

[Bar04b] L. Barrandon, S. Crand et D. Houzet, « Exploration Architecturale Appliquée au Front-End Mixte d'un Récepteur pour la Radio Logicielle Restreinte », Journées Nationales de Réseau Doctoral de Microélectronique – JNRDM 2004, 4-6 mai 2004, Marseille.

[Bar04c] L. Barrandon, S. Crand, D. Houzet, « Behavioral Modeling and Simulation of Mixed Signal Front-End for Software Defined Radio Terminals », International Symposium on Industrial Electronics – ISIE 2004, 5-7 mai 2004, Ajaccio, France.

[Bar05a] L. Barrandon, S. Crand et D. Houzet, « Systematic Figure of Merit Computation for the Design of Pipeline ADC », Design Automation and Test in Europe – DATE 2005, 7-11 mars 2005, pp. 277-278, Munich, Allemagne.

[Bar05b] L. Barrandon, S. Crand et D. Houzet, « Développement et Synthèse d'un Filtre Farrow Entièrement Générique en VHDL, Application à la Radio Logicielle Restreinte », Journées Nationales de Réseau Doctoral de Microélectronique, JNRDM 2005, 10-12 mai 2005, Paris.

[Bar05c] L. Barrandon, S. Crand, D. Houzet, « Outils d'Aide à la Conception de Systèmes Mixtes Analogiques/Numériques Dédiés à la Radio Logicielle », 3ème MAnifestaction des JEunes Chercheurs en Sciences et Technologies de l'Information et de la Communication - MajecSTIC 2005, 16-18 novembre 2005, pp. 372-375, Rennes.

[Bat02] R. D. Batten, A. Eshraghi, T. S. Fiez, « Calibration of Parallel ΔΣ ADCs », IEEE transactions on Circuits ans Systems-II : Analog and Digital Signal Processing, Vol. 49, No. 6, juin 2002, pp. 390-399.

[Bei03] N. Beilleau, H. Aboushady and M.-M. Louërat, « Systematic Appoach for Scaling Coefficients of Discrete-Time and Continuous-Time Sigma-Delta Modulators » MWSCAS'03, décembre 2003, Le Caire, Egypte.

[Ben04]] P. Benoît, « Architectures des Accélérateurs des Traitements Flexibles pour les Systèmes sur Puce », thèse de doctorat soutenue le 11 octobre 2004, Université de Montpellier II.

[Bil02] T. Bilau, M. Majnik, T. Megyeri, A. Sárhegyi, J. Márkus, I. Kollár « Four-Parameter Fitting of Sine Wave Testing Results – Iteration and Convergence », 4th International Conference on Advanced A/D and D/A Conversion Techniques and their Applications & 7th European Workshop on ADC Modelling and Testing, 26-28 juin 2002, Czech Technical University in Prague, République Tchèque.

[Boe03] G. Boeck, D. Pienkowski, R. Circa, M. Otte, B. Heyne, P. Rykaczewski, R. Wittmann, R. Kakerow, « RF Front-End Technology for Reconfigurable Mobile Systems », IEEE International Microwave and Optoelectronics Conference (IMOC'03), septembre 2003, pp 863-868, Foz do Iguaçu, Brésil.

[Bos02] L. Bossuet, G. Gogniat, J. P. Diguet, J. L. Philippe, « A Modeling Method for Reconfigurable Architectures », International Workshop on SoC for Real-Time Applications, 6–7 juillet 2002, Banff, Canada.

[Bro97] T. L. Brooks, D. H. Robertson, D. F. Kelly, A. Del Muro, S. W. Harston, « A Cascaded Sigma-Delta Pipeline A/D Converter with 1.25MHz Signal Bandwidth and 89 dB SNR », IEEE Journal of Solid-State Circuit, Vol. 32, No. 12, 12 décembre 1997, pp.1896-1906.

[Car00] P. Carbone, D. Petri, « Effective Frequency-Domain ADC Testing », IEEE transactions on circuits and systems II – analog and digital signal processing, vol. 47, no. 7, juillet 2000.

[Cha99] H. Chang et al, « Surviving the SOC Revolution : a Guide to Platform-Based Design », Kluwer, 1999, ISBN 0-7923-8679-5.

[Chi05] Y.-T. Chien, J.-H. Lou, D. Chen, G.-K. Ma, R. Rutenbar, and T. Mukherjee, « Designer-Driven Topology Optimization for Pipelined Analog to Digital Converters » Design Automation and Test in Europe – DATE 2005, 7-11 mars 2005, pp. 279-280, Munich, Allemagne.

[Cli95] D. W. Cline, « Noise, Speed, and Power Trade-offs in Pipeline Analog to Digital Converters », thèse de doctorat soutenue en novembre 1995 à l'Université de Berkley, USA.

[Col93] W. T. Colleran, A. A. Abidi, « A 10-b, 75-MHz Two-Stage Pipelined Bipolar A/D Converter », IEEE Solid State Circuits, Vol. 23, pp. 1334-1344, dec. 1993.

[Col03] E. Colin, « Architecture Reconfigurable pour la Numérisation du Signal Radio de Récepteurs Mobiles Multistandards », thèse de doctorat soutenue en février 2003 à l'Ecole Nationale Supérieure des Télécommunications (ENST) de Paris.

[Con01] G. Constantidines, P. Cheung, W. Luk, « Heuristic Datapath Allocation for Multiple Wordlength Systems », Design Automation and Test in Europe – DATE 2001, 13-16 mars 2001, pp. 791-796, Munich, Allemagne.

[Cra04] S. Crand, L. Barrandon, C. Guillemot, C. Ménard et D. Levallois, « Mise en Œuvre et Modélisation d'une Chaîne d'Acquisition sur une Carte de Prototypage Mixte Analogique – Numérique », 8èmes journées pédagogiques de la Coordination Nationale pour la Formation en Microélectronique (CNFM), 1-3 décembre 2004, Saint-Malo.

[Cru03] A. Cruz Serra, F. Alegria, R. Martins and M. Fonseca da Silva, « Analog-to-Digital Converter Testing – New Proposals », Computer Standards & Interfaces, Volume 26, Issue 1, janvier 2004, pp. 3-13.

[Cum99] M. Cummings, S. Haruyama « FPGA In The Software Radio », IEEE Communications Magazine, février 1999.

[Dec02] C. Decroze, « Etude et Optimisation d'un Nouveau Type d'Antenne Coplanaire. Application à des Liaisons de Proximité et Utilisation pour des Dispositifs Multifonctions et Large Bande », thèse de doctorat soutenue à l'Université de Limoges le 16 décembre 2002.

[Del05] J. P. Delahaye, J. Palicot, P. Leray, « Un Modèle Hiérarchique pour les Architectures Reconfigurables en Radio Logicielle », colloque GRETSI sur le traitement du signal et des images, 6–9 septembre 2005, pp. 803–806, Louvain-la-Neuve, Belgique.

[Des] Site Internet de la compagnie Design and Reuse, www.design-reuse.com

[Des01] M. Dessouky, « Conception en Vue de la Réutilisation de Circuits Analogiques. Application : Modulateur Delta-Sigma à Très Faible Tension », thèse de doctorat soutenue le 18 janvier 2001 à l'Université de Paris VI.

[Don] M. P. Donadio, « CIC filter introduction ». www.dspguru.com/info/tutor/cic.htm

[Don97] S. Donnay, G. Gielen, W. Sansen « High-Level Analog/Digital Partitioning in Low-Power Signal Processing Applications », proceedings of the 7th international workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 8-10 septembre 1997, pp. 47-56, Louvain-la-Neuve, Belgique.

[Ell05] D. Elléouet, N. Julien, O. Déforges, J. G. Cousin, D. Houzet, « Méthodologie de Modélisation à Haut-niveau de la Consommation des Applications du T.D.S.I sur FPGA », Journées Francophones Adéquation Algorithme Architecture (JFAAA 2005), 18–21 janvier 2005, Dijon.

[ETSI] Site Internet de l'European Telecommunications Standards Institute, www.etsi.org

[Far88] C. W. Allemagne, « A Continuously Variable Digital Delay Element », Proc. IEEE International Symposium On Circuits and Systems, vol. 3, pp. 2641-2645, 7-9 juin 1988, Espoo, Finlande.

[Fon01] M. Fonseca da Silva, A. Cruz Serra, « Improving Convergence Of Sine Fitting Algorithms », proceedings of the 6th Euro Workshop on ADC Modelling and Testing, 13-14 septembre 2001, pp. 121-124, Lisbonne, Portugal.

[Gar00] S. Garcia Sabiro, « Mixed-Mode System Design : VHDL-AMS », Microelectronic Engineering N° 54, 200, pp.171-180 (Elsevier).

[Gin02] J. Gines, E. Peralias, A. Rueda, N. Martinez Madrid, R. Seepold, « A Mixed-Signal Design Reuse Methodology Based on Parametric Behavioural Models with Non-Ideal Effects », Conference on Design Automation and Test in Europe (DATE 2002), 4-8 mars 2002, pp. 310-315, Paris.

[Gul01] K. Gulati, H.-S. Lee, « A Low Power Reconfigurable Analog-to-Digital Converter », IEEE journal of Solid State and Circuit, vol. 36, n°. 12, décembre 2001.

[Hen99a] T. Hentschel, G. Fettweis, « Sample Rate Conversion for Software Radio », in ACTS Mobile Communication Summit (AMOS), juin 1999, pp. 733-738, Sorrento, Allemagne.

[Hen99b] T. Hentshel, M. Henker, G. Fettweis, « The Digital Front-End of Software Radio Terminals », IEEE Personal Communications, août 1999.

[Hen00a] T. Hentschel, G. Fettweis, « Sample Rate Conversion for Software Radio », IEEE Communications Magazine, août 2000.

[Hen00b] T. Hentschel, G. Fettweis « Continuous-Time Digital Filters for Sample-Rate Conversion in Reconfigurable Radio Terminals » European Wireless 2000, 12-14 septembre 2000, pp. 55-59, Dresde, Allemagne.

[Hen00c] M. Henker, G. Fettweis « Combined Filter for Sample Rate Conversion, Matched Filtering, and Symbol Synchronization in Software Radio Terminals » European Wireless 2000, 12-14 Septembre 2000, pp. 61-66, Dresde, Allemagne.

[Hen02] Matthias Henker, Gerhard Fettweis « Extended Algorithms for Sample Rate Conversion » Proceedings of the 2nd Karlsruhe Workshop on Software Radios, 20-21 mars 2002, pp. 33-40, Karlsruhe, Allemagne.

[Hers02] M. Hershenson, D. Colleran, A. Hassibi, N. Nandra « Synthesizable Full Custom Mixed-Signal IP », 9ème IEEE/DATC Electronic Design Process (EPD 2002) Workshop 21- 23 avril 2002, Monterey, Canada.

[Herv02] Y. Hervé, « VHDL-AMS – Applications et Enjeux Industriels », Dunod, ISBN2-10-005888-6, édition 2002.

[Hog81] E.B. Hogenauer, « An Economical Class of Digital Filters for Decimation and Interpolation », IEEE Transactions Acoustics, Speech and Signal Processing, Vol. 29(2) (1981), pp. 155-162.

[Hou02] D. Houzet, S. Ouadjaout, J.-G. Cousin, S. Crand, F. Nouvel « Proposition d'une Plateforme de Prototypage pour Systèmes sur Puce », Journée Francophone sur l'Adéquation Algorithme Architecture (JFAAA), 16-18 décembre 2002, Monastir, Tunisie.

[Hou05] D. Houzet et L. Barrandon, « Conception de Circuits en VHDL – Principes et Méthodologie », chapitre VHDL-AMS : Modélisation de Systèmes Analogiques et Mixtes, ISBN 2.85428.527.1, 2ème édition, Editions Cépaduès – à paraître.

[Hun] Site Internet de la compagnie Hunt Engineering, www.hunteng.co.uk

[Int] Extrait du site Internet de la compagnie Intel, www.intel.com/technology/silicon/mooreslaw

[Jin00] H. Jin, E. K. F. Lee, « A Digital-Background Calibration Technique for Minimizing Timing-Error Effects in Time-Interleaved ADCs », IEEE transactions on Circuits and Systems-II – Analog and Digital Signal Processing, vol. 47, n° 7, juillet 2000, pp. 603-613.

[Kar03] K. Karnofsky, « Signal-Processing System Design Tackles Tough Wireless Apps », EE Times, août 2003.

[Kob99] H. Kobayashi, M. Morimura, K. Kobayashi, Y. Onaya « Aperture Jitter Effects in Wideband ADC Systems », SICE'99, 28-30 juillet 1999, Morioka, Japon.

[Kou01a] A. Kountouris, C. Moy, L. Rambaud, P. Le Corre, « A Reconfigurable Radio Case Study : a Software-Based Multi-Standard Transceiver for UMTS, GSM, EDGE and BlueTooth », in Proceedings of the IEEE Vehicular Technology Conference, 2001, octobre 2001, Atlantic City, USA.

[Kou01b] A. Kountouris, C. Moy, L. Rambaud, P. Le Corre, « Full digital IF UMTS Transceiver for Future Software Radio Systems », international conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'01), 25-28 juin 2001, Las Vegas, USA.

[Kou02] A. Kountouris, C. Moy, « Reconfiguration in Software Radio Systems », Workshop on Software Radios, mars 2002, pp. 119-124, Karlsruhe, Allemagne.

[Kur01] N. Kurosawa, H. Kobayashi, K. Maruyama, H. Sugawara, K. Kobayashi, « Explicit Analisys of Channel Mismatch Effects in Time-Interleaved ADC Systems », IEEE Transactions on Circuits and Systems-I : Fundamental Theory and Applications, vol. 48, n° 3, mars 2001, pp. 261-271.

[Leg04] L. Le Garrec, « Etude et Conception en Bande Millimétrique d'Antennes Reconfigurables Basées sur la Technologie des MEMS », thèse de doctorat soutenue à l'Université de Rennes 1 le 15 novembre 2004.

[Len03] S. Le Nours, « Etude, Optimisation et Implantation de Systèmes MC-CDMA sur Architectures Hétérogènes », thèse de doctorat soutenue à l'Institut National des Sciences Appliquées (INSA) de Rennes le 15 décembre 2003, N° D03-23.

[Lil03] H. Lilien « Une brève histoire de l'électronique », Vuibert, 2e édition, 2003.

[Lin] Site Internet de la compagnie Linear Technology, www.linear.com

[Liu02] H. Liu, M. Hassoun, « A 9-b 40-Msample/s Reconfigurable Pipeline Analog-to-Digital Converter » IEEE transactions on Circuits and Systems II – analog and digital signal processing, vol. 49, n° 7, juillet 2002.

[Lop02] F. Lopez, J. Vesma, M. Renfors, « Defining the Wordlength of the Fractional Interval in Interpolation Filters », 9th European Signal Processing Conference (EUSIPCO 2002), 3-6 septembre 2002, vol. 1, pp. 679-682, Toulouse.

[Lot03] R. Lotfi, M. Taherzadeh-Sani, M. Yaser Azizi, O. Shoaei, « Low-Power Design Techniques for Low-Voltage Fast-Settling Operational Amplifiers in Switched-Capacitor Applications », VLSI Journal n°36, 2003, pp. 175-189.

[Lou02] P. Loumeau, J. F. Naviner, H. Petit, L. Naviner, P. Desgreys, « Analog to Digital Conversion : Technical Aspects » Annales des Télécommunications, Tome 57, n° 5-6, mai-juin 2002.

[Mad01] N. Madrid, E. Peralías, A. Acosta, and A. Rueda « Analog/Mixed-Signal IP Modeling for Design Reuse », Design Automation and Test in Europe – DATE 2001, pp. 766-767, 13-16 mars 2001, Munich, Allemagne.

[Mal01] F. Maloberti, « High-Speed Data Converters for Communication Systems », IEEE Circuits and Systems magazine, vol. 1, n° 1, pp. 26-36, 2001.

[Mar] J. Màrkus, « ADC Test Data Evaluation Program for Matlab »

www.mit.bme.hu/services/ieee/ADC-test

[Mar01] J. Màrkus, I. Kollàr, « A MATLAB Tool to Execute IEEE-ETD 1241 », proc. IEEE Instrumentation and Measurments Technology Conference, IMTC/2001, 21-23 mai 2001, pp.1847-1852, Budapest, Hongrie.

[Mat] Site Internet de la compagnie The MathWorks, www.mathworks.com

[Max] Site Internet des companies Maxim et Dallas Semiconductor, www.maxim-ic.com

[Mcc98] J. McCloskey, « Application of VHDL to Software Radio Technology », Verilog HDL Conference and VHDL International Users Forum, 16-19 Mars 1998, pp. 90-95, Santa Clara, USA.

[Men] Site Internet de la compagnie Mentor Graphics, www.mentor.com

[Men02] D. Ménard, O. Sentieys, « Automatic Evaluation of the Accuracy of Fixed Point Algorithm », Design Automation and Test in Europe – DATE 2002, 4-8 mars 2005, pp. 529-535, Paris.

[Mit02] J. Mitola et al, « Software Defined Radio, Enabling Technologies », Tuttlebee, 2002.

[Moe03] K. Moessner, D. Bourse, D. Greifendorf, J. Stammen « Software Radio and Reconfiguration Management », Computer Communications 26 (2003), pp. 26–35.

[Nav04] L. A. de B. Naviner, J. F. Naviner, M. Alves de Barros, « Programmable Digital Channel Selector for Multimode Radio Receiver », IEEE Symposium on Signal Processing and Information Technology (ISSPIT), décembre 2004, Rome, Italie.

[Nun98] A. Nuñez-Aldana, A. Doboli, R. Vemuri, « A Top-Down Synthesis Methodology for Behavioral Mixed-Signal Systems Specified in VHDL-AMS », Proceedings of the Second International Workshop on Design of Mixed-Mode Integrated Circuits and Applications, 1998, Guanajuato, Mexique.

[Nuz03] P. Nuzzo, « Architectural Space Characterization of a Transconductance Amplifier », thèse de doctorat soutenue à l'Université de Pise, 2003.

[Oll04] G. Ollivier, « L'industrie de la Microélectronique, les Enjeux de la Microélectronique », 8èmes journées pédagogiques de la Coordination Nationale pour la Formation en Microélectronique, CNFM, 1-3 décembre 2004, Saint-Malo.

[Oua02] S. Ouadjaout, D. Houzet, « VCI Interface Methodology in Rapid Prototyping Platform with PCI NoC », WSEAS IEHS2002 Conference, juin 2002, Cadix, Espagne.

[Pal01] J. Palicot, « La Radio Logicielle est Incontournable dans l'Evolution des Radiocoms », Electronique Internationale Hebdo, 8 novembre 2001, N°452.

[Pal03] J. Palicot, C. Roland, « FFT : a Basic Function for a Reconfigurable Receiver », International Conference on Telecommunication (ICT 2003), février 2003, pp. 898–902, Papeete, Tahiti.

[Pan99] H. Pan, « A 3.3-V, 12-bit, 50-MS/s A/D Converter in 0.6-µm CMOS », thèse de doctorat soutenue à l'Université de Californie, 1999, Los Angeles, USA.

IEEE Jounal of Solid-State Circuits, Vol. 35, No. 12, Décembre 2000.

[Pan04] P. Panaia, C. Luxey, G. Jacquemod, R. Staraj, G. Kossiavas, L. Dussopt, F. Vacherand, C. Billard « MEMS-based Reconfigurable Antennas », International Symposium on Industrial Electronics - ISIE 2004, pp. 175-180, 5-7 mai 2004, Ajaccio.

[Pci] Site Internet du Peripheral Component Interconnect Special Interest Group, www.pcisig.com

[Per95] E. J. Peralias, A. Rueda, J. L. Huertas, « Statistical Behavioral Modeling and Characterization of A/D Converters », International Conference on Computer-Aided Design, ICCAD-95, 5-9 novembre 1995, pp. 562-566.

[Pol02] J.-F. Pollet « Mixed-Signal Virtual Components : The Challenges », IMSTW - 8th IEEE International Mixed-Signal Testing Workshop, Panel : IP demands for Mixed-Signal testing, 18-21 juin 2002, Montreux, Suisse.

[Pro96] J. G. Proakis, D. G. Manolakis « Digital Signal Processing – Principles, Algorithms and Applications », Macmillan (1996), ISBN 0133942899.

[Rah02] T. Rahkonen « Error Correction Techniques in High-Speed A/D and D/A Converters », Error correction grad. Course, janvier 2002, Université de Oulu, Electronics Laboratory, Finlande, www.ee.oulu.fi

[Rob04] M. Robert, « Défis en CAO de Circuits et Systèmes Integrés : Recherche, Formations, Métiers », 8èmes journées pédagogiques de la Coordination Nationale pour la Formation en Microélectronique, CNFM, 1-3 décembre 2004, Saint-Malo.

[Rou04] C. Rougier, J.-B. Begueret, H. Lapuyade, Y. Deval, A. Malvasi, « Multi-Functional RF Frequency Synthesizer for Multi-Standard Smart Objects » International Symposium on Industrial Electronics - ISIE 2004, pp. 163-168, 5-7 mai 2004, Ajaccio.

[Ruo05] T. Ruotsalainen, R. Holden, V. Loukusa, J. Kivari, « Top Down Mixed Signal Design using Matlab and VHDL/VHDL-AMS », Design Automation and Test in Europe - DATE 2005, Focus on Business and Industry, 7-11 mars 2005, Munich, Allemagne.

[Set99] P. Setty, J. Sonntag, « Reconfigurable Analog-to-Digital Converter. », brevet US patent No. 5.877.720, 2 mars 1999.

[Sit] Site Internet du Syndicat des Industries de Tubes ELEctroniques et Semi Conducteurs, www.sitelesc.fr

[Std99] IEEE Standard 1076.1-1999, « IEEE Standards VHDL Analog and Mixed-Signal Extensions », 18 mars 1999, ISBN 0-7381-1640-8.

[Std01] IEEE TC-10 « Standard for Terminology and Test Methods for Analog-to-Digital Converters », IEEE standard 1241-2001.

[Sum02a] L. Sumanen, « Dual-Mode Pipeline A/D Converter for Direct Conversion Receivers », Electronic Letters, 12 septembre 2002, vol. 38, n° 19.

[Sum02b] L. Sumanen « Pipeline Analog-to-Digital Converters for Wideband Wireless Communications », thèse de doctorat soutenue le 13 décembre 2002, University of Technology, Electronic Circuit Design Laboratory, Report 35, Helsinki, Finlande.

[Sun] Site Internet de la compagnie Sundance, www.sundance.com

[Syn] Site Internet de la compagnie Synopsys, www.synopsys.com

[Tah04] M. Taherzadeh-Sani, R. Lotfi, and O. Shoaei, « An Analytical Approach to the Estimation of Dynamic Non-linearity Parameters in Pipeline A/D Converters », 29th European Solid-State Circuits Conference, ESSCIRC 2004, 20-24 septembre 2004, Leuven, Belgique.

[Tho99] H. Thomas, J. P. Diguet, J. L. Philippe, « Estimation et Métriques au Niveau Système pour la Conception Conjointe Logiciel/Matériel », Colloque GRETSI, septembre 1999, Vannes.

[Ti] Site Internet de la compagnie Texas Instruments, www.ti.com

[Tro04] J.P. Troadec, « Principes de Conversions Analogique-Numérique et Numérique-Analogique. Cours, exercices et problèmes résolus », Dunod, ISBN : 2100074784, 2004.

[Tut02] W. Tuttlebee, « Software Defined Radio Enabling Technologies », John Wiley and Sons Ltd, 2002.

[Van02] J. Vandenbussche, E. Lauwers, K. Uyttenhove, M. Steyaert, G. Gielen « Systematic Design of a 200 Ms/S 8-bit Interpolating A/D Converter » Design Automation and Test in Europe, DATE 2002, 4-8 mars 2002, pp. 357-361, Paris.

[Vau91] R. G. Vaughan, N. L. Scott, D. R. Whit, « The Theory of Bandpass Sampling », IEEE Transactions on Signal Processing, vol. 39, n° 9, septembre 1991, pp. 1973-1984.

[Vil02] G. Villemaud, « Etude d'antennes ruban tridimensionnelles compactes pour liaisons de proximité. Application à des systèmes de télémesure et de localisation de téléphone cellulaire », thèse de doctorat soutenue à l'Université de Limoges le 16 décembre 2002.

[Vog03] M. Vogels, G. Gielen « Figure of Merit Based Selection of A/D Converters », Design and Test Automation in Europe, DATE 2003, 3-7 mars 2003, p.1190-1191, Munich, Allemagne.

[Wep95] J. A. Wepman, « Analog-to-digital converters and their applications in radio receivers », IEEE Communication Magazine, vol. 33, pp. 39-45, mai 1995.

[Xil] Site Internet de la compagnie Xilinx, www.xilinx.com

Résumé

Les technologies électroniques actuelles offrent la possibilité de réaliser des systèmes sur puce (SoC). Les méthodologies doivent s'adapter à ce nouveau cadre de conception afin de tirer profit des capacités que présentent les SoC.

Dans ce contexte, le thème émergent de radio logicielle a pour rôle de répondre à une diversité de fonctionnalités et de standards de télécommunication sans fil à l'aide d'une interface matérielle unique.

L'objet de cette thèse est d'élaborer des méthodologies et des outils contribuant au dimensionnement et à la synthèse systématique d'un module analogique/numérique dédié à des applications radio logicielle.

Les aspects modélisation et prototypage d'une tête de réception mixte constituent les principales contributions de ce travail. Un exemple de dimensionnement d'une tête de réception mixte répondant aux standards GSM et UMTS est développé. La simulation globale de ce système ainsi que son implémentation matérielle sont proposées.

Mots clés

Radiocommunications mobiles, conception de systèmes, convertisseurs analogique/numérique, dispositifs logiques programmables, filtres numériques.

Abstract

Present electronic technologies enable to carry out systems-on-chip (SoC). The design methodologies must take into account this new field of design. In this context, the emergent concept of software radio is targeted to deal with a variety of functionalities and telecommunication standards thanks to a single hardware interface.

The subject of this thesis is to elaborate top-down methodologies and tools contributing to the sizing and to the systematic synthesis of an analog/digital module dedicated to software radio applications.

Modelling and prototyping aspects of a mixed front-end represent the main contributions of this work. An example of the sizing of a mixed front-end corresponding to the GSM and UMTS standards is developed. Global simulation of this system and hardware implementation are proposed.

Keywords

Mobile radiocommunications, system-level design, analog-to-digital converters, programmable digital devices, digital filters.