Design of a generic digital front-end for the internet of things

Ali Zeineddine

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CentraleSupélec

THESE DE DOCTORAT DE

CENTRALESUPELEC
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Automatique, Productique et Robotique – Signal, Image, Vision – Télécommunications

Par

« Ali ZEINEDDINE »

« Conception d'un Front-End Numérique Générique pour l'Internet des Objets »

« Design of a Generic Digital Front-End for the Internet of Things »

Thèse présentée et soutenue à « Rennes », le « 06 Novembre 2019 »
Unité de recherche : IETR - SCEE
Thèse N° : 2019CSUP0001

Rapporteurs avant soutenance :

Christophe JEGO	Professeur des Universités, IMS Bordeaux, France
Markku RENFORS	Professor of Telecommunications, Tampere University of Technology, Finlande

Composition du Jury :

Présidente	Marie-Laure BOUCHERET	Professeur des Universités, INP-ENSEEIHT / IRIT, France
Rapporteurs	Christophe JEGO	Professeur des Universités, IMS Bordeaux, France
	Markku RENFORS	Professor of Telecommunications, Tampere Uni. of Technology, Finlande
Examinateurs	Dominique MORCHE	Ingénieur de Recherche au CEA-LETI, Grenoble, France
	Marie-Laure BOUCHERET	Professeur des Universités, INP-ENSEEIHT / IRIT, France
Encadrants	Amor NAFKHA	Enseignant-Chercheur à CentraleSupélec, Campus de Rennes, France
	Pierre-Yves JEZEQUEL	Ingénieur Expert Technique et Chef de Projet à TDF, Liffré, France
	Stéphane PAQUELET	Responsable de Laboratoire et Senior Scientist à b-com, Rennes, France
Directeur de thèse	Christophe MOY	Professeur des Universités, Université de Rennes 1 / IETR, France

"Ecoutez, je crois que j'ai fait tout ce qui était en mon pouvoir, j'ai fait le maximum, à ce niveau là où on en est, je crois qu'on peut dire à la grâce de Dieu, et allez..."
- Simon JÉRÉMI

Acknowledgments

This thesis is the result of three years of work financed by TDF, and took place at the premises of b-com. Through the few lines below, I would like to acknowledge, at the beginning of this thesis, the help and support of many people, without whom this thesis and its success would not have been possible. These acknowledgments are not in order of importance and are not limited to the names mentioned.

My deepest gratitude goes first to my supervisor, Christophe Moy, from University of Rennes 1, for his continuous guidance throughout these three years, for his help in theoretical and practical developments, and of course for the efforts he put into revising every article published, including this thesis manuscript. I also appreciate the time he dedicated to manage many administrative procedures all along the duration of this PhD.

I am very grateful to TDF, for proposing and financing this PhD. I would like to thank Pierre-Yves Jezequel for investing his time into supervising my work, and for his valuable insights on many subjects. I also appreciate the help of my administrative supervisor Christian Nieps who always ensured that my PhD is progressing well. I would also like to thank François Picand, the director of the innovation department at TDF, who regularly followed the advancement of my PhD, and showed genuine interest in my work. Last but not least, special thanks go to Amélie Kerbellec who helped me in my integration at TDF and in managing many administrative procedures at work.

During these three years, I worked at b-com in a very productive environment. I am deeply grateful to Stéphane Paquelet, my supervisor at b-com, who initiated many of the ideas developed in this thesis thanks to his deep mathematical and technical expertise in the domain, and provided the basis of most contributions. I am also grateful to Michel Corriou, my research lab director, for ensuring I had the needed resources for performing my work at b-com, and presenting my work in international conferences. I would also like to address special thanks to Bertrand Guilbaud, the CEO of b-com, for following the advancement of my work, and for his cordial connection with the b-com staff.

My most special thanks goes to Amor Nafkha, my supervisor at CentraleSupélec, who worked closely with me to develop many ideas, provided me with many insights, and also helped me on many occasions to manage both professional and personal aspects of my PhD life. I would also like to thank Karine Bernard from CentraleSupélec, who helped me in many administrative procedures. I'm also grateful to Philippe Benabes and Francis Trélin, who gave me access to the ASIC design tools available in their laboratory. I will surely not forget to recognize the efforts of Jacques Palicot, the recently retired director of the SCEE research team, who put constant effort to make sure that the work of all the PhD students in the team is successful.

This work was possible thanks to the "Investissements d'avenir" project initiated by the French government in 2010, and is underway until today. I thank everyone who worked on this project, from governmental, to industrial, to academic entities.

Finally, I would like to address one big *thank you* to my family, friends and colleagues, who were around me during these three years, and who constantly tried to remind me that there is more in life than just work.

Résumé

Le nombre de technologies et de normes de communications sans fil est en augmentation constante afin de fournir des solutions de communication à distance pour les différents besoins technologiques actuels et à venir. Ceci est particulièrement le cas de l'Internet des objets (IoT), où déjà de nombreuses solutions sont disponibles, et de nombreuses autres sont attendues. Pour un déploiement efficace du réseau IoT, l'interopérabilité entre les différentes solutions est essentielle pour éviter la fragmentation de ce réseau, ce qui augmente ses coûts d'installation et d'exploitation et complique sa gestion. L'interopérabilité sur le plan physique est fournie par des modems multistandards prenant en charge le plus grand nombre de normes avec un coût de mise en œuvre minimal. Ces modems sont possibles grâce au front end numérique (DFE), qui offre une interface radio flexible capable de traiter une large gamme de signaux. Cette thèse développe tout d'abord deux architectures génériques de DFE pour la transmission et la réception, pouvant être facilement adaptées aux différentes normes IoT. Ces architectures mettent en évidence le rôle principal du changement de rythme (SRC) dans le DFE et l'importance de l'optimisation de la mise en œuvre de cette fonction. Cette optimisation est ensuite réalisée grâce à une étude approfondie des fonctions SRC, et au développement de nouvelles structures plus efficaces en termes de complexité de mise en œuvre et de consommation, pour des performances égales ou supérieures. La dernière partie de la thèse concerne l'optimisation de la mise en œuvre matérielle du DFE, réalisée par le développement d'une méthode de quantification optimale qui minimise l'utilisation de ressources matérielles tout en garantissant un certain niveau de performance. Les résultats obtenus sont enfin mis en valeurs en comparant différentes stratégies de mise en œuvre sur des cibles FPGA et ASIC.

Abstract

The number of wireless communication technologies and standards is constantly increasing to provide communication solutions for today's technological needs. This is particularly relevant in the domain of the Internet of Things (IoT), where many standards are available, and many others are expected. To efficiently deploy the IoT network, the interoperability between the different solutions is critical in order to avoid the fragmentation of this network, which increases its installation and operation costs, and complicates its management. Interoperability on the physical level is achieved through multi-standard modems that support the largest number of standards with minimal added implementation costs. These modems are made possible through the digital front-end (DFE), that offers a flexible radio front-end able of processing a wide range of signal types. This thesis first develops a generic architecture of both transmission and reception DFEs, which can be easily adapted to support different IoT standards. These architectures highlight the main role of sample rate conversion (SRC) in the DFE, and the importance of optimizing the SRC implementation. This optimization is then achieved through an in-depth study of the SRC functions, and the development of new structures of improved efficiency in terms of implementation complexity and power consumption, while offering equivalent or improved performance. The final part of the thesis addresses the optimization of the DFE hardware implementation, which is achieved through developing an optimal quantization method that minimizes the use of hardware resources while guaranteeing a given performance constraint. The obtained results are finally highlighted through implementing and comparing different implementation strategies on both field programmable gate array (FPGA) and application specific integrated circuit (ASIC) targets.

Résumé Étendu en Français

Contexte et Motivations

Suite aux dernières révolutions technologiques de l'Internet et les réseaux mobiles qui ont rendu possible la communication quasi permanente et directe des hommes à travers la planète, une nouvelle révolution commence à prendre forme qui va permettre à presque tout objet de communiquer. On appelle "Internet des objets" (*Internet of Things IoT*) le concept de réseau reliant ces objets, qui devrait accueillir plus de 75 milliards objets d'ici 2025. Chaque objet connecté répond à une application spécifique qui impose des contraintes particulières et la grande variété d'applications crée une importante diversité de contraintes. Par conséquent, une multitude de technologies et de standards de communication sont proposés, chacune utilisant des techniques de transmission et des caractéristiques de signal spécifiques, pour proposer une solution à un certain cas d'application. Parallèlement, de nouvelles normes sont en cours de développement qui ont l'objectif de prendre en charge le plus grand nombre de cas d'applications. Un exemple est le réseau mobile de quatrième génération (4G) *Long Term Evolution* (LTE), qui définit de nombreuses catégories d'appareils, chacune utilisant des techniques de transmission spécifiques au cas d'application ciblé. La cinquième génération de ce réseau vise à encore plus considérer de cas différents. En outre, les concepts de radio logicielle et de radio intelligente visent la construction des systèmes radio hétérogènes de plus en plus complexes, capables de commuter automatiquement de technologie de communication en fonction de l'environnent radio.

Cette diversité des solutions de communication crée un défi pour les opérateurs de réseau et les fournisseurs de services de télécommunication. C'est le cas à TDF, un des principaux opérateurs français d'infrastructures radio, soucieux du déploiement efficace des solutions IoT récemment proposées et du futur réseau mobile de cinquième génération (5G). Si chacune des solutions de communication est traitée indépendamment des autres, le déploiement du réseau devient fragmenté, ce qui tue l'interopérabilité, complique la gestion de la qualité de service (*Quality of Service QoS*) et augmente considérablement les coûts de déploiement (comprenant l'installation initiale, la maintenance et la consommation). Ces problèmes sont au cœur des systèmes 5G. TDF s'est associé à l'institut de recherche technologique (IRT) b-com pour développer des solutions de déploiement efficaces qui favorisent et améliorent l'interopérabilité dans les futurs réseaux.

Il est alors utile de se référer au modèle usuel de représentation en couches des systèmes de communication, le modèle OSI, pour comprendre sur une description suffisamment générique et abstraite, comment ces questions peuvent être traitées. Nous pourrions nous contenter d'assurer l'interopérabilité et les fonctions transverses sur les couches dites « hautes » (du transport à l'application), mais ce serait négliger les questions de compacité et sobriété énergétique des équipements, critiques aussi bien pour les objets que pour les stations de collectes (*Gateways*). Il faut donc considérer les couches « matérielles », i.e physique (traitements analogiques et numériques), liaison (adresse MAC), paquet (adresse IP). La priorité de ce travail est la couche physique.

S'agissant de la couche physique, les enjeux d'interopérabilité et de transversalité se ramè-

nent à fournir une plateforme modem suffisamment flexible pour accueillir les différentes technologies. En ajoutant les contraintes de compacité et sobriété énergétique, les choix techniques s'orientent naturellement vers une mutualisation des fonctions sur des unités de traitement définitivement câblées (compacité et consommation) mais paramétrables par un jeu de registre permettant le basculement d'un système à l'autre (flexibilité). Il s'agit d'une vision *Software Defined Radio* (SDR) dans laquelle le partage des traitements entre le processeur et la structure matérielle câblée (FPGA/ASIC) est plutôt centré sur cette dernière, qui est chargée des traitements simples (additions/multiplications), systématiques et parallélisables, tandis que le processeur se voit affecté des calculs complexes dont les contraintes de temps d'exécution sont relâchées. Le niveau d'optimalité d'un tel système peut être évalué selon le critère suivant : le degré de flexibilité qui est assurée vis-à-vis d'un certain nombre de technologies considérées, et dans un mode donné, le degré de complexité ou de consommation du modem par rapport à celle qu'aurait une réalisation dédiée pour ce mode.

Objectifs et Contributions

On peut décomposer la partie numérique du modem en trois parties : le front-end numérique (*Digital Front-End DFE*), le traitement en bande de base et la correction d'erreur (*Forward Error Correction FEC*). Dans un modem moderne, le DFE est généralement réalisé en matériel, tandis que les deux autres parties sont souvent réalisées en logiciel. Par "matériel", on désigne ici une mise en œuvre sur cible matérielle numérique tels que les FPGA ou ASIC (conçue en VHDL). Par "logiciel", on considère une mise en œuvre sur un processeur (par programmation en langage informatique). Ce travail vise l'optimisation de la mise en œuvre matérielle du front-end numérique (DFE) afin de réduire les coûts et la consommation de l'interface radio.

Le DFE comporte plusieurs fonctions de traitement du signal, mais il a deux rôles principaux: le filtrage et le changement de la fréquence d'échantillonnage (*Sample Rate Conversion SRC*). La fonction SRC est souvent intégrée dans autres fonctions du DFE, comme le filtrage et la synchronisation. Cela fait du SRC la fonction la plus essentielle du DFE. Les principaux objectifs de la thèse sont alors définis :

1. Concevoir une architecture générique et efficace du DFE qui s'adapte facilement à toute norme ou technologie de communication, en particulier pour l'IoT.
2. Optimiser la fonction SRC dans le DFE en développant des nouvelles solutions, puis mettre en œuvre les structures nouvellement développées et évaluer leurs niveaux de complexité et de consommation.
3. Définir une méthode à suivre pour la mise en œuvre matérielle, à utiliser avec toutes les fonctions du DFE, concernant la quantification et l'architecture du circuit numérique.

Pour parvenir à ces objectifs, cette thèse développe les contributions suivantes.

La première de ces contributions est la proposition d'architectures génériques pour le DFE dans les deux sens d'émission et de réception, ainsi que leur intégration dans le modem en combinaison avec le front-end analogique. Cette contribution est présentée dans le Chapitre 1. Ce chapitre présente d'abord le contexte de l'IoT ainsi que le problème de la diversité des normes et des technologies de communications sans fil. Un modem multistandard est alors proposé comme solution, le DFE étant un élément principal de ce modem. Puis, les fonctions principales du DFE sont présentées, qui sont ensuite utilisées pour construire les architectures génériques du DFE. Ces architectures génériques mettent en évidence que les fonctions SRC, qui sont utilisées à plusieurs endroits dans le DFE, constituent le cœur de ce système et que leur optimisation est cruciale pour une mise en œuvre efficace de l'ensemble du modem.

La deuxième contribution développée dans le Chapitre 2 propose une nouvelle présentation concise des filtres SRC disponibles dans la littérature. Le but de cette présentation est de

rassembler la littérature des filtres SRC sous une seule vision unifiée. Cette présentation est réalisée en regroupant les cinq principales structures de filtre SRC à réponse impulsionnelle finie (FIR) autour d'une racine commune, le filtre FIR à phase linéaire, à partir de laquelle les cinq structures sont ensuite dérivées. En pratique, il existe deux catégories de SRC : l'une réalisant un facteur d'interpolation ou de décimation entier, éventuellement grand, la seconde réalisant une fraction proche de un, aussi précise que l'on veut. Leur combinaison permet systématiquement d'atteindre un changement de rythme arbitraire (ASRC). S'agissant de la mise en œuvre du SRC fin, les choix techniques restent encore très ouverts à l'intérieur du cadre général des filtres à retard variable (V-FDF), qui s'avèrent aujourd'hui les structures les plus appropriées.

Le Chapitre 2 présente également les aspects de la mise en œuvre matérielle pour chaque structure afin de fournir un guide complet et concis sur le SRC, tant du point de vue théorique que pratique. Cette vision unifiée est développée pour les filtres SRC d'interpolation dans le Chapitre 2, le cas complémentaire des filtres SRC de décimation est présenté dans l'Appendice A. La suite du travail étudie en profondeur la structure de Newton, et nous montrons son intérêt pour construire de nouvelles solutions SRC plus efficaces.

Le Chapitre 3 aborde deux aspects théoriques liés à la structure de Newton, et nécessaires pour définir de manière rigoureuse la réponse du filtre, ce qui constitue la troisième contribution. Tout d'abord, la relation directe entre les expressions de l'interpolation de Lagrange et la série de Newton est développée. Cette relation est ensuite utilisée pour démontrer la convergence de la fonction de transfert de la structure de Newton.

La quatrième contribution est la conception de nouveaux filtres SRC basés sur la structure de Newton. Jusqu'en 2009, la structure de Farrow était la solution la plus répandue pour la mise en œuvre des V-FDF [Far88]. Ensuite, la structure de Newton [Tas96] a été proposée afin de tirer profit de sa simplicité et de sa faible complexité [Leh09], mais limitée à un seul gabarit de filtrage, celui de l'interpolation de Lagrange. En 2016, le travail publié par D. Lamb et al. dans [Lam16] a montré que les structures de Farrow et de Newton peuvent être reliées par une transformation matricielle. Cette transformation permet d'exploiter davantage la structure de Newton en la modifiant pour mettre en œuvre différentes réponses V-FDF. C'est cette approche que nous nous proposons d'étendre dans cette contribution. Ceci fournit non seulement des solutions ASRC plus performantes, mais réduit également la complexité matérielle, et donc le coût et la consommation du DFE.

Trois nouvelles modifications sont développées et présentées dans le Chapitre 4. Tout d'abord, nous proposons une modification de la structure de Newton pour l'interpolation de Hermite. Cette interpolation est un cas général d'interpolation de Lagrange, dans lequel les contraintes d'égalité sont imposées non seulement aux valeurs des échantillons, mais également à leurs dérivées. Par conséquent, davantage de degrés de liberté sont disponibles pour ajuster l'interpolation utilisée. En général, la réponse du filtre d'interpolation de Hermite conserve la même bande passante que celle de l'interpolation de Lagrange, tout en améliorant les performances en termes de rejet des images. Parallèlement à cette performance de filtrage améliorée, la mise en œuvre de cette interpolation offre une complexité plus faible vis-à-vis la structure de Newton classique pour l'interpolation de Lagrange.

La transformation de Farrow-Newton permet théoriquement la mise en œuvre de toute interpolation polynomiale par le biais d'une structure de Newton généralisée. Cependant, en s'appuyant sur cette transformation, la relation entre les coefficients de cette structure généralisée et la réponse du filtre n'est pas connue. Pour trouver cette relation, l'expression de la réponse fréquentielle de la structure de Newton généralisée doit être développée en utilisant les propres coefficients de cette structure. Cela fait, il est ensuite possible d'utiliser des méthodes d'optimisation de filtre permettant d'approximer une réponse de filtre idéale souhaitée. Ces méthodes d'optimisation offrent également l'avantage d'imposer des contraintes pour personnaliser la structure optimisée. Par exemple, il est possible d'imposer que seules les lignes de

retour provenant du dernier élément de retard soient possibles. Cela permet de concevoir des structures de Newton optimisées ayant des performances comparables à celles des structures de Farrow optimisées, tout en nécessitant qu'une fraction de complexité de ces dernières.

La structure de Newton classique peut être identifiée dans toutes les nouvelles structures proposées. En écrivant la fonction de transfert sous la forme d'une combinaison linéaire de celle de la structure de Newton classique et de n'importe quelle seconde structure de Newton modifiée, il est possible de trouver des structures qui sont reconfigurables via un seul paramètre variable. Une telle structure est capable de mettre en œuvre les deux interpolations considérées, ainsi qu'une plage d'interpolations combinant les caractéristiques des deux interpolations en faisant varier le paramètre de reconfiguration. De plus, cette structure reconfigurable garde une complexité comparable à celle de la structure de Newton classique, ce qui n'est pas le cas si on rend la structure de Farrow reconfigurable. Cela est dû au fait que le paramètre de reconfiguration introduit ne modifie que certains coefficients de la structure de Newton, et par conséquent seul un nombre limité de multiplieurs doit être ajouté pour permettre la reconfigurabilité. Ces structures reconfigurables offrent une solution très intéressante pour les applications de radio intelligente ou les systèmes multistandards, dans lesquels le système doit s'adapter en permanence au signal reçu et à son environnement.

La suite de ce travail aborde les problèmes d'optimisation de la quantification et de la mise en œuvre matérielle. La cinquième contribution, présentée dans le Chapitre 5, est le développement d'un algorithme simple de quantification optimal, qui trouve le nombre d'éléments binaires minimal pour représenter chaque signal, en garantissant une contrainte de précision donnée. La première section de ce chapitre présente les principes et l'état de l'art des méthodes de quantification disponibles dans la littérature. La deuxième section développe la nouvelle méthode analytique de quantification optimale. La dernière section applique la méthode développée à un exemple d'un DFE simpliste, afin de démontrer l'utilisation pratique de cette méthode.

Le Chapitre 6 développe la dernière contribution, qui est la réalisation matérielle de différents filtres SRC sur cibles FPGA et ASIC. La première partie étudie les contraintes de mise en œuvre et les stratégies utilisées pour développer des architectures matérielles. Cette étude montre l'importance d'utiliser la stratégie de mise en œuvre appropriée en fonction des exigences de l'application. Puis, les architectures de plusieurs structures de SRC sont développées en utilisant les différentes stratégies de mise en œuvre. La complexité et la consommation de ces implémentations sont analysées et comparées pour mettre en évidence les avantages de chaque architecture. Un résultat important du travail de cette thèse est le développement des architectures qui permettent la mise en œuvre du SRC d'un facteur fin à un coût similaire à celui de SRC grossier, ce qui était inconcevable avant le début de ce travail.

Conclusion et Perspectives

Cette thèse propose une architecture générique optimisée d'un front-end numérique destiné aux modems multistandards de l'Internet des objets (IoT) en ciblant la fonction SRC au cœur du DFE. Les développements réalisés dans ce travail offrent des solutions SRC plus flexibles avec une efficacité améliorée. La deuxième optimisation est faite en développant une méthode de quantification optimale utilisée ensuite pour une mise en œuvre matérielle efficace du modem multistandard.

Les études et développements réalisés dans cette thèse ouvrent par la suite de nombreux nouveaux sujets de recherche. Les sujets les plus prometteurs sont: le développement des mesures de performance du SRC plus exhaustives, l'intégration du SRC dans des bancs de filtres reconfigurables, l'automatisation de la méthode de quantification, et l'optimisation éventuelle du reste des fonctions du DFE.

Contents

Acknowledgments **v**

Résumé **vii**

Abstract **ix**

Résumé Étendu en Français **xi**

Contents **xvii**

List of Figures **xx**

List of Tables **xxi**

Introduction **1**

1 Digital Front-End for the Internet of Things **5**
1.1 Internet of Things Network 6
1.1.1 What is the Internet of Things 6
1.1.2 IoT Networks Types, Standards, and Technologies 9
1.1.3 Internet of Things Deployment Challenges 11
1.2 Multi-standard Internet of Things Modems 12
1.2.1 Generic Architecture 13
1.2.2 Hardware/Software Partitioning 14
1.2.3 Multi-standard LP-WAN Modem Example 15
1.3 Generic Digital Front-End 17
1.3.1 The Main Roles of the Digital Front-End 17
1.3.2 Digital Front-End Core: Linear Functions 19
1.3.3 Complete Digital Front-End with Enhancement Functions 23
1.4 Conclusion 30

2 Unified Vision of Sample Rate Conversion FIR Filters **31**
2.1 Sample Rate Conversion 32
2.1.1 Sampling Theory Basics 32
2.1.2 Sampling Rate Conversion Definition 34
2.1.3 Unified Vision of SRC FIR Filters 37
2.2 U-F-D and Polyphase Filters: Fundamental SRC Solutions 38
2.2.1 Filter Structure Derivation 38
2.2.2 Practical Implementation 40
2.3 Farrow Structure: An Efficient Solution for Arbitrary SRC 41
2.3.1 Filter Structure Derivation 42
2.3.2 Practical Implementation 43
2.4 CIC Filter: A Multiplier Free Solution for Coarse SRC 46

2.4.1 Filter Structure Derivation . . . 46
2.4.2 Practical Implementation . . . 47
2.5 Newton Structure: A Low Cost Solution for Arbitrary SRC . . . 49
2.5.1 Newton Structure Derivation . . . 49
2.5.2 Farrow to Newton Transformation . . . 51
2.5.3 Practical Implementation . . . 52
2.6 Comparison and Applications . . . 52
2.6.1 Characteristics Comparison . . . 53
2.6.2 Implementing an SRC Application . . . 54
2.7 Conclusion . . . 55

3 Theoretical Insights on the Newton Structure 57
3.1 Equivalence Between Lagrange and Newton Interpolations . . . 58
3.1.1 Lagrange Interpolation . . . 58
3.1.2 Newton's Forward Difference Formula . . . 59
3.1.3 Newton's Backward Difference Formula . . . 60
3.2 Newton Structure Transfer Function Convergence . . . 61
3.2.1 Fractional Delay Definition . . . 61
3.2.2 The Newton Structure Transfer Function . . . 63
3.2.3 Proof of convergence . . . 65
3.3 Conclusion . . . 68

4 Generalized and Reconfigurable Newton Structures 69
4.1 Generalized Newton Structure . . . 70
4.1.1 Generalization of the Newton Structure . . . 70
4.1.2 Farrow to Newton Transformation . . . 71
4.1.3 Vector Form Transformation . . . 72
4.2 Modified Newton Structure - Hermite Interpolation . . . 72
4.2.1 V-FDF Based on Hermite Interpolation . . . 73
4.2.2 Proposed Low Complexity Hermite Interpolation Based V-FDF . . . 74
4.2.3 Performance and Complexity . . . 75
4.3 Re-configurable Newton Structure . . . 78
4.3.1 Derivation of the Transfer Function . . . 78
4.3.2 Complexity Comparison . . . 81
4.3.3 Application Examples . . . 81
4.4 Optimized Newton Structure . . . 82
4.4.1 Generalized Newton Structure Frequency Response . . . 82
4.4.2 Filter Optimization Application . . . 83
4.5 Conclusion . . . 86

5 Optimal Fixed-Point Quantization 87
5.1 Hardware Quantization . . . 88
5.1.1 Quantization Fundamentals . . . 88
5.1.2 Literature of Quantization Methods . . . 94
5.2 Optimal Analytical Quantization . . . 96
5.2.1 Integer Part Quantization . . . 97
5.2.2 Fractional Part Quantization . . . 98
5.3 Application Example . . . 106
5.3.1 CIC Module Quantization . . . 107
5.3.2 Newton Module Quantization . . . 109
5.4 Conclusion . . . 111

6 Hardware Implementation **113**
6.1 Digital Front-End Implementation 114
6.1.1 Deployment Environment 114
6.1.2 Hardware Implementation 116
6.2 SRC Implementation Architecture 118
6.2.1 Interconnection Protocol 118
6.2.2 SRC System Architecture 120
6.2.3 SRC Controller Architecture 122
6.2.4 SRC Filter Architecture 124
6.3 Implementation Results 128
6.3.1 Implementation Context 128
6.3.2 ASIC Implementation 130
6.3.3 FPGA Implementation 133
6.4 Conclusion 137

Conclusion and Perspectives **139**

Appendices **143**

A Duality and Transposition **145**
A.1 Definition 145
A.2 Transposed SRC Filters 146

List of Publications **149**

Bibliography **162**

List of Acronyms **163**

List of Figures

1.1 Evolution of the IoT concept and deployment 7
1.2 IoT Deployment Scenario for LP-WAN networks 12
1.3 A generic architecture for a wireless multi-standard system 13
1.4 The role of the digital front end in reception 18
1.5 Digital Up/Down Conversion Structures 20
1.6 Tx Digital Front-End Core 22
1.7 Rx Digital Front-End Core 23
1.8 Generic Transmitter DFE Architecture 25
1.9 Generic Receiver DFE Architecture 28
1.10 Interface between the AFE and the DFE 29

2.1 Time and frequency representations of the continuous and sampled signals . . 33
2.2 Sampling, Sampling Rate Conversion, and Reconstruction 34
2.3 Effects of up-sampling (a) and down-sampling (b) on the signal spectrum . . 35
2.4 Up-sampling / Filtering / Down-Sampling SRC Model 36
2.5 The Unified Vision of SRC FIR Filters 37
2.6 The Polyphase Structure before and after Applying the Noble Identity 39
2.7 Polynomial Based Filter Structure and its Impulse Response 42
2.8 The Farrow structure consisting of M filters $G_l(z)$ of N taps each 44
2.9 Fractional delay quantization effects on the impulse and frequency responses . 45
2.10 The interpolation cascaded-integrator-comb (CIC) filter structure of order N 47
2.11 Normalization of the CIC output gain through coarse and fine adjustments . . 48
2.12 The relation between the centered and the causal fractional delays 50
2.13 The Newton Structure implementing Lagrange interpolation of order N . . . 51

3.1 Interpolation Indexing 58

4.1 The generalized Newton structure of order 3 ($M = N = 4$) 71
4.2 Modified Newton structure adapted to Hermite interpolation of order 3. . . . 75
4.3 Modified Newton structure adapted to Hermite interpolation of order 5. . . . 76
4.4 Frequency response of the discussed interpolation methods for order 3 and 5. 76
4.5 The Spline based reconfigurable Newton structure. 79
4.6 The Hermite based reconfigurable Newton structure. 80
4.7 The optimized Newton structure with $N = M = 6$ 85
4.8 The frequency responses of Farrow and Newton optimized structures 86

5.1 Floating-point and fixed-point quantization 88
5.2 Effects of parameters and signals quantizations 91
5.3 A simplistic NB-IoT DFE example 106
5.4 CIC Decimation Filter of order 4 107
5.5 Fractional Part Quantization of the Newton Module 109

6.1 Three deployment strategies that place the DFE in different parts of the system 114
6.2 The different implementation strategies implementing the same module 117
6.3 Three deployment strategies that place the DFE in different parts of the system 119
6.4 Coarse SRC hardware module 120
6.5 Fine SRC hardware module 122
6.6 SRC controller state transition diagram 123
6.7 Newton structure of order 3 124
6.8 Pipeline architecture of the Newton structure of order 3 125
6.9 ALU-based implementation architecture 126
6.10 Newton structure of order 5 implemented in ALU-based architecture 126
6.11 Time-shared ALU-based implementation architecture 128
6.12 ASIC design flow 130
6.13 Die surface of different X-FAB XH018 logic gates 131
6.14 ASIC implementation results using the X-FAB XH018 technology 133
6.15 FPGA design flow 134
6.16 Simplified presentation of the Virtex-6 FPGA architecture 135
6.17 FPGA implementation results using the Xilinx Virtex-6 136

A.1 Duality Property Between Interpolation and Decimation 145
A.2 Polyphase Decimation Structure with $D > U$ 146
A.3 Transposed Farrow Structure 147
A.4 CIC decimation filter with $D > U$ 148
A.5 The Decimation Newton Structure of order N 148

List of Tables

1.1 Wireless Area Network IoT Standards 10
1.2 LP-WAN Standards Specifications Summary 16

2.1 Comparison of SRC FIR Filters of Order N 53
2.2 The most adapted SRC filters for different SRC applications 54

4.1 Frequency Response for order 3 interpolations 76
4.2 Frequency Response for order 5 interpolations 77
4.3 Complexity of Order 3 Newton Implementations 78
4.4 Complexity of Order 5 Newton Implementations 78
4.5 Complexity comparison of reconfigurable Newton structures 81
4.6 Complexity comparison of the optimized structures 85

5.1 Fractional Part Quantization Parameters for the CIC Filter 108
5.2 Fractional Part Quantization Parameters for the Newton Module 110

6.1 Handshake protocol behavior 119
6.2 Coarse SRC hardware module ports 121
6.3 Fine SRC hardware module specific ports 122
6.4 Scheduling of the ALU-based implementation 127
6.5 ASIC implementation results using the X-FAB XH018 technology 132
6.6 FPGA implementation results using the Xilinx Virtex-6 136

Introduction

After the Internet and mobile network revolutions of the 1990's that allowed humans to communicate across the planet, the digital world is preparing a new revolution allowing objects to communicate. The internet of things (IoT) is the label given to the network connecting these objects, that is expected to accommodate over 75 billion devices by 2025 [Luc16]. Each connected object has a specific application with particular requirements. The wide variety of applications makes the communication requirements of these objects very diverse. Hence, a large number of technologies and communication standards have been proposed for the different application cases, each using specific transmission techniques and signal characteristics. In parallel, new standards are being built with the objective to support the largest number of application cases. An example is the fourth generation (4G) long term evolution (LTE) mobile network, that defines many different device categories, each having specific technical transmission specifications. Furthermore, smart radio concepts are currently being developed, with the objective to build radio systems able of autonomously switching the technology or standard used for communication depending on the radio environment.

The diversity of modern communication solutions creates a challenge for network operators and telecommunication service providers. This is the case at TDF, a leading French radio infrastructure operator, concerned with the efficient deployment of the recently proposed IoT solutions and the future fifth generation (5G) mobile network. Considering each of these communication solutions independently induces deployment fragmentation, which kills interoperability, complicates quality of service (QoS) control, and increases deployment costs (including initial setup, maintenance, and power consumption). These issues are at the heart of 5G systems. TDF teamed up with the technological research institute b-com to solve this challenge by developing efficient deployment solutions.

To approach this problem, it is useful to refer to the Open Systems Interconnection (OSI) model, that represents network connectivity through seven layers. Implementing interoperability at the high software layers (from transport to application) offers a unified processing and management solution, however it does not solve the physical deployment challenge. To address the energy efficiency and the total costs of deploying radio infrastructures, it is necessary to consider the lower material layers (physical, data-link, and network). The priority in this manuscript is the physical layer that is the main influence on the cost and consumption of the radio infrastructure interfaces.

Regarding the physical layer, the issues of interoperability and transversality are addressed by providing a modem platform that is sufficiently flexible to accommodate the different standards and technologies. For the best energy efficiency and implementation compactness, the privileged solution is a single hardware processing system that is configurable through a set of variable parameters, allowing switching between multiple technologies. This is the software defined radio (SDR) vision, in which the processing is shared between a general purpose processor (GPP) and a hard wired structure implemented using field programmable gate array (FPGA) or application specific integrated circuits (ASIC). The hard wired part implements the main digital signal processing functions, while the GPP is assigned complex computations whose execution time constraints are relaxed. The optimal system would be the one capable of supporting all the considered technologies, while having for a given transmission

mode a complexity and consumption equivalent to that of a dedicated implementation.

Objectives and Contributions

Having provided the general framework, this work's objectives are identified. A digital modem consists of three main sections: digital front-end (DFE), baseband processing, and forward error correction (FEC). In modern systems, the DFE is usually implemented in hardware, while the two following sections are often done in software. This work focuses on optimizing the hardware implementation to reduce the cost and consumption of the physical interface. Therefore the main system in question is the digital front-end.

The DFE contains numerous signal processing functions, however it has two main roles: filtering and sample rate conversion (SRC). It is very common to combine filtering with SRC, and to use the SRC function to implement more advanced functions. This makes the SRC a core function of the DFE. Knowing this, the main objectives of this work are the following:

1. Design an efficient generic DFE architecture that is easily adapted to any communication standard or technology.

2. Optimize the SRC function in the DFE, then implement the newly developed structures, and evaluate their practical complexity and consumption.

3. Develop hardware implementation guidelines, to be used with all the DFE functions, related to the quantization problem and the implementation strategies.

To achieve these objectives, this work developed the following contributions:

1. First, generic DFE architectures for both transmission and reception are proposed, and their integration in combination with a generic analog front-end (AFE) is explained.

2. Second, a new concise presentation of the SRC filters available in the literature is developed. The purpose of this presentation is to assemble the scattered literature of SRC filters under one single unified vision. Then the work studies in depth the Newton structure, that seems promising to build new and more efficient SRC solutions. A tutorial article presenting the unified vision is planned to be published in the near future.

3. The third contribution is the development of certain missing theoretical aspects related to the Newton structure, that are needed to rigorously define the response of this filter. A new mathematical proof of the Newton series convergence was published in the IEEE signal processing letters.

    - Paquelet, S.; Zeineddine, A.; Nafkha, A.; Jezequel, P. & Moy, C., Convergence of the Newton Structure Transfer Function to the Ideal Fractional Delay Filter, IEEE Signal Processing Letters.

4. The fourth contribution is the design of new SRC filters based on the Newton structure, where three new modifications are developed: Hermite interpolation, reconfigurable response, and optimized coefficients filters. This work resulted in three conference publications and one patent application.

    - Zeineddine, A.; Nafkha, A.; Moy, C.; Paquelet, S. & Jezequel, P.-Y. Variable fractional delay filter: A novel architecture based on hermite interpolation, 25th International Conference on Telecommunications (ICT), 2018, 93-97
    - Zeineddine, A.; Paquelet, S.; Nafkha, A.; Moy, C. & Jezequel, P.-Y. Generalization and Coefficients Optimization of the Newton Structure, 25th International Conference on Telecommunications (ICT), 2018, 98-103

- Zeineddine, A.; Paquelet, S.; Kanj, M.; Moy, C.; Nafkha, A. & Jezequel, P. Reconfigurable Newton structure for sample rate conversion, IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2018, 271-275
- Zeineddine, A.; Paquelet, S.; Filtre interpolateur numérique, dispositif de changement de rythme et équipement de réception correspondants, INPI n°FR17 62632

5. The rest of this work addressed the quantization and the hardware implementation optimization problems. The fifth contribution is the development of an optimal yet simple quantization algorithm, that finds the minimal signals bit width that guarantees a given precision constraint. The developed quantization method is kept confidential and is not published in the public version of this thesis.

6. Finally, different SRC solutions for the DFE are implemented in hardware using both FPGA and ASIC, using different implementation strategies. Then the complexity and consumption of these SRC solutions and the different implementation strategies used are analyzed and compared, which helps in selecting the appropriate solution for certain given application requirements. Some of these results were published in the New Circuits and Systems (NEWCAS) conference.

   - Zeineddine, A.; Paquelet, S.; Nafkha, A.; Jezequel, P.-Y. & Moy, C. Efficient Arbitrary Sample Rate Conversion for Multi-Standard Digital Front-Ends, 17th International IEEE NEW Circuits and Systems Conference (NEWCAS), 2019

Organization of the Thesis

The work of this thesis is presented over 6 chapters having the following organization.

Chapter 1 presents the IoT context and the communication standards and technologies diversity problem. The multi-standard modem is then presented as the solution to this problem, with the DFE being at the heart of this modem. Finally, the DFE concept and its main functions are presented, and a generic DFE architecture enabling multi-standard modems is developed on both transmission and reception sides, alongside the interfaces with the AFE.

Chapter 2 develops the unified vision of the finite impulse response (FIR) based SRC filters. The five main filter structures available in the literature are grouped under one common root, that is the linear-phase FIR filters, from which the five structures are then derived. This chapter also presents the hardware implementation aspects for each filter structure in order to provide a complete and concise guide on SRC from both theoretical and practical point of views. This chapter develops the case of interpolation SRC filters, the complementary decimation SRC filters case is presented in Appendix A.

Chapter 3 addresses two missing theoretical aspects related to the Newton structure that were needed for the developments done in this work. First, the direct relation between the expressions of Lagrange interpolation and the Newton backward difference formula (NBDF) is developed. Second, a new rigorous proof of the Newton structure transfer function convergence is proposed.

Chapter 4 is based on the proposition done in [Lam16] that proposes a generalization for the Newton structure through matrix transformations. This chapters starts by developing the closed form expressions of these transformations which are then used to design new SRC filter structures. First, the modified Newton structure for Hermite interpolation is developed. Second, by writing the transfer function of the generalized Newton structure as a linear combination of two interpolation methods, the reconfigurable Newton structure is derived. Third and finally, the closed form expression of the frequency response for the generalized Newton structure is developed, and is then used in combination with filter optimization techniques to propose optimized Newton structures.

Chapter 5 studies the quantization problems, necessary for the efficient implementation of the DFE signal processing functions in hardware. The first section of this chapter presents the fundamentals and state-of-the-art of the available quantization methods. The second section then addresses the complexity of the proposed methods in the literature, and develops a new analytical method that is simple yet optimal. The developed quantization method is omitted from the public version of this thesis. The final section applies the developed method to a simplistic DFE example, in order to show how this method is practically used.

Chapter 6 develops the hardware implementation of different SRC filters on both FPGA and ASIC targets. First, the different hardware implementation constraints and strategies are presented. Then the architectures of the newly proposed SRC solutions are developed using different implementation strategies. Finally, the filters are implemented in hardware and then the complexity and consumption results are analyzed.

Chapter 1

Digital Front-End for the Internet of Things

Contents

1.1 Internet of Things Network 6
1.1.1 What is the Internet of Things 6
1.1.2 IoT Networks Types, Standards, and Technologies 9
1.1.3 Internet of Things Deployment Challenges 11
1.2 Multi-standard Internet of Things Modems 12
1.2.1 Generic Architecture 13
1.2.2 Hardware/Software Partitioning 14
1.2.3 Multi-standard LP-WAN Modem Example 15
1.3 Generic Digital Front-End 17
1.3.1 The Main Roles of the Digital Front-End 17
1.3.2 Digital Front-End Core: Linear Functions 19
1.3.3 Complete Digital Front-End with Enhancement Functions 23
1.4 Conclusion 30

Internet of Things (IoT) is the network that connects everyday devices to the Internet. This concept witnessed an important growth since early 2010, but only recently its concrete implementation and deployment started to appear. The wide adoption of the IoT trend created an important global market that is expected to continuously grow with time. However, IoT applications are numerous, and the required performance from the connected devices varies with each application. This motivated many actors to propose different transmission standards and technologies, with the objective to offer deployment solution for the IoT network. To propose solutions for all IoT applications, each standard and/or technology proposition tried to address the communication needs for IoT applications with a different approach, by exploiting the advantages of a specific transmission technique and signal type. Moreover, the number of connected devices is estimated to reach 75.4 billion by 2025 [Luc16]. The IoT network will not be uniform, but rather heterogeneous, e.g. there will be a plurality of IoT networks in each given covered area, necessitating to have a dedicated gateway for each communication standard. An efficient deployment solution would be the use of multi-standard gateways, that are capable of communicating with the largest number of standards, and thereby the largest number of devices. Then one multi-standard gateway can replace multiple ones to provide service for many devices of different standards.

These multi-standard gateways are based on a Digital Front-End (DFE), as emphasized by the software defined radio (SDR) technology [Mit95][Pal13]. The software radio concept has

an ideal objective of implementing the radio functions, found today in the analog front-end (AFE), using digital signal processing functions in hardware and software. This is desirable in order to offer a fully reconfigurable radio front-end, that can be adapted to any radio signal at both transmission and reception. However, with today's technology, this stays an ideal concept and some AFE function are always necessary, followed by a DFE having digital functions that replace or compensate what was previously processed in the analog domain. Nonetheless, by using a hybrid AFE/DFE combination, it is possible to build multi-standard gateways that process different IoT standards. This manuscript is focused on some of the DFE functions allowing this flexibility.

In this chapter, the concepts that motivated this PhD work are presented, and a generic DFE architecture is developed. The IoT concept, technologies, and deployment challenges are presented in Section 1.1. Section 1.2 then develops the multi-standard modem solution that helps addressing some of the IoT deployment challenges. The DFE function in the multi-standard modem is then studied in Section 1.3, and generic architectures are proposed for both transmitter (Tx) and receiver (Rx) implementations. Section 1.4 concludes this chapter and presents the work done in this thesis that aims at optimizing the DFE implementation.

1.1 Internet of Things Network

The context of the present thesis is the deployment of the Internet of things network. A good understanding of this context is necessary for the correct formulation of the problematic, and for identifying the best corresponding solution. This section starts by introducing the Internet of things concept in Section 1.1.1. The standards and technologies designed for IoT applications are then classified and presented in Section 1.1.2. Finally, the deployment challenges of the IoT network are identified in Section 1.1.3. The rest of this thesis then aims at finding a deployment solution that helps addressing these challenges, and facilitating the deployment of the IoT network.

1.1.1 What is the Internet of Things

The Internet of things is a commonly used term in today's research community. This section presents the IoT concept from three points of view: birth and evolution, applications, and market. This will help to better understand the IoT technologies and challenges presented in later sections.

1.1.1.1 Birth and Evolution

In the IoT literature, it is often considered that the first example of a connected object was developed at the Carnegie Mellon School of Computer Science in the 1980s [Eve90]. It consisted of a coke machine that was connected to the Advanced Research Projects Agency Network (ARPANET) at first, before it was later connected to the Internet in the 1990s. What motivated the researchers to connect this machine was their need to verify that the machine is not empty before making the trip to get a coke, since their office was far from the machine. This was the first example of a monitoring application using connected objects.

Through the 1990s, the IoT concept witnessed some basic proof of concept propositions by researchers. The advancement of integrated circuits technologies, and their price reduction, further motivated the research in this area. This concept started as a ubiquitous computing concept at first [Wei91], where the idea was to add computing and communication capabilities to everyday objects. Later on more visions for the future IoT "smart networks" were proposed, and adequate communication standards for such applications started to be investigated and developed [Raj94]. Device to device (D2D) communication using a multitude of standards (WiFi, Bluetooth, ZigBee, radio-frequency identification RFID, ...) was the main trend with

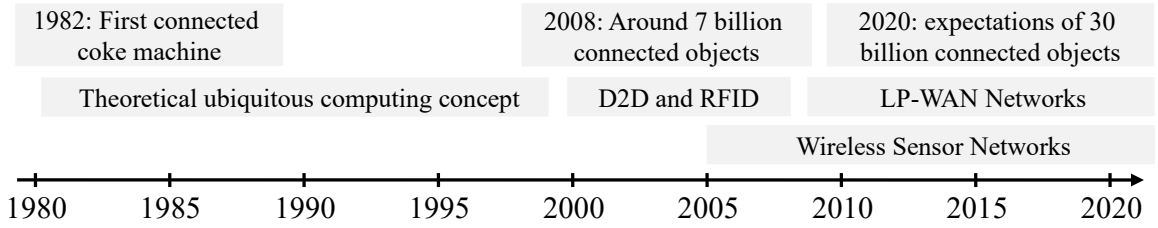


Figure 1.1: Evolution of the IoT concept and deployment

the beginning of the 21st century. RFID technology and wireless sensor networks were the foundation of many applications with a massive scale deployment.

The International Telecommunications Union ITU addressed this topic for the first time in 2005 with a dedicated report [Int05], providing a definition for this concept: "Connections will multiply and create an entirely new dynamic network of networks - an Internet of Things". According to Cisco, the IoT was born sometime between 2008 and 2009, at the point in time when more objects were connected to the Internet than the number of people on earth [Eva11]. Since 2008, the number of commercialized connected devices keeps on multiplying, and it was expected to surpass the 30 billion devices by 2020 [Lun14][Luc16].

To offer wireless connectivity to these devices, new solutions and technologies emerged destined for IoT applications. Low-power wide area networks (LP-WAN) that started appearing as of 2009 [Raz17], were a solution aimed at answering most IoT applications constraints on low power consumption and long distance transmissions. These proposed standards are discussed in more details in Section 1.1.2. The birth and evolution of the IoT concept and deployment are summed-up in Figure 1.1.

The list of applications and opportunities the IoT domain opens up is endless, and this is justified by the great number of published standards, the regional and national efforts of many countries to develop this sector, and the continuously growing market. In addition, the large scale deployment of IoT devices creates and helps the development of many other domains. For example it is a catalyst to the big data concept, that is concerned with managing the gigantic amount of data generated by all of these devices. IoT devices can also contribute to the spread of artificial intelligence, by farming their generated data and using it to train algorithms. Another domain that can also benefit from the IoT is augmented reality, where devices deployed everywhere can be used to manage the user virtual experience for example. These are some examples of the many domains tied to the IoT network.

1.1.1.2 Applications

The wide variety of IoT applications explains the great number of proposed standards that will be presented later. This variety also highlights the challenges that are associated with the deployment of the IoT network. Below is a non-exhaustive list of the possible applications that are enabled by the IoT network, sorted by the application domain. After presenting the applications, we can then deduce the constraints imposed on the devices depending on their application target.

1. **Agriculture** [Dlo15]
   (a) **Cultivation**: Connected low-cost sensors can be implemented in the field to monitor different aspects of the planting environment. These devices can also be controllers that automate certain tasks, such as the watering of plants.
   (b) **Farming**: Connected devices can be either animal health monitors, or sensors tracking the food or waste levels, and also the animals and their movement. Controllers can be used to manage automated food distribution mechanisms.

2. **Industry** [Da 14]
   (a) **Monitoring**: Sensors and RFID tags can be implemented to follow the inventory stocks in the factory, and also to monitor the automated processes in the factory.
   (b) **Controlling**: Connected actuators may be used to implement finely tuned control mechanisms in order to automatize the manufacturing process.
   (c) **Security**: Connected devices used to regulate access and detect intrusion are necessary to guarantee the security of factories.
3. **Smart City** [Li11][Zan14]
   (a) **Environmental**: In a smart city, the levels of pollution are already monitored. Whether it's natural, sound, nuclear or electromagnetic pollution. Forest fire, water quality, and many other aspects may also be monitored.
   (b) **Infrastructures**: Sensors will remotely monitor infrastructures status, such as bridges vibration, traffic congestion, waste pipelines, etc. Smart connected cars of the future are expected to be autonomous thanks to many sensors, actuators, and trackers responsible for ensuring their correct operation and safety.
   (c) **Smart Grid**: Monitoring the consumption of power and controlling its distribution is a key point to improve the efficiency of the electrical grid. Smart metering will also greatly facilitate managing the electricity grid of any scale.
4. **Smart Home** [Sto17]
   (a) **Automation**: From simple light bulbs, to coffee and washing machines, ending with completely automated heating and cooling controllers, almost everything in our homes may be remotely and automatically controlled.
   (b) **Monitoring**: Automation functions using information generated by monitoring different aspects such as temperature, humidity, power consumption, etc.
   (c) **Security**: Smart locks, intrusion detectors, and trackers, may be all connected to the Internet to alarm their owner or the police in case of any emergency.
5. **Tracking** [Gni15]
   (a) **Private Property**: Tracking valuable vehicles and properties will be possible with the IoT network by using very small devices that have geo-localization capabilities that will continuously report their position.
   (b) **Logistics & Delivery**: We can already find delivery services that offer the customer the ability to track the position of his/her parcel. Tracking logistics are also an important concern for production companies.
   (c) **Rented Assets**: In recent years we witnessed many renting services that aim to promote sharing bikes, cars, or boats between citizens. Tracking these rented assets is crucial to keep control over the business.
6. **Wearables** [Wei14]
   (a) **ID Tags**: Identification tags that a person carry are already part of many workplaces. However with the IoT network development, we might see a much wider use for these tags in everyday life, for identification, fast payment, or to save a person's medical record.
   (b) **Health**: These devices will be widespread between seniors to keep track of their health status, or between persons with a sickness that requires constant monitoring.

(c) **Fitness**: Activity trackers are already widely commercialized, and they're very helpful in informing the person using them of his activity level and his sleeping time and quality.

These different applications do not share the same operation constraints. For example when it comes to cultivation applications, the main concern would be low-power consumption, where it is not practical to frequently change the devices batteries. However when it comes to the different control and security applications the reliability of the wireless connection is far more critical. Therefore this variety of application scenarios can explain the large number of IoT standards that are proposed in the market, presented in Section 1.1.2. Each of these standards tries to answer the requirements of a certain number of applications.

1.1.1.3 Market

Every year, the number of connected devices keeps on increasing. And as the IoT applications get more mature, and the IoT standards become available, a great number of new devices will be commercialized and deployed in the next coming years. This IoT market has been investigated by many marketing and finance analysis firms, that published an important number of reports. The number of IoT devices was estimated by the International Data Corporation (IDC) to be 9.1 billion devices in 2013 [Lun14]. This number was expected to grow and reach 13.7 billion and 28.1 billion on 2015 and 2020 respectively. IHS Markit published another study [Luc16] in 2016 that estimated the number of connected devices in 2015 to be 15.4 billion, with a grow to 30.7 billion and 75.4 billion by 2020 and 2025 respectively.

Behind this great number of deployed devices, there is an important global market. The value of this market was estimated by IDC in 2013 to be worth 1.9 trillion dollars, with an expected worth of 7.1 trillion dollars in 2020 [Lun14]. In this growing sector, telecommunication providers and industries are facing new challenges. In the classical cellular network era, the service was provided mainly as a subscription based model, or as a metered post-payed service. However for the new applications of IoT devices, these business models are not always applicable. For example we find today IoT services that are billed as 1$ per device per year. Alongside the financial challenge, the technical deployment of the IoT services will also be different. With the many newly proposed IoT standards using non-licensed ISM bands, private or public sector companies can start offering wireless services, and start competing with the classical network service providers using licensed bands for IoT (LTE CAT-M1, EC-GSM, etc).

To better understand the evolution of this market, different studies investigated its adoption and its future. The work in [Hsu16] is one example, where the authors addressed the influences affecting the adoption of the IoT market. The results showed that the perceived critical mass, compatibility, and complementary were the main influences. Users also cared about the availability of access points. The widely discussed privacy and security aspects of IoT were a concern as well, however they did not seem to disable the adoption of this market.

1.1.2 IoT Networks Types, Standards, and Technologies

The majority of the IoT applications presented in the previous section rely on wireless connections to transfer their data. To answer the different needs for wireless service, many actors proposed telecommunication standards and technologies built for IoT applications. These solutions have different characteristics, and each one of them targets a certain sector of IoT applications. Before introducing these standards, the different types of IoT networks are presented. Then for each network type, the associated proposed standards are presented.

Table 1.1: Wireless Area Network IoT Standards

Standards	Network Type	Range
W-PAN		
NFC / FeliCa / Mifare	Proximity	< 20 cm
Radio Frequency Identification (RFID)	Proximity	< 10 m
ZigBee / MiWi / Thread / Z-Wave / EnOcean	W-HAN	< 50 m
WirelessHART / ISA100.11a	W-FAN	< 50 m
Bluetooth	W-PAN	< 100 m
W-LAN		
JupiterMesh / WiSun / Wireless M-Bus	W-NAN	< 1000 m
WiFi 802.11 (a, b, g, n, ac, ad, ah, af, p)	W-LAN	< 1000 m
W-WAN		
LoRa / Sigfox / Ingenu (RPMA) / Weightless / DASH7 / MulteFire	Unlicensed LP-WAN	< 10 km
EC-GSM / LTE CAT-0 / LTE eMTC / LTE NB-IoT	Licensed LP-WAN	< 10 km

1.1.2.1 Classification

Different connected devices have different constraints for their wireless connection. This can be either transmission distance, throughput, power, or something else. In the IoT literature, a large number of network types is used to classify the different standards based on their application sector. In this section, we attempt to keep the presentation concise by using the classification associated with the IEEE 802 standardization group [Jor02]. The different standards and their applications can be distributed between these three main groups: Wireless Personal/Local/Wide Area Networks (W-PAN, W-LAN, W-WAN) for short/intermediate/long range communications respectively.

1.1.2.2 Standards & Technologies

This section develops a non-exhaustive presentation of IoT standards and technologies. The objective is to help emphasize the challenges of deploying the IoT network that are discussed later. This presentation is organized following the classification given above.

Wireless Personal Area Network (W-PAN): Table 1.1 shows some examples of standards belonging to this class. Proximity networks are used for close range identification. Wireless Home and Field Area Networks (W-HAN/W-FAN) include the standards targeting home and industrial applications respectively. Bluetooth is the most versatile and the most known standard of this type of network [Bis01][Gom12].

Wireless Local Area Network (W-LAN): Examples of standards for this class are shown in Table 1.1. Wireless Neighborhood Area Network (W-NAN) regroups the standards that target metering and industrial applications, where one gateway should be capable of serving an urban neighborhood area [Ye15]. WiFi is the most known W-LAN standard having a variety of specifications [Hie10], most notably 802.11ah published in 2016, that uses the sub-GHz ISM bands and was specifically designed for IoT applications [Ada14][Kho15].

Wireless Wide Area Network (W-WAN): For the long range communication class, IoT applications have the main concern of long battery life in order to have autonomous devices. This explains the large number of solutions proposed as Low Power Wide Area Network (LP-WAN) standards [Gou15][Raz17]. Table 1.1 presents some examples for proposed standards using both unlicensed and licensed frequency bands. This thesis focuses on the LP-WAN applications that present new challenges for the radio interface conception.

1.1.3 Internet of Things Deployment Challenges

The LP-WAN IoT network imposes new challenges on both the end devices and the network infrastructure. The IoT network have typical requirements comparable to those of mobile wireless networks, that can be resumed under 6 points:

1. **Battery life**: in the majority of IoT applications, changing or recharging the battery of the devices is not practical, or even not possible. Therefore in many cases, the device is expected to have a battery that lasts many years, with 10 years being a common requirement. This is possible by reducing to a maximum the complexity of the end devices, by using power management techniques, and by limiting the transmission/reception duty cycle of IoT devices.
2. **Cost**: the core definition of the IoT network is the very large number of connected devices. Therefore the cost of both gateways and devices should be of a great concern, where low-cost options are what is going to make the IoT network possible. The total cost is not only limited to the deployed hardware value, it also includes the energy consumed by the gateways and their power amplifiers, the computation power of software processing, and all the costs required to operate and maintain the network.
3. **Coverage**: in most of the IoT applications, the goal is to achieve the farthest transmission range at the lowest possible transmit power. This facilitates and reduces the cost of network deployment, since one gateway is able to cover a large area and a large number of devices.
4. **Quality of Service**: each IoT application have a specific requirement for the quality of service (QoS). Certain applications can tolerate high latency and retransmission of lost packets, and sometimes even to lose some packets completely. However certain critical applications may rely on a good QoS connection, and standards should respect this requirement. Therefore the challenge would be to study the possibility to offer different QoS levels using the same standard, or in the other case to have different specifications for each QoS level.
5. **Scalability**: this is the ability to maintain connectivity with a very large number of devices in the network. This can be made possible in different ways, the most simple but the most costly one is by increasing the number of gateways. Another option is exploiting the diversity in time, space, and/or frequency. Finally, the most complicated option but probably the most efficient, is to use adaptive gateways and devices that are capable of managing their transmission time, frequencies, and data rates. The regional frequency regulations play a big role in defining the possible achievable level of scalability.
6. **Security**: this aspect is very challenging because most of the IoT devices are expected to be of a very low complexity. This poses many questions on security and privacy concerns. We already find many reports of denial of service attacks powered by IoT devices, showing the importance of securing the connections of these devices.

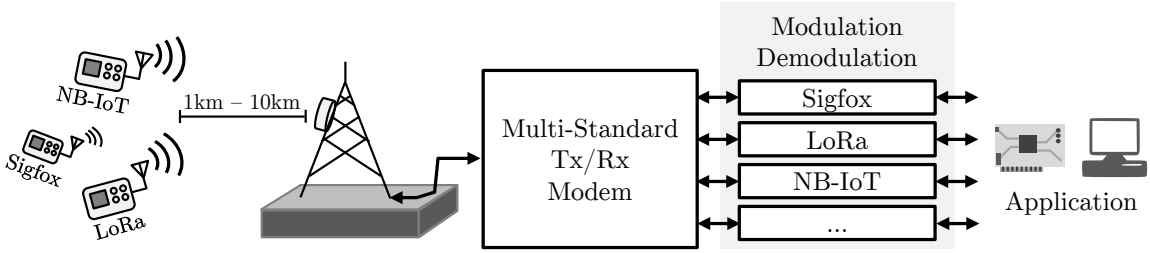


Figure 1.2: IoT Deployment Scenario for LP-WAN networks

These challenges need to be addressed on both hardware and software levels. In the next section, the scope of challenges that this work will address is presented. Once we understand the challenges we want to tackle, then it is possible to start the technical development of the proposed solution.

1.2 Multi-standard Internet of Things Modems

As mentioned in the previous section, the motivation of this work comes from the deployment challenges of IoT networks. The most common IoT deployment scenario is the one where we need to serve many battery-powered connected objects, that use wireless transmission, and are expected to be autonomous for many years. Each one of these objects transmits radio signals using a given standard, having its own specific signal waveform and transmission distance. When deploying an IoT network, the scalability and cost challenges depend on the type of network in question. For example, in a W-HAN we are dealing with tens of devices deployed in a home over an area of 50 m radius. While for an LP-WAN, we can have hundreds or thousands of devices deployed over an area of up to 10 km radius. Regardless of the differences, it is clear that in both cases there is a benefit of aggregating the gateways of the different used standards into one multi-standard modem that is capable of serving the majority, if not all, of the devices. This is illustrated in Figure 1.2 for an LP-WAN network example, where one multi-standard modem can replace multiple standard specific modems. The biggest contributions of a multi-standard modem are lowering the cost of deployment, and improving the scalability of the network. This is due to the possibility of replacing multiple standard-specific gateways by one multi-standard gateway, which can serve a great number of devices in the same area, and organize the communication of all the devices. This will also help in reducing the interference between the antennas of different separate gateways. Having such multi-standard gateways also improves the quality of service and the level of security by providing better control over the network. This in turn will help improving the general efficiency, increasing thereby the battery life and coverage area. To make this possible, both hardware and software functions need to be developed, as proven in software radio approaches [Mit95][Pal13]. The features that the software functions can offer are limited by the hardware part of the system. For example it is not possible to adapt the multi-standard modem to a new standard specification by simply updating the software, if it is not possible to reconfigure the hardware to be adapted to the signal types of the new specification. Therefore in multi-standard IoT modems we want the maximum level of flexibility for the system hardware, in order to make possible the adaptation of the same hardware implementation to different standards. In this section, a generic architecture for a multi-standard modem is considered in Section 1.2.1, and its main blocks are briefly presented. The repartition of these blocks between hardware and software is then discussed in Section 1.2.2. Finally, Section 1.2.3 presents an example for the case of LP-WAN standards.

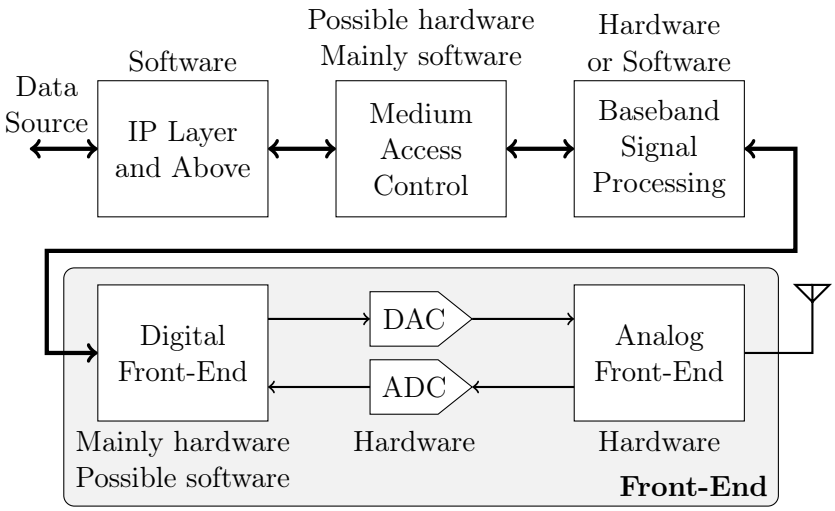


Figure 1.3: A generic architecture for a wireless multi-standard system

1.2.1 Generic Architecture

A multi-standard IoT modem is not different from any other telecommunication system. The main difference is the added support of flexibility, as proposed by software radio. A generic architecture for such system is presented in Figure 1.3, showing the main processing blocks. The possible implementation target for each block between hardware or software is also shown (detailed in Section 1.2.2). This section discusses below in details the different parts of this architecture, starting from the antenna side, and focuses mainly on the case of reception.

The antenna and the analog front-end: The antenna has the role of receiving and transmitting analog signals. Antennas operate at radio frequency, therefore the AFE are made of very high frequency components. Directly converting the received signal to the digital domain is impractical and often impossible, due to analog to digital converters (ADC) of a very high sampling rate being not achievable with today's technology at low cost and low power consumption. Even if such ADC existed, the anti-aliasing protection before the ADC is always required, in order to ensure the correct conversion. Finally, before processing the useful signal, any unwanted signals need to be removed, and the useful signals need to be prepared for processing by centering them on the correct frequency. This is why every radio frequency communication system has an analog front-end, that is an interface between the antenna, and the ADC component. The front-end part of the system is highlighted in gray in Figure 1.3. Classically, a radio front-end was an analog signal processing system. Before digital technology matured to the level it has today, engineers needed to bring the radio signal back to baseband frequency before processing it digitally. Therefore the role of the analog front-end (AFE) was to do filtering, frequency transposition, automatic gain control, and other functions to extract the baseband signal [Neu15]. However the pure AFE implementation was often destined to only one type of signal of a certain standard, and the reconfiguration capacities were minimal. Using today's technology, three function of the front-end has to be implemented using analog electronics to have an efficient system, since their implementation using digital circuits is impractical if not impossible:

1. Analog amplifiers: whether low noise amplifiers (LNA) in the receiver (Rx) AFE, or power amplifiers (PA) in the transmitter (Tx) AFE.
2. Analog filters: which may have multiple roles, either band-pass filtering, impedance matching, or an anti-aliasing role that is mainly used at the analog to digital converter (ADC) input.

3. Power switching devices: such as duplexers, mixers, circulators, or others that are used to route the radio-frequency signal.

Digital front-end: With the improvement of digital systems, and the analog to digital converters technologies, it is possible today to implement many functions of the front-end digitally, in what is called the digital front-end (DFE). This made possible the implementation of reconfigurable front-ends that are the core of efficient multi-standard modems [Tex15a]. This is because digital functions can more easily be made reconfigurable than the same functions that used to be in the AFE. Therefore it is possible to use the same DFE architecture to process different types of signal types and standards. During the time this thesis was written, it is not practically possible to implement the ideal fully reconfigurable software radio system proposed by Mitola in [Mit95]. This ideal system requires an all-digital front-end with wideband converters that are not feasible today at reasonable cost and power consumption. What is practically possible is a combination of an AFE and a DFE [Hen99][Fet02], that are connected through an ADC in reception and a DAC in transmission. Then the DFE is made maximally reconfigurable by implementing it through software, or by developing a reconfigurable parametrized hardware architecture. Through this combination, the totality of the front-end can be adapted to different standards and signal types.

Baseband signal processing: A multitude of functions may be required in this block, and its architecture depends mainly on the type of the signal processed and the associated standard definition. One of these functions is modulation, that is the classical function that transforms the data bits into symbols following a certain modulation scheme. In the majority of cases, the processing done between this step and the data source is not shared between the different standards, and each signal will have its own processing chain. Another important function is error correction, which became an essential function in telecommunication systems due to its high profitability. Whether it is a simple hamming code, or a complicated combination of convolutional and LDPC or turbo code, every standard defines its own requirements concerning this function. Other examples of baseband signal processing functions are: time and frequency synchronization, channel estimation and equalization, prefix insertion and removal, or any other function performed on the base-band signal to improve the system efficiency.

Medium access control layer and above: Between the data source and error correction, we are dealing with data bits, where we have almost complete software processing. For some specific implementations, it is possible to use hardware accelerators to maximize the speed of these functions [Sta10]. The role of these remaining blocks is either to extract and process the information sent, or to route the received packet to its next destination.

Each one of the different functions presented above is implemented in either digital hardware or software, except the analog front-end function. In the next section, the partitioning of these functions between hardware and software implementation is studied.

1.2.2 Hardware/Software Partitioning

In designing modern modems, the repartition of functions between hardware and software is one of the first decisions to make, that will greatly influence the system implementation strategy. This is a classical task that has proposed methodological approaches [Tho93][Gaj95], and has been a major topic of SDR research [Moy10]. The hardware functions are implemented using digital circuitry destined to perform one particular function. This is the case of implementations on an application specific integrated circuit (ASIC) or a field programmable

gate array (FPGA). Software functions on the other hand are implemented as a series of instructions that are run by a general purpose processor (GPP) or a digital signal processor (DSP).

Partitioning the processing functions depends on the complete system design, in terms of technical, geographical, and financial criteria. For a multi-standard IoT gateway in a city environment, the system may be installed on top of a building equipped with a fiber optics connection. In this case, it is very possible to maximize the use of software processing, and thereby reconfigurability, by transmitting the samples captured by the analog to digital converter (ADC) directly to a server that will handle the processing. This will offer a system that is very reconfigurable, and that can easily be adapted to handle variations in data traffic, or new standards updates. Such implementation approach is known as cloud radio access network (RAN) or cloud SDR. Considering now the opposite case of an implementation of an IoT gateway in a rural area, a similar approach to the previous one is not practical. This is because it is much more complicated and costly to transfer the large amount of data generated by the ADC through the network. In this case the hardware processing is maximized on site to extract the useful data bits from the signal, and then only these information bits are transferred to the application server.

In practical implementations, there is no 100% hardware or software implementation. A repartition of different proportions will always be required, since certain digital functions cannot be implemented in software. Software implementation is usually preferred to achieve a high level of flexibility, however it may not be always practical due to power consumption or execution speed limitations. In today's systems there exists a common partitioning trend that is shown in Figure 1.3. Software processing can be either implemented locally using a GPP system, or the signal can be sent to the cloud. In the latter case, many possibilities are open for the implementation architecture. By using cloud processing, it is possible to adapt the allocated computation resources to the transmitted data traffic, offering a very efficient and resilient implementation. The key point is that the border between hardware and software processing is very flexible. This border can be directly next to the analog/digital converters, or it can be deep in the processing chain next to error correction. It is also not necessary to have this border between two processing blocks, since it is possible to implement some parts of the digital front-end for example in hardware, while the rest is done through software.

The main feature of a multi-standard system is to unify the processing of functions that are common to the different standards. This is critical in order to build an efficient system. On the hardware level, it may be possible to unify similar filtering or sample rate conversion functions. And in software many management features may be made mutual. In the next section, an example for LP-WAN standards is given, to investigate what processing functions can be unified between the different standards.

1.2.3 Multi-standard LP-WAN Modem Example

To build a multi-standard LP-WAN modem, the technical specifications of each standard needs to be studied first. Table 1.2 sums-up the main specifications of the physical layer of the current leading LP-WAN standards in the market. The objective is to identify the similarities and differences between the standards. This identification will guide the design of the multi-standard modem, with the objective to find the most efficient architecture that can process the different signal types with minimal complexity. An example of similarity is the bandwidth in the order of 100 to 500 kHz for Lora, EC-GSM-IoT and LTE Cat-NB1 specifications.

Designing the hardware part of the modem is more challenging than designing the part running in software. The latter can be very flexible and reconfigurable, where all the parameters and also the functions can be updated at any time. However in a hardware implementation, the supported reconfigurable features need to be predefined. Therefore the hardware designer

Table 1.2: LP-WAN Standards Specifications Summary

Technology	LoRa	Sigfox	EC-GSM-IoT	LTE Cat-0 Release 12	LTE Cat-M1 Release 13	LTE Cat-NB1 Release 13
Governing body / std.	LoRa Alliance	SIGFOX	3GPP Rel 13	3GPP Rel 12	3GPP Release 13	
Frequency Band	Regional Sub-GHz bands		GSM bands	LTE frequency bands	LTE frequency bands GSM bands re-farming	
DL/UL[1] Bandwidth	$<$ 500 kHz[2]	600/100 Hz[3]	200 kHz	20 MHz	1.4 MHz	180 kHz
DL/UL Multiple Access	Chirp Spread Spectrum - CSS	Proprietary UNB / FHSS	TDMA	OFDMA / SC-FDMA[4]		
DL/UL Modulation	LoRa; (G)FSK	GFSK / DBPSK	GMSK optional 8PSK	QPSK $\rightarrow$ 64QAM		BPSK $\rightarrow$ 16QAM
Peak Data Rates	$<$ 10 kbps	$<$ 100 bps	$<$ 10 kbps	$<$ 1 Mbps		$<$ 150 kbps
Maximum Coverage Maximum Coupling Loss	$<$ 11 km 157 dB	$<$ 13 km 160 dB	$<$ 15 km 164 dB	$<$ 11 km 156 dB		$<$ 15 km 164 dB

[1] DL : Down-Link / UL : Up-Link
[2] DL : 125 kHz, 500 kHz / UL : 125, 250, or 500 kHz
[3] Base station listening bandwidth: 200 kHz
[4] Possible single-tone FDMA for LTE Cat-NB1 in up-link

needs to carefully consider the evolution of the standards, and how the partitioning of the functions between software and hardware is done. This will greatly influence the level of reconfigurability that is required from the hardware part [Moy10].

In this example, both LoRa and Sigfox standards operate on the sub-GHz bands, while the 3GPP solutions use GSM or LTE bands. Therefore if we want one multi-standard modem that is reconfigurable to support all of these standards, the first hardware requirement is a reconfigurable frequency transposition function, capable of bringing any kind of signal back to baseband. If the modulation is also to be implemented in hardware, we notice that GFSK is shared between LoRa and Sigfox, and BPSK is used by both CAT-NB1 and Sigfox. This can be exploited to reduce the system complexity by using one implementation to serve more than one standard when possible. Other hardware front-end functions that are closer to the antenna can be often shared between standards, such as sample rate conversion, automatic gain control, I/Q imbalance compensation, etc.

The discussion above considers the case of a multi-standard modem that is programmed to support one given standard at a time (e.g. a common platform for several applications). In this case, one reconfigurable processing chain may be sufficient to serve many standards. This is possible by configuring the modem and its function to be compatible with the standard of interest. However, in the case of a base station gateway, we are often interested in supporting multiple standards simultaneously. This is much more complicated, and often impossible to implement with only one processing chain. This difficulty is most notable in the front-end. The standards of interest can have very different carrier frequencies, signal bandwidths and waveforms. Thereby very different signal processing functions are required that cannot be implemented using only one processing chain. However, the interest always stays to maximize the mutual signal processing functions as much as practically possible.

By identifying the shared functions between the different standards, an efficient multi-standard modem can be built. As it was mentioned in Section 1.2.1, designing the front-end is the most complicated task. The majority of the functions between the data source and the front-end shown in Figure 1.3 are defined by the standards. Therefore the implementation of these functions is a straightforward deduction from the different standard definitions in question. This is not the case for designing the front-end, where for a given standard, there are many different approaches to design a compatible architecture. This is the motivation of this chapter, that is to develop a generic digital front-end, that can be then used as a reference, and can be adapted to process any IoT standard in the multi-standard modem. In the next section, we present this generic architecture and its main functions.

1.3 Generic Digital Front-End

The previous section showed the fundamental role of the DFE in building a multi-standard modem. This section aims at developing a generic architecture for the DFE, which facilitates the design of multi-standard modems by providing a reference that can be easily adapted to any standard to be added. The generic DFE architectures in this section are developed for both cases of transmitters and receivers. Before presenting these architectures, the main roles of the DFE are first given in Section 1.3.1. The DFE core architecture is then developed first in Section 1.3.2, that includes only the basic linear signal-processing functions in the DFE. Finally, by adding the performance enhancement functions, the complete DFE architectures are developed in Section 1.3.3.

1.3.1 The Main Roles of the Digital Front-End

Section 1.2 presented that the DFE is implemented to enhance the performance and capabilities of the front-end, by providing processing features that are beyond the capability of a

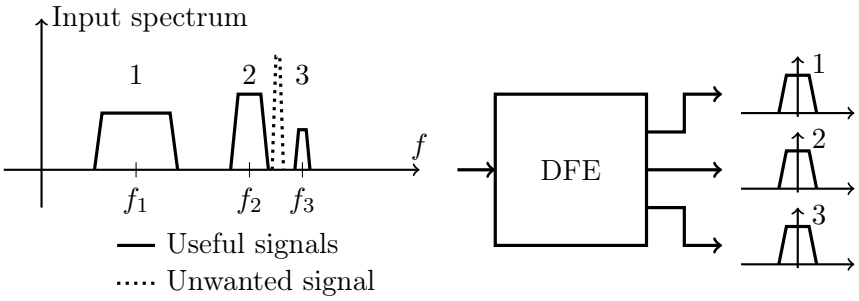


Figure 1.4: The role of the digital front end in reception

simple AFE. In the ideal case of software radio, the front-end consists of only the DFE that is implemented in software [Mit95]. In this case, the DFE should perform all the roles of a front-end, from filtering to amplifying. However such an implementation is not practically possible with today's technology, and it is only useful as a concept. A practical description of the DFE for reconfigurable radio systems is given in the work of Tim Hentschel et al. in [Hen99] and [Fet02], where the front-end consists of a combination of an AFE and a DFE. The amplifying and radio-frequency filtering functions are kept in the AFE, and then the DFE has the roles of completing the signal processing. The main DFE roles identified in [Hen99][Fet02] are: digital up and down conversion, sample rate conversion, and channelization. With the development of the DFE literature, other roles than the three identified previously were associated to the DFE [FaL11]. In the following, we sum-up the roles of a DFE in any wireless telecommunication system under four main categories.

1. Channel selection / aggregation : When considering a multi-channel wireless receiver, the DFE receives at its input a wideband signal that has been processed by the AFE. The DFE input spectrum in will contain many signals of different frequency components, with some of them being useful and others being unwanted. This is illustrated in Figure 1.4, where the channel selection role of the DFE can be directly seen. This role consists of filtering out any unwanted signals, and then extracting the useful signals of the different channels. These extracted signals are most often translated back to baseband, and their sampling rate is adapted to improve the processing efficiency, as well as their amplitude level using an automatic gain control.

In the case of a wireless transmitter, the role of the DFE is more simple. When the radio equipment has several signals to transmit, its role consists of combining the different useful signals provided at its input, in order to create the final signal that is to be transmitted over the antenna. This final signal contains the useful signals centered on different frequency channels.

2. Improving efficiency : On the receiver side, the DFE can also play a role in improving the system efficiency by using digital functions. For reception, this can be done through automatic gain control (AGC), DC offset cancellation (DCOC), I/Q imbalance correction, automatic frequency control (AFC), etc. All of these functions improve the signal quality, and contribute to lowering the error rate of the demodulation, thereby improving the general system performance.

On the transmitter side, the efficiency is improved by using the crest factor reduction (CFR) and digital pre-distortion (DPD) functions. These functions aim at improving the power amplifier efficiency, by shaping the signal in a way that allows the optimal use of this power amplifier. These functions are presented in Section 1.3.3.

3. AFE distortion compensation : At the receiving end, both the low noise amplifier and the analog mixers my distort the signal amplitude and phase. The DFE has the role of compensating these distortions, using digital filters that have the inverse response of the AFE.

For transmission, the non-linearity compensation is done in advance. The power amplifier is the main concern of non-linearity, and the digital pre-distortion (DPD) function is used to linearize the total response of both the DPD and power amplifier.

4. Front-end reconfigurability : This is the most prominent role in this study. The DFE enables multi-standard operation by providing hardware functions that can be reconfigured and adapted to different signals. This can be filter banks having various channel selection configurations, or sample rate conversion functions having a reconfigurable factor and filter responses. Almost all digital functions can be made reconfigurable, which is the strong point of using a DFE in the wireless system front-end. This reconfigurability is controlled using a certain set of parameters that are managed in software.

1.3.2 Digital Front-End Core: Linear Functions

The DFE roles presented above are realized using numerous functions that are either common to both receiving and transmitting DFE, or exclusive to one side. In the rest of this section, we develop a brief description of the most common functions found in the DFE, that we use to construct the generic architecture. These functions are separated over two sections. In this section, the DFE core is presented that consists of filtering and digital up/down conversion [Fet02]. These functions are linear and time-invariant in the case of filters and frequency translation, or time-variant in the case of sample rate conversion. In the next section, the complete generic DFE architecture is developed by adding the enhancement functions used to improve the DFE performance. Not all of these functions are required in every DFE. The choice of the included functions depends on the processed standards, the destined application, and the required performances. The objective of this work is to develop a generic DFE architecture that serves as a starting point reference.

1.3.2.1 Digital Up and Down Conversion

The main role of this function is to manage the signal's carrier and sampling frequencies. This function has always been a part of digital multirate systems, classically used to implement quadrature modulation techniques [Cro83, p.52]. During the 1990s, many propositions of digital transceiver architectures used digital up and down conversion (DUC/DDC) as parts of their implementation [Lue90; Che95; Osc95]. The wide adoption of these functions in digital transceivers pushed the main FPGA development tools providers to offer IP cores for these functions. Xilinx DUC and DDC IP cores are documented in [Xil02; Tar07; Cre08], with application examples for GSM, WCDMA, and MC-GSM respectively. Altera DUC and DDC applications are documented in [Alt07b; Alt07a; Alt09], with examples for W-CDMA, WiMax, and multi-standard receivers respectively. The DUC and DDC functions consist of either one or a combination of frequency transposition and sample rate conversion, that are detailed below.

Frequency Transposition: This function is not always included in the DFE, where in some cases the frequency transposition and I/Q modulation or demodulation is done by the AFE. In many cases, this function is associated with the I/Q modulation or demodulation, however this is not always necessary and it depends on the modulation type used. Frequency transposition is responsible of managing the position of the signal on the frequency axis, and most generally between intermediate frequencies (ADC/DAC side) and baseband frequencies

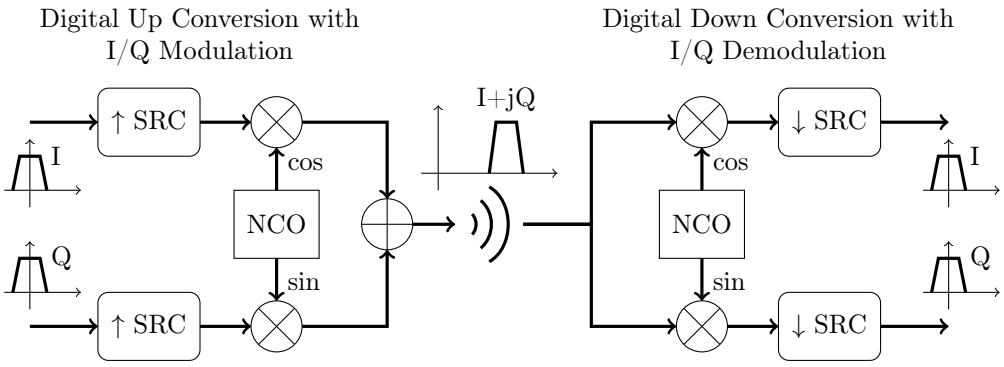


Figure 1.5: Digital Up/Down Conversion Structures

(baseband signal processing side). This can be seen in Figure 1.4, where the useful signals are translated from frequencies f_1, f_2, and f_3 back to baseband at reception. In this case, the Digital Down Conversion (DDC) block shown in Figure 1.5 is used to extract the I/Q signals in BB from the complex IF signal. In the transmitter case going from modulation to the AFE, the Digital Up Conversion (DUC) block is used to create the complex intermediate frequency (IF) signal using the baseband (BB) I/Q signals. This function can be implemented using the architecture shown in Figure 1.5 in the general case, or through sub-sampling techniques in some special cases. The general case architecture consists of three main components:

1. Numerically Controlled Oscillator (NCO), which is responsible for generating the orthogonal sinusoidal waves. This can be implemented using the CORDIC algorithm or by storing the sinusoidal waves values in a memory block, that are then read using an incremental counter.
2. Mixers: which are multipliers, having at their inputs the useful signal and the NCO's output. In the transmitter case, the outputs are then added together to create the complex signal to be transmitted.
3. Sampling Rate Conversion (SRC): when translating the signal's frequency, the sampling rate should also be adapted to always respect the Nyquist sampling criterion, and to optimize the calculation complexity and power consumption.

In multi-channel or multi-standard systems (also known as multimode system in the literature), multiple instances of frequency transposition are implemented in parallel. Each one of these instances will have an NCO generating a specific frequency that is used to transpose the associated signal between its proper intermediate frequency and baseband. Examples of multimode DUC systems can be found in chapter 12 of [FaL11] (Figure 12.29 and 12.30), and in [Alt09] (Figure 18) proposing a DUC system for simultaneous GSM and WCDMA/LTE processing.

Sample Rate Conversion: The SRC function is not exclusive to the DUC/DDC blocks, and can be used at different places in the DFE. This function handles the sampling rate of the signal passing through the DFE, where this frequency increases at transmission as we get closer to the antenna, and decreases at reception when going toward the modulation/demodulation side. When manipulating the signal sampling rate, the digital signal must be protected against aliasing and imaging caused by lowering and raising the sampling rate respectively. In general, we identify two types of SRC: Coarse SRC and Fine SRC. The first type concerns the big integer (or sometimes simple rational) SRC factors, while the second type concerns finely tuned factors around the value of one. We find different structures for SRC filters in the literature, each better adapted to a certain SRC factor. For example, the polyphase

[Hsi87], half-band [Cro83, p.155], and CIC [Hog81] filters are better used with coarse SRC operations. Variable fractional delay filters [Laa96], such as the Farrow [Far88] and Newton [Leh09] structures are better suited for fine SRC operations. Chapter 2 of this thesis presents a survey of the literature related to this function that will be the focus of this PhD work.

1.3.2.2 Filtering and Channel Selection or Aggregation

The filtering function is more prominent in the receiver DFE than in the transmitter DFE. For multi-standard operation, the DFE is responsible for recovering different received signals transmitted over different bands. This requires that the antenna captures a wide band spectrum, and therefore many interfering signals may be captured in the spectrum alongside the useful signals. The filtering function ensures that the DFE removes any unwanted frequency components, and that the output contains the useful signals only. The filtering function is often included in the more prominent channel selection function.

The channel selection function is used to separate and extract the different useful signals in the received spectrum. These signals on the different channels may belong to the same or to different standards. After separating these signals, the outputs of this function might be delivered separately, or can be multiplexed over the same stream. This function can be combined with the frequency transposition function to output the signals at baseband frequencies directly using direct Fourier transform (DFT) filter banks. Figure 1.4 illustrates these functions, where the unwanted signal in the input spectrum is filtered, and the different useful signals are separated and delivered at the output. Designing this function is complex, since we expect in modern systems that the channelizer should be capable of handling signals of different bandwidths. A study of the classical filter bank channelizers is developed in [Mah11], that also proposes new channelizer architectures.

Finally the channel aggregation function is used in the transmitter DFE. Filter banks are not needed on the transmission side since the system controls in advance the transmitted signals and the corresponding used frequency bands. For both multi-channel and multi-standard operations, multiple signals are to be transmitted at the same time. Therefore at a certain point in the transmitter DFE, the different input signals need to be combined into one signal that will be delivered eventually to the AFE. This is the responsibility of the channel aggregation function, that is analogous to a summation operation that combines the signals centered on different frequencies. Therefore its design is much simpler relatively to the channel selection function.

1.3.2.3 Tx Digital Front-End Core

A generic architecture for the Tx DFE core that supports the transmission of multiple IoT standard signals simultaneously is shown in Figure 1.6. At the input we find the different data streams in parallel that are fed to their corresponding modulation function. Before combining the different modulated signals, each signal must be translated into its proper position on the frequency spectrum. This translation is performed through the digital up-conversion shown in Figure 1.6. In this case we require a multi-mode digital up-conversion operation as developed in [Tex15b]. At first each in-phase and quadrature signal sample rate must be increased through the multi-stage SRC block to allow the correct frequency translation. The multi-stage SRC implementation allow for better reconfigurability with low complexity, and therefore giving the possibility to use the same transmission chain for multiple types of modulation [Alt09]. Once the signals sampling rate is sufficiently increased, the signals are mixed with the different NCO outputs of different frequencies to translate each signal to its proper channel on the frequency spectrum. The sum chain function has the role of combining all these translated signals into one final signal that is to be sent to the AFE for transmission. Before sending the signal to the AFE, the sum chain output is run through a

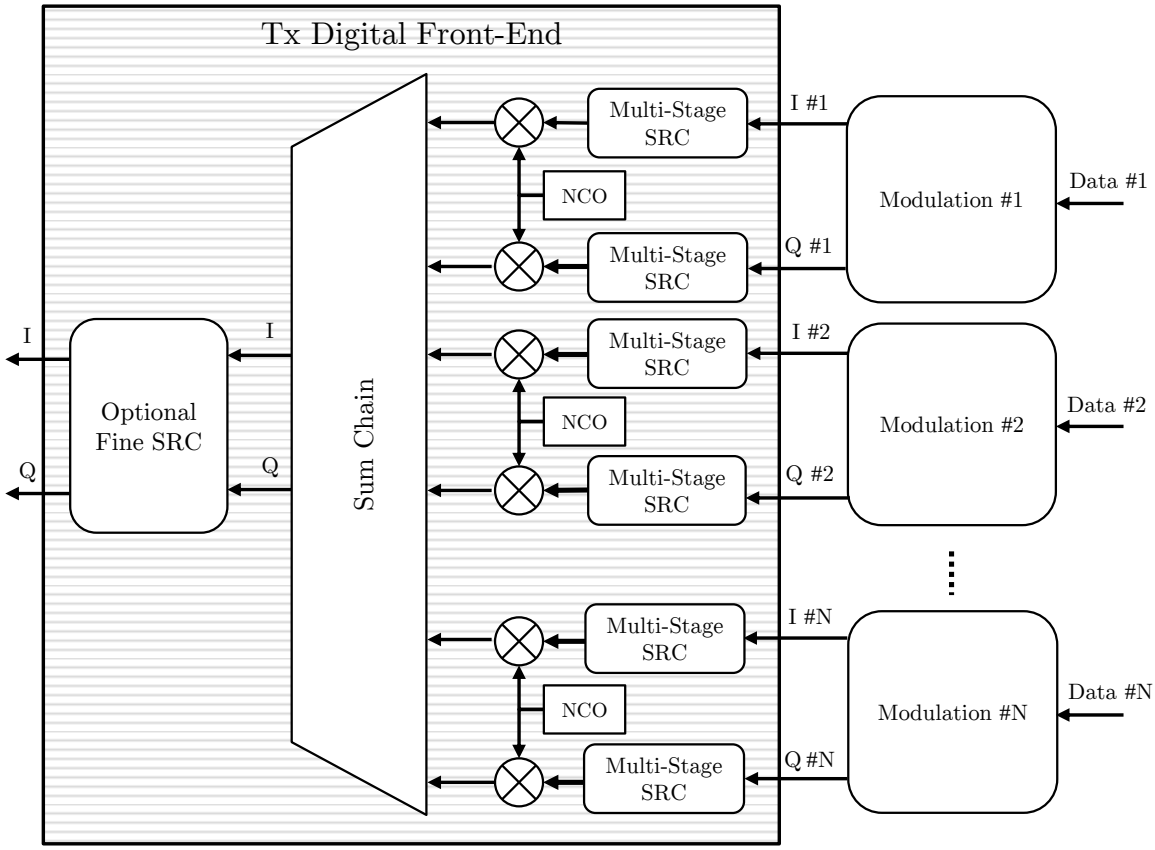


Figure 1.6: Tx Digital Front-End Core

fine SRC operation that has the role of matching the signal sample rate to the required rate by the DAC interface between the AFE and DFE.

1.3.2.4 Rx Digital Front-End Core

A generic architecture for the Rx DFE core that supports multi-channel and/or multi-standard reception is shown in Figure 1.7. In the majority of cases, the input sampling rate is much higher than required in order to improve the signal-to-noise ratio and to avoid any unexpected aliasing. This is why a coarse SRC operation using CIC filters is often used to lower the sampling rate by an important factor to keep only the required bandwidth [Tex15b; Alt07b; Cre08]. A gain control is applied at the CIC output depending on the SRC factor and the CIC gain. Following the CIC filter, we find in most implementations a decimation FIR filter that has the role of further filtering, decimation, and of compensating the passband droop in the CIC frequency response [Tex09a; Xil02]. Another gain control may be applied to the FIR decimation output. Before sending the signal to the channelizer, a finely adjusted SRC operation may be required depending on the standard that may be performed using the Newton structure.

The next block, referred to as the channelizer in the literature [Fet02; Mah11], has the role of extracting the useful signals from the input spectrum. This is required in both cases of multi-channel or multi-standard operations, where the system is expected to receive multiple signals simultaneously. The channelizer can be implemented using per-channel filtering chains, DFE filter banks, or many other more complex structures [Mah11]. This block is equivalent to frequency de-multiplexing, with the output of this block consisting of the different channel signals separated with their sampling rate adapted. Each output may pass then by a dedicated gain control implemented using a digital AGC to normalize the signals level. At this point in

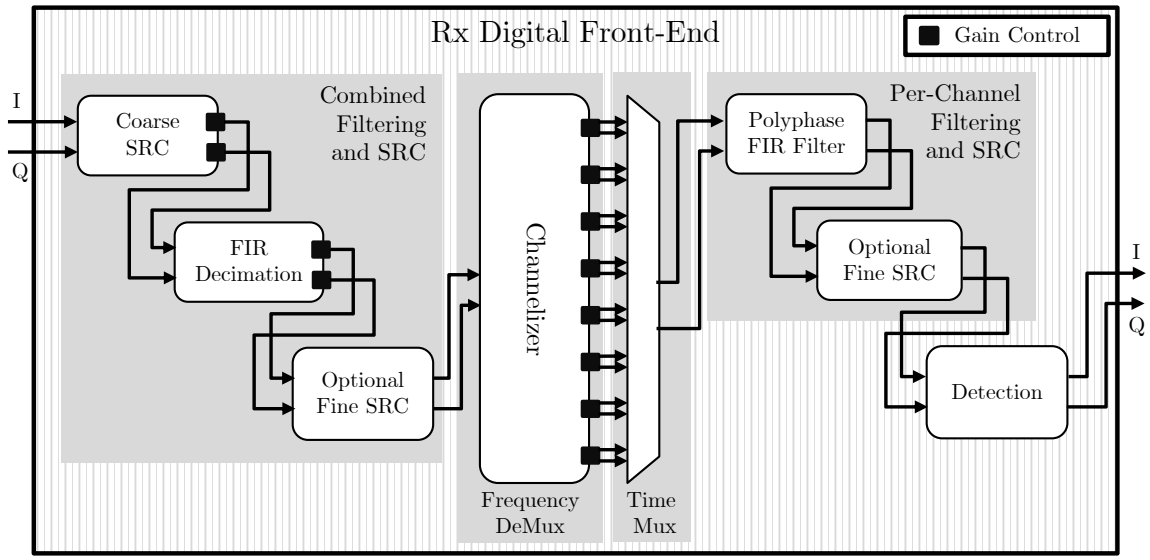


Figure 1.7: Rx Digital Front-End Core

the DFE, the sample rate is much lower than the digital clock rate, giving the possibility of time-shared processing of the signal [Tex15b, p.41]. Therefore the outputs of the channelizer are time multiplexed and then sent channel-by-channel to a polyphase FIR filter for any final filtering requirements. Then the signals may go through one more fine SRC adjustment before the detection process, that is responsible of identifying if an active transmission is taking place on each channel. If this is the case, the detection module forwards the useful signals for further processing.

1.3.3 Complete Digital Front-End with Enhancement Functions

The previous section presented the functions that form the core of the DFE. However many more functions can be included in the DFE to improve its performance and efficiency [FaL11]. In the following we present a very brief description of these functions that are grouped according to their usage: Transmission (Tx) or Reception (Rx) functions.

1.3.3.1 Tx Enhancement Functions

Crest Factor Reduction: Crest Factor measures the ratio of the highest peak in the signal to its average value, which is similar to the Peak-to-Average-Power-Ratio (PAPR) measurement. In order to increase the efficiency of the power amplifier (PA), it is desirable to operate the PA at its full power capacity. However, when the crest factor is too high, the average power will be low in order to avoid saturating the PA. Crest Factor Reduction (CFR) is the method of lowering the crest factor of the signal, increasing thereby its average power, and improving the PA efficiency. This function can be done through many different techniques: from simple clipping and filtering of the signal, up to using more complex techniques such as selected mapping. An overview of crest factor reduction techniques as of 2005 was published in [Han05], and a more recent discussion of these techniques can be found in chapter 11 of [FaL11].

Digital Pre-Distortion: Another inconvenience that may limit the operation of the PA at full power is the intrinsic non-linear characteristic of the PA. This non-linearity has both static and variable parts. Digital Pre-Distortion (DPD) attempts to compensate for this non-linearity by modifying (pre-distorting) the signal in a way to make the total response of the

DPD and the PA combined as linear as possible. We mainly distinguish 2 types of DPD. The memoryless DPD, which has a simple implementation but has sub-optimal performance. And the memory DPD that is more complex, however offering better performance. The interested reader can refer to part II of [FaL11] that offers a good introduction to both DPD and CFR techniques, and the possible methods to combine them as one function.

I/Q Imbalance Correction: IQ imbalance takes place in both the transmitting and the receiving front-end. The AFE mixer stage results in an imbalance of the total response between the I and Q lines due to different analog filters and the DAC or ADC properties, in additions to a non-exact 90 degrees phase difference between the LO outputs. This imbalance creates cross-talk between frequencies components around the center frequency, that results in the creation of signal images in the spectrum. Implementing this function consists of three main steps. First, the I/Q circuit is modeled to define the imbalance parameters to be considered. Second, an estimation of the imbalance parameters has to be done. Third and finally, using the estimated parameters, whether pre-distortion or post-distortion is applied to compensate for the imbalances. The imbalance parameters estimation can be done either blindly or through data-aided methods. The interested reader may refer to chapter 16 of [FaL11] for more details, where the basics of this function and a quick survey of the related literature are presented.

1.3.3.2 Tx Digital Front-End

Based on the Tx DFE core developed in the last section, the complete DFE generic architecture in Figure 1.8 is developed by including the enhancement functions. We also consider in this case two options for the output I/Q modulation. The first problem concerns the output signal of the sum chain. Combining the modulated signals will result in an output that may have important peaks in the time-domain waveform. As discussed above, this can result in a poor efficiency for the PA. To handle this issue, the CFR function is inserted after the sum chain to improve the signal's waveform by reducing its CF. The CFR function is followed by the DPD block that also aims to increase the efficiency of the PA by pre-distorting the signal in order to compensate for the PA non-linearity. To implement efficient and high performance CFR and DPD function, most algorithms require a feedback from the PA output [FaL11, p.150][Tex15b; Tex09b]. This feedback (Tx Feedback) is processed by the DPD and CFR control block. The latter then configures the DPD and CFR functions correspondingly. This control block can also manage the gain level of the analog transmission chain through the VGA control signal that is sent to the Tx AFE. The interface between the Tx DFE and AFE is discussed in more details later below.

Before sending the signal to the AFE, the DPD output is run through a fine SRC operation that has the role of matching the signal sample rate to the required rate by the DAC interface. This interface can be implemented in two different ways. The first option is by performing the I/Q modulation digitally, and then the complex signal is fed to the output DAC [Alt09]. The second option is by having an analog I/Q modulation, and in that case two separate DACs are required [Pan16]. Having two separate paths in the AFE is a source of unbalance that can cause non-linear distortion and image creation. This is handled through the I/Q pre-distortion function that compensates for the unbalance [FaL11, p.508][Din08]. This function also requires a feedback signal from the AFE for an efficient implementation. Finally, the outputs of the Tx DFE are sent through digital to analog conversion to the AFE. The AFE then creates the RF signal that will be sent to the PA to be transmitted next by the antenna.

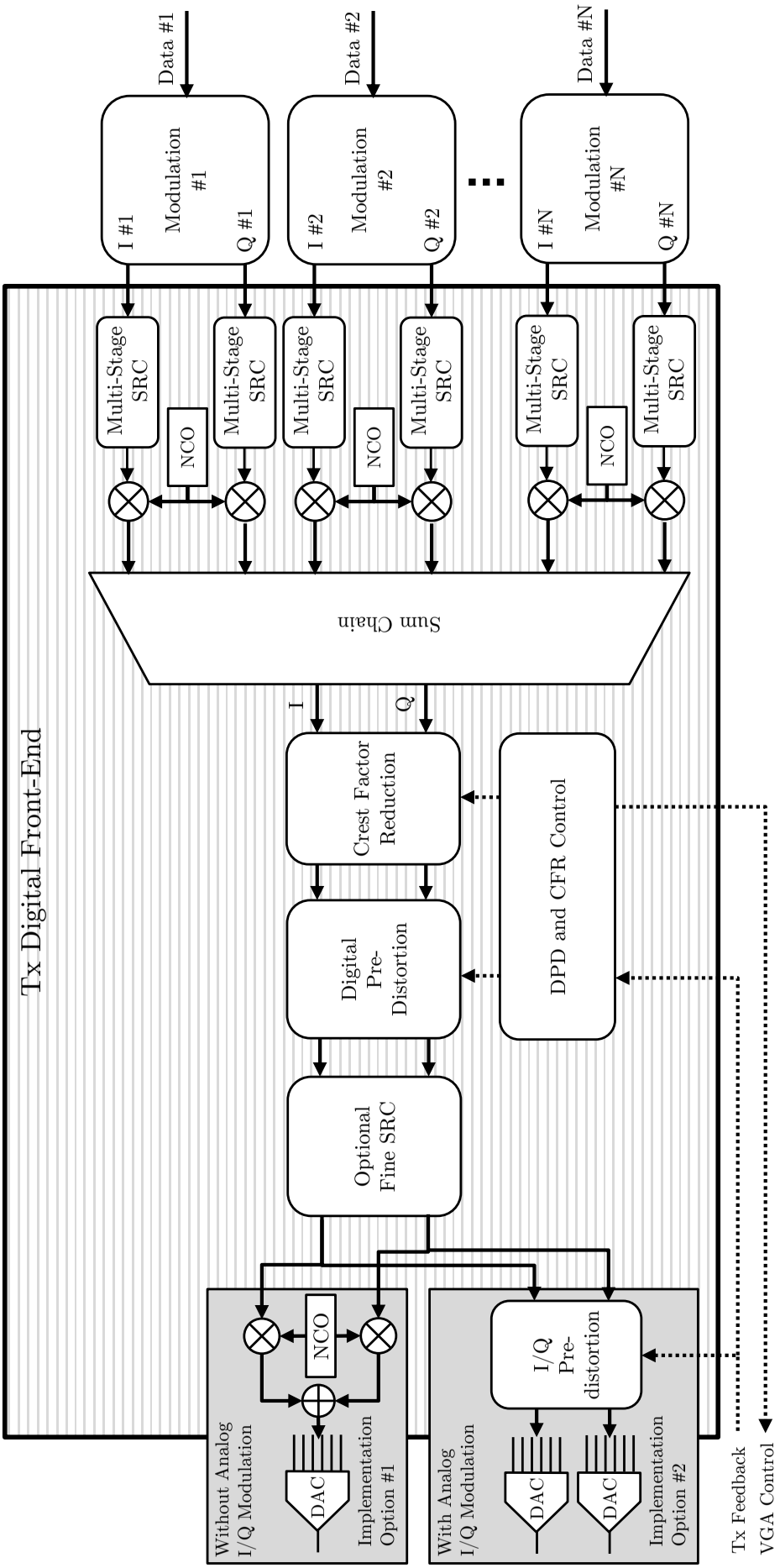


Figure 1.8: Generic Transmitter DFE Architecture

1.3.3.3 Rx Enhancement Functions

Equalization & Compensation: Channel equalization tries to compensate for the distortion that are due to the transmission channel, that corrupt or lower the quality of the radio-frequency signal. Channel equalization can be done blindly through relying solely on the signal properties [Joh98], or through learning by taking into consideration the channel characteristics (i.e. using pilot signals to estimate the channel). The latter case has many different approaches, most notably the linear zero forcing (ZF) and minimum mean squared error (MMSE) techniques [Kra00], turbo equalization techniques [Tuc11], and many others.

The other type of equalization is concerned with compensating the frequency response of the AFE and the ADC, where the analog filters and components are most often non-ideal in their amplitude and phase response. By inverting the response of the AFE in the DFE, a linear response is obtained at the output. This function has been specifically addressed by Horlin and Bourdoux in [Hor08], and in the PhD Thesis of Tandur in [Tan10]. It is also common to use this function to compensate for the frequency response defects of other components in the same DFE. A common example is the FIR CIC compensation filter that compensates the passband droop in the CIC frequency response [Alt07c].

Synchronization: This function is needed since the receiver should be locked to the received signal in order to perform correct demodulation. The implementation of this operation can be done in one of three ways:

1. All-Analog: included in the AFE, which will control the local oscillator (LO) clock.
2. Hybrid: included in the DFE, with a feedback into the AFE to control the LO clock.
3. All-Digital: all the synchronization functions can be done in the digital domain (after the ADC)

For an introduction to the different types of this function, the reader may refer to the classic books of U. Mengali [Men97] and H. Meyer [Mey01]. Mainly on the physical layer, we are concerned with three types of synchronizations: carrier phase synchronization, sampling and carrier frequency synchronization, and timing synchronization. The lack of these synchronizations will results in an offset in the constellation, a rotating constellation, or the creation of inter-symbol interference respectively for the three types. It is important to also study the synchronizers stability, where it is possible to have the synchronization wrongly locked to a false stable point, which is referred to as cycle slip.

Automatic Gain Control (AGC): Due to fluctuations in the channel's response, and in the receiver's temperature and other conditions, the output signal from the AFE can have a varying amplitude level, which may cause loss in ADC dynamic range in case of low amplitude levels, or may result in ADC saturation in the case of excess amplitude levels. It is also often required to have a constant power level at the DFE input. These problems are addressed by the AGC function that aims to keep a constant signal level that uses the full ADC dynamic range. As in the case of the previous synchronization function, the AGC can have an all-analog, hybrid, or all-digital implementations. The all-analog approach is the most classical one and is well covered in the literature [Pér11; Lia08], however it lacks the reconfigurability of hardware implementations. The most flexible solution is the hybrid one, where the AGC control is run by the DFE and then the generated control signal is fed back to the low noise amplifier (LNA) in the AFE to adjust the gain [Stu06; Che12]. The all-digital implementation is another common efficient approach [Du03; Vuc09], however its drawback is not being able to control the input of the ADC. Therefore in some cases it may be of interest to combine the hybrid and the all-digital AGC in the same DFE for a complete AGC solution.

DC Offset Cancellation (DCOC): The purpose of this function is to protect the ADC dynamic range from the DC offset caused by the AFE. Especially in direct conversion receiver (DCR) architectures, the LO leakage on the LNA and itself (self-mixing) will result in a DC component in the output that will risk a one-side saturation of the ADC. Non-linearity of AFE components can be another source of DC offset. To protect the ADC from saturation, only all-analog and hybrid approaches are efficient. The preferred hybrid implementation estimates the DC offset value in the DFE (using a low pass filter or an iterative approach), and then it applies the opposite bias to the LNA or another components of the AFE through a feedback loop. However we also find combined all-digital and hybrid DCOC implementations, that performs both fine and coarse DCOC respectively [Rah11]. The AGC and DCOC functions are usually implemented together in a DFE due to their similar feedback architecture. A comprehensive illustration of the implementation of hybrid AGC and DCOC function was developed by Gore et al. in [Gor16], that shows the interconnection between the digital and analog sections of the front-end.

1.3.3.4 Rx Digital Front-End

Starting from the Rx DFE core in Figure 1.7, the complete generic architecture for the Rx DFE is developed hereafter. This architecture is shown in Figure 1.9. The Rx DFE can have multiple processing chains in parallel of different complexities to support multiple standards simultaneously [Sin04; Hol14], and not necessarily only one chain as shown in this example. The input interface of the Rx DFE with the Rx AFE has two implementation options. The first one considers the case of an AFE without I/Q demodulation, therefore only one ADC is used to convert the complex signal. The I/Q demodulation is then performed digitally to extract the in-phase and quadrature signals [Tex09a]. In the second implementation option, the case of analog I/Q demodulation is considered, where two separate ADCs are required in baseband to convert the input [Pan16]. In this case, the I/Q imbalance correction function is useful to compensate for any analog circuit unbalances that may cause non-linear distortions [Sch01].

For both implementation options, the ADC output is fed to the AGC and DCOC functions. The AGC function is used to maximize the use of the ADC dynamics and to prevent its saturation. This is done through the "AGC control" signal that is fed back to the Rx AFE that modifies the signal amplitude as required [Pai01]. The DCOC function is used to eliminate any DC offset resulting from non-linear distortion and analog leaking signals in the AFE. The DCOC can be separated into coarse and fine corrections [Rah11]. The first is done through the feedback "DCOC control" signal that biases the AFE to remove the DC component. The second is done through the "Fine DCOC" blocks in the DFE that aim to minimize the DC component level.

The I/Q signal is then run through the Rx DFE core, that consists of SRC and filtering operations to extract the useful signals. Phase equalization is then added to compensate for the AFE non-linear distortions on the signal's phase. Following the phase equalizer, the detection function is responsible for detecting if there is an actual transmission on the channel. Only in the case a transmission is detected, the automatic frequency control (AFC) block is activated. The AFC function has the role of detecting any carrier or sampling frequency synchronization deviations. Information from the detection functions can be used to implement more efficient AFC algorithms. Once the deviations found, the carrier frequency synchronization is corrected using a feedback control signal "AFC Control", that may be sent to the AFE to adjust the oscillator clock [Fu17], or to the I/Q demodulation NCO for digital tuning. The sampling rate synchronization is corrected with fine-tuned SRC that follows the AFC function. Finally the synchronized baseband signals are sent to software for demodulation and further processing.

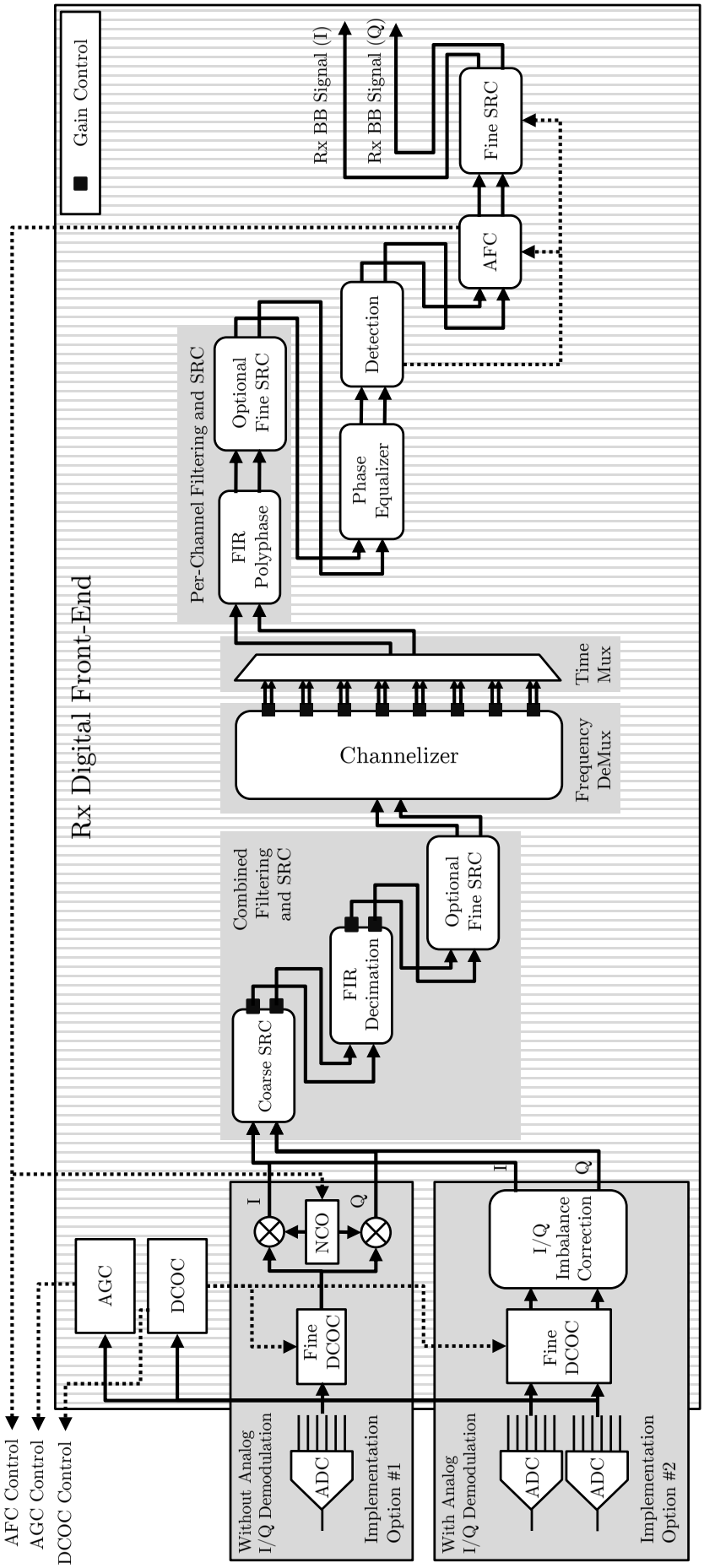


Figure 1.9: Generic Receiver DFE Architecture

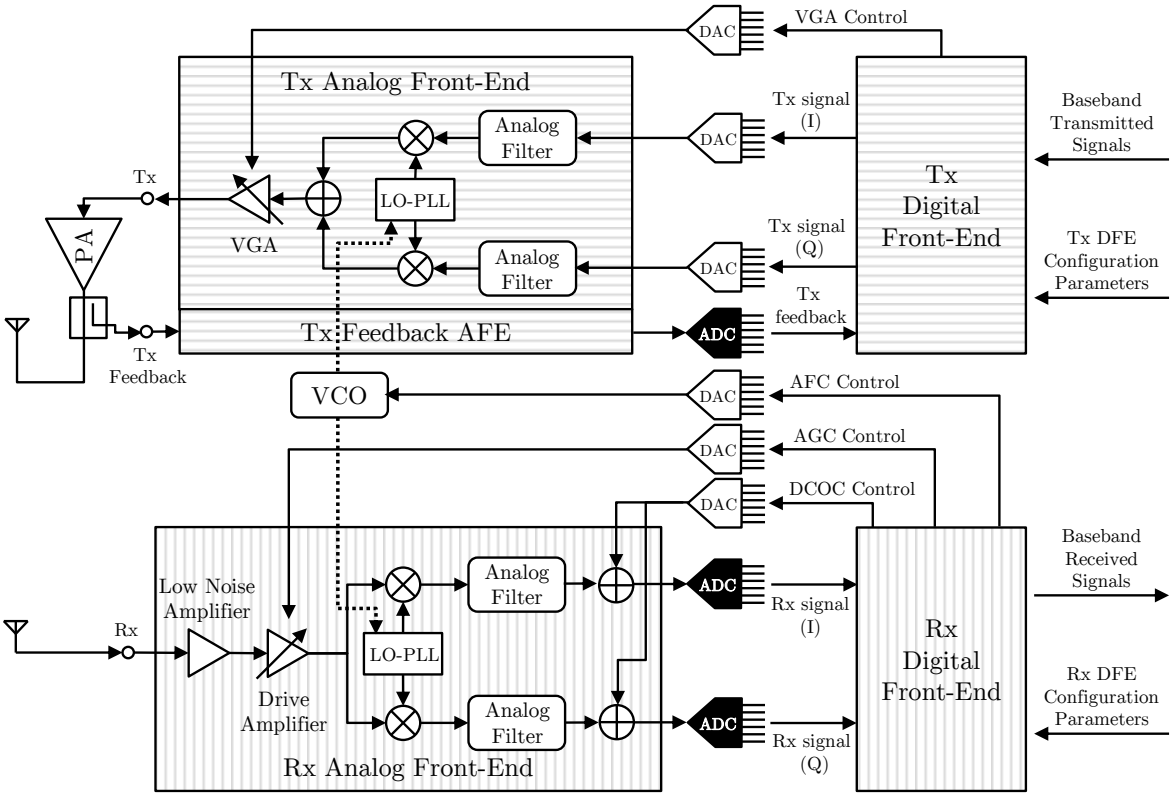


Figure 1.10: Interface between the AFE and the DFE

1.3.3.5 Interface Between the AFE and the DFE

In Figure 1.10, simple generic Tx and Rx AFE architectures are shown as an example [Pan16]. It is considered in this case that the I/Q modulation and demodulation are done in the AFE, with the data interface between the AFE and DFE implemented using separate I/Q DACs for Tx, and ADCs for Rx. The design and implementation of this AFE system is delicate due to the many imperfections of analog circuitry [Neu15]. These can be corrected either in the DFE, or by using feedback control signals that tune the AFE operation. In this example, we consider the control signals generated by the enhancement functions included in the generic architectures (Figure 1.8 and Figure 1.9). We can identify four interfaces in these developed architectures, that are the following:

1. Data interfaces between the DFE and the baseband processing section that are digital, either in software or hardware. In general, these interfaces carry the modulated baseband signal.
2. Data interfaces between the DFE and the AFE, which carry the signals transmitted and received.
3. Configuration digital interface between the DFE and the software section. This interface is responsible for carrying the parameters used to configure the DFE (SRC conversion rates, filter coefficients, ...).
4. Control interface between the DFE and the AFE through DACs. This interface carries the AFE control signals generated by the enhancement functions (AGC, DCOC, AFC, CFR, etc...). The feedback from the Tx antenna to the DFE is also part of this interface.

In the Tx control interface case, the "VGA Control" signal is used to command the variable gain amplifier (VGA). The "Tx Feedback" signal going back to the DFE is generated by the "Tx Feedback AFE" that takes as input the PA output signal. For the Rx control interface,

there are three control signals. The "DCOC Control" signal biases each of the I/Q analog paths that may have different DC offset levels [Gor16; Gor17]. The "AGC Control" signal commands the "Drive Amplifier" to adjust its gain. It is possible to have multiple amplifiers that are controlled by the AGC signal [Gor16]. Finally, the "AFC Control" signal is used to tune the voltage controlled oscillator (VCO) [Anz13], that controls the clock of the local oscillator phase locked loop (LO-PLL).

1.4 Conclusion

This chapter developed generic architectures for the transmitter and receiver digital front ends, with a focus on LP-WAN IoT technologies. The variety of these technologies stems from the diversity of IoT applications. In order to be capable of deploying an IoT network supporting all of these technologies at a reasonable cost, multi-standard gateways are essential. To build these gateways, the DFE is required to offer the needed flexibility to process different signal types. This processing is done through many functions that are either shared between the Tx and Rx DFE, or exclusive to one of the two. The generic architectures of the Tx and Rx DFE are built, and a generic implementation of a hybrid front-end containing both analog and digital sections is presented. These architectures confirm the claims in the DFE literature that two functions form the DFE core: sample rate conversion and filtering.

After developing the generic DFE architectures, our interest is to optimize the implementation of the proposed generic DFE architectures. The main focus of this work is on the modules of the DFE core that appear at several points in the architecture: sample rate conversion. The fundamental role of SRC in the DFE was highlighted multiple times in this chapter, and as it can be seen in Figure 1.8 and Figure 1.9, both Tx and Rx DFE architectures contain multiple SRC operation of several types at different points in the architecture. An efficient implementation of these modules is crucial to have a good efficiency of the total system. The rest of the thesis studies the SRC function, with the objective to build low-cost solutions that offer a similar filtering performance relatively to the currently available SRC solutions. Optimizing the implementation of the DFE is also addressed by developing a quantization approach that minimize the hardware complexity, while guaranteeing a given precision constraint. This quantization approach can be applied to all the DFE functions. The rest of this thesis is organized as follow. Chapter 2 explores the different SRC solutions proposed in the literature, and a unified vision of the different structures is developed. The development of novel high efficiency SRC structures based on the Newton structure are developed in Chapter 3 and Chapter 4. After defining the optimal quantization parameters of the developed structures in Chapter 5, the implementation results on both FPGA and ASIC targets are presented in Chapter 6.

Chapter 2

Unified Vision of Sample Rate Conversion FIR Filters

Contents

2.1 Sample Rate Conversion 32
2.1.1 Sampling Theory Basics 32
2.1.2 Sampling Rate Conversion Definition 34
2.1.3 Unified Vision of SRC FIR Filters 37
2.2 U-F-D and Polyphase Filters: Fundamental SRC Solutions . . . 38
2.2.1 Filter Structure Derivation 38
2.2.2 Practical Implementation 40
2.3 Farrow Structure: An Efficient Solution for Arbitrary SRC . . . 41
2.3.1 Filter Structure Derivation 42
2.3.2 Practical Implementation 43
2.4 CIC Filter: A Multiplier Free Solution for Coarse SRC 46
2.4.1 Filter Structure Derivation 46
2.4.2 Practical Implementation 47
2.5 Newton Structure: A Low Cost Solution for Arbitrary SRC . . 49
2.5.1 Newton Structure Derivation 49
2.5.2 Farrow to Newton Transformation 51
2.5.3 Practical Implementation 52
2.6 Comparison and Applications 52
2.6.1 Characteristics Comparison 53
2.6.2 Implementing an SRC Application 54
2.7 Conclusion 55

Manipulating the sampling frequency is a key function in many digital systems, making sample rate conversion (SRC) a ubiquitous process in digital signal processing. SRC is mainly characterized by its factor that defines how much the sampling frequency is to be increased or decreased. Based on this factor, certain SRC structures are more efficient than others. The development of these SRC structures and techniques has been a subject of studies for decades, and new structures are still being proposed nowadays. Indeed, the generalization of SDR design and multi-standard architectures are the source of new challenging issues. The special focus of this work is to investigate SRC solution for IoT applications. In this chapter introducing SRC, we present the solutions based on finite impulse response (FIR) filters, that are the most widely adopted solutions due to their efficiency, simplicity, and linear phase response. This chapter assembles the literature of FIR based SRC and presents

it in a unified way to provide a concise tutorial on how to understand and implement the different available solutions. The SRC operation is first introduced in Section 2.1.1. It is then shown how all the FIR based SRC literature revolves around five principal structures: up-sampling / filtering / down-sampling (UFD) and polyphase in Section 2.2, Farrow structure in Section 2.3, Cascaded-Integrator-Comb (CIC) filter in Section 2.4, and Newton structure in Section 2.5. This chapter derives the five structures as anti-imaging (AI) SRC filters, and shows how to obtain the anti-aliasing (AA) filters version presented in Appendix A. The five structures are then compared in Section 2.6, and it is shown how to select and combine the different structures to implement any SRC operation efficiently.

2.1 Sample Rate Conversion

Before introducing the concept of SRC, Section 2.1.1 explains first the sampling theory, that shows the sampling effects on the signal in both time and frequency domains. With this understanding, Section 2.1.2 then develops the definition of SRC and presents certain theoretical and practical aspects. Finally, the motivation and contribution of this chapter is summed-up in Section 2.1.3.

2.1.1 Sampling Theory Basics

Digital systems are discrete-time systems that process a signal as a series of samples. To obtain these samples, certain criteria need to be observed in order to ensure that the acquired samples represent correctly the continuous time signal. To better understand the effects of sampling, we present the problem using the Fourier transform. The Fourier transform is the operation linking the frequency and time domains of a signal, as depicted in Figure 2.1-a. It is useful to note that for signals of real value (as opposite to complex valued signals), the spectrum is always symmetric such that the frequencies $-f$ and f represent the same frequency component. For a given time signal $x(t)$ having $X(f)$ as its spectrum, the continuous time Fourier transform (FT) and its inverse (IFT) are defined as:

$$X(f) = FT\{x\}(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}dt \Leftrightarrow x(t) = IFT\{X\}(t) = \int_{-\infty}^{+\infty} X(f)e^{j2\pi ft}df \quad (2.1)$$

Sampling: Supposing now that the signal $x(t)$ has been sampled with a sampling period of T_s, which is represented mathematically as multiplying $x(t)$ by the Dirac comb $\delta_{T_s}(t)$. The latter is defined as an infinite number of delta functions periodic by T_s, from which the discrete-time signal $x_d(t)$ is found as:

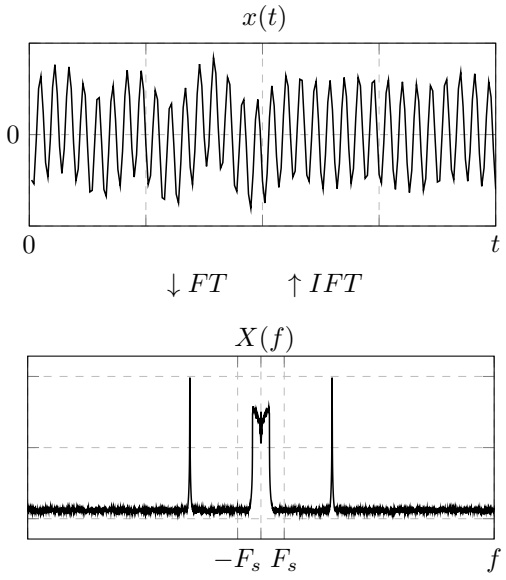
$$\delta_{T_s}(t) \triangleq \sum_{n=-\infty}^{+\infty} \delta(t - nT_s) \quad \Rightarrow \quad x_d(t) = x(t)\delta_{T_s}(t) = \sum_{n=-\infty}^{+\infty} x(nT_s)\delta(t - nT_s) \quad (2.2)$$

The FT of $x_d(t)$ will now result in the discrete-time FT (DTFT):

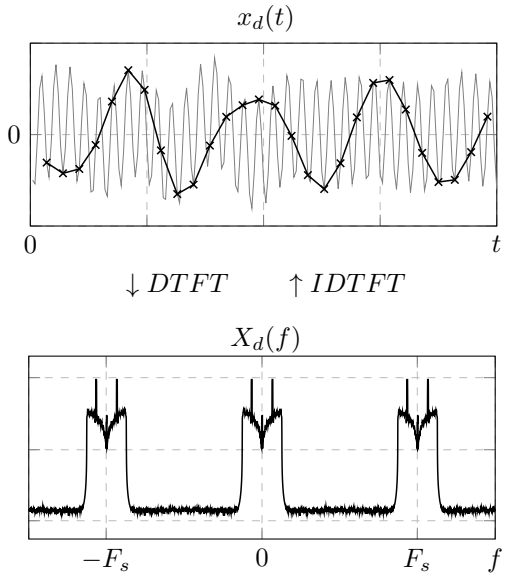
$$X_d(f) = \int_{-\infty}^{+\infty} x_d(t)e^{-j2\pi ft}dt = \sum_{n=-\infty}^{+\infty} x(nT_s)e^{-j2\pi fnT_s} = DTFT\{x\}(f) \quad (2.3)$$

This spectrum is periodic with a period of F_s. Sampling the continuous signal gives rise to the notion of aliasing, illustrated in Figure 2.1-b. Mathematically, the notion of aliasing is found as following:

$$X_d(f) = \left(FT\{x\} * FT\{\delta_{T_s}\}\right)(f) = \left(X * \frac{1}{T_s}\delta_{F_s}\right)(f) = \frac{1}{T_s}\sum_{n=-\infty}^{+\infty} X(f - nF_s) \quad (2.4)$$



(a) Continuous time signal composed of a high frequency sinusoid and a baseband component



(b) Corresponding sampled signal and its periodic aliased spectrum when F_s is inferior to the high frequency sinusoid

Figure 2.1: Time and frequency representations of the continuous and sampled signals

Practically, this expression means that for any interval $[(i)F_s, (i+1)F_s[$ with $i \in \mathbb{R}$, the spectrum of the discrete signal at this interval will contain the original frequency components, plus all the external components that were shifted into this interval by multiples of F_s [Cro83]. Therefore to protect a certain frequency interval from being corrupted by aliases, the only available option is to ensure that all the aliasing components that are shifted by multiples of F_s are equal to zero. The well-known Nyquist criterion handles this by supposing that the continuous signal spectrum is zero beyond a given interval of interest (which is the bandwidth B of the signal). Then to avoid any aliasing, the signal should be sampled with a sampling frequency $F_s \geq 2B$ [Sha49]. If these two conditions are not respected, the aliasing resulting from sampling $x(t)$ is irreversible. This can be seen in Figure 2.1-b, where the high frequency component aliased back into the baseband, since the sampling frequency was not sufficiently large. Finally, the definition of the inverse DTFT (IDTFT) considers only the zero-centered period of the spectrum, where the periodic images do not contain any extra information:

$$x_d(nT_s) = IDTFT\{X\}(nT_s) \triangleq T_s \int_{-\frac{F_s}{2}}^{+\frac{F_s}{2}} X_d(f) e^{j2\pi f n T_s} df \tag{2.5}$$

Where the multiplication by T_s is used to re-normalize the spectrum (c.f. Equation (2.4)).

Reconstruction: If the Nyquist criterion is respected, it is then possible to perfectly reconstruct the continuous signal $x(t)$ using the samples $x_d(t)$. The original spectrum $X(f)$ can be found from the discrete signal spectrum $X_d(f)$ over the support $[-F_s/2, F_s/2[$ as:

$$X(f) = H^{id}_{T_s}(f) X_d(f) \quad \text{with} \quad H^{id}_{T_s}(f) \triangleq \begin{cases} T_s & -F_s/2 < f < F_s/2 \\ 0 & else \end{cases} \tag{2.6}$$

Then the original signal is the result of a filtering operation defined as:

$$x(t) = IFT\{X\}(t) = IFT\{H^{id}_{T_s} X_d\}(t) = \left(IFT\{H^{id}_{T_s}\} \ * \ IFT\{X_d\}\right)(t) = \left(h^{id}_{T_s} * x_d\right)(t) \tag{2.7}$$

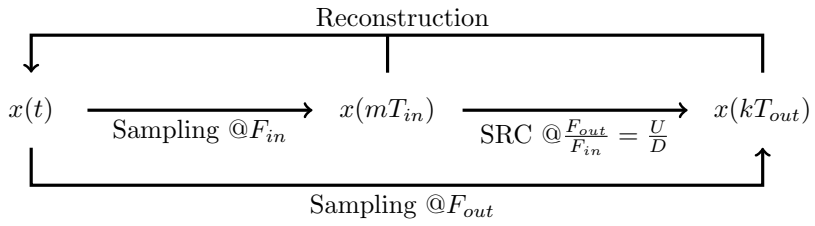


Figure 2.2: Sampling, Sampling Rate Conversion, and Reconstruction

Where we identify the filtering operation by the impulse response $h^{id}_{T_s}(t)$:

$$h^{id}_{T_s}(t) = h^{id}\left(\frac{t}{T_s}\right) = \int_{-\infty}^{+\infty} H^{id}_{T_s}(f)e^{j2\pi ft}df = sinc\left(\frac{t}{T_s}\right) \quad \text{with} \quad sinc(t) = \frac{\sin(\pi t)}{\pi t} \tag{2.8}$$

This filtering operation is known in the literature as the Whittaker-Shannon interpolation formula, that allows the theoretical perfect reconstruction of a band-limited signal [Sha49]. This reconstruction is possible for any sampling frequency F_s respecting the Nyquist criterion, and for any phase delay the sampling may have. However the impulse response $h^{id}_{T_s}(t)$ is non-causal and of infinite order, therefore it cannot be practically implemented. Many approximations were studied in the literature, and are presented later in this chapter.

2.1.2 Sampling Rate Conversion Definition

Sampling rate conversion (SRC) consists of changing the sampling frequency of a signal without going through reconstruction and re-sampling. The definition of SRC and its relation to sampling and reconstruction is shown in Figure 2.2. Sampling $x(t)$ at a frequency F_{in} and F_{out} results in the samples $x(mT_{in})$ and $x(kT_{out})$ respectively. Sampling rate conversion is then defined as the operation that takes the samples $x(mT_{in})$ as input and outputs $x(kT_{out})$, with the change in the sampling frequency defined by the SRC factor R:

$$R = \frac{F_{out}}{F_{in}} = \frac{U}{D} \quad \text{such that} \quad U, D \in \mathbb{N} \tag{2.9}$$

The SRC operation with $R > 1$ is called interpolation, while decimation is used to refer to the case of $R < 1$. The assumption that U and D are integers results from the fact that the SRC operation will be implemented in a digital system, that is quantized and can only handle finite precision. In the literature, integer SRC designates the case for when R is integer. Rational SRC is used to refer to operations having a factor R expressed as a ratio of integers of reasonable value (e.g. 3/5, 480/441) [Cro75][Hsi87][Bre11]. Many references use the designation of arbitrary SRC [Tsu05][Bab05][Leh09][Har97][Bab02][Ram84], or SRC of incommensurate factor [Hen01][Bab05][Laa96][Gar93][Eru93], to identify SRC operations of irrational factors. This is only valid theoretically, and in practical implementations it is only possible to approximate the desired irrational SRC factor up to a certain precision. This approximation is explained later in this chapter in Section 2.3.2.

Multistage SRC: It is often desirable to implement the SRC operation through a multi-stage approach, where the SRC factor R is written as:

$$R = \frac{U}{D} = \frac{U_1}{D_1} \times \frac{U_2}{D_2} \times \ldots \times \frac{U_m}{D_m} \quad \text{such that} \quad (U_k, D_k) \in \mathbb{N}^2 \tag{2.10}$$

Each factor U_k/D_k is implemented as an SRC stage. This is usually done to separate coarse and fine-tuned SRC operations, allowing a more efficient implementation. The multi-stage

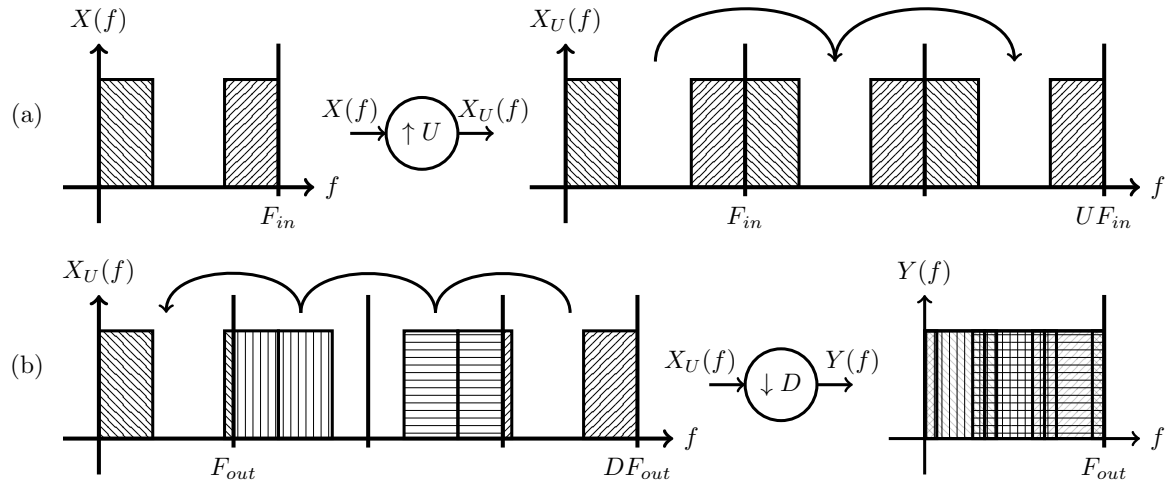


Figure 2.3: Effects of up-sampling (a) and down-sampling (b) on the signal spectrum

apporach is also used to relax the filtering constraints reducing thereby the complexity of the implementation. Also in the case where interpolations or decimation by a factor of two are identified, half-band filters that are presented later offer a significantly efficient implementation. Finally, this separation offers reconfiguration capabilities to the SRC chain, where by bypassing certain stages, different SRC factors can be achieved. Section 2.6.2 at the end of this chapter discusses in more details how to implement multistage SRC chains, and how to select the appropriate filters.

Universal SRC Model: To implement a change in sampling frequency by an SRC factor $R = U/D$, using discrete time signal processing, the most straightforward solution is to increase the sampling frequency by U, followed by a decrease of factor D. Manipulating the sampling rate has diverse effects on the signal spectrum that should be handled correctly for a correct implementation. The universal model of any SRC operation implementation is then developed as follows:

1. **Up-sampling** consists of inserting $U - 1$ zeros between input samples in the time domain. This can be seen in the frequency domain as unfolding the spectrum U times as shown in Figure 2.3-a.

2. **Down-sampling** consists of keeping only one sample among every D samples of the input. Viewed from the frequency domain, it can be seen as cutting the spectrum into D pieces and folding them one on top of the other as shown in Figure Figure 2.3-b. It is shown in this figure the case where the input spectrum of down-sampling contains frequency components above $F_{out}/2$ which results in having a corrupted output spectrum where aliasing has occurred.

3. **Low Pass Filtering** is required to handle the problems of performing down-sampling directly after up-sampling, where the created images will risk aliasing back into base-band, and where the signal itself may have frequency components beyond $F_{out}/2$. This is where the low pass filter (LPF) comes in, shown in Figure Figure 2.4, that we call SRC filter, having the role of removing images and protecting from aliasing.

These three basic elements can be used to represent any SRC operation, even when the practically implemented structure is completely different as it will be seen below. We identify this model as the up-sampling / filtering / down-sampling (U-F-D) SRC model [Cro75]. Ideally

Figure 2.4: Up-sampling / Filtering / Down-Sampling SRC Model

the filter has a cut-off frequency F_c that is the minimum between $F_{in}/2$ and $F_{out}/2$ to completely eliminate all images and protect from all aliasing. The ideal desired response $h[n]$ of the low pass filter is then the ideal reconstruction response developed in Equation (2.8), that corresponds to the desired cut-off frequency depending on the factor R:

$$h\left[n\right] = \begin{cases} sinc\left(n/U\right) & R > 1 \\ sinc\left(n/D\right) & R < 1 \end{cases} \tag{2.11}$$

SRC Using FIR Filters: In literature, we can find a variety of ways and solutions to implement the SRC operation, that can be classified following three alternative methods of signal processing. First one is a straightforward approach that consists in reconstructing the signal to the continuous-time domain and then re-sampling it with the required sample rate. Second one consists in re-sampling the discrete-time signal through an Infinite Impulse Response (IIR) filter [Mil09, Chap. V]. And third one consists in re-sampling the discrete-time signal through a Finite Impulse Response (FIR) filter. The first and second methods are known to introduce inevitable nonlinear distortion effects to the processed signal. Moreover, the implementation of the first method requires an important amount of hardware resources and a couple of very precise digital clock frequencies associated to the input and output signals. The IIR-based SRC methods are afflicted with stability issues and must be handled with care, adding complexity to their implementation. In this thesis, we are interested in the FIR based SRC methods that are stable, due to the construction of the filter having an impulse response that is finite in length. Moreover, FIR filters can be built to have a linear phase response, avoiding any distortion to the signal's phase. And as it will be shown in this chapter, the FIR based SRC structures can be optimized to provide very efficient hardware implementations. And finally, by understanding the FIR based SRC method, it becomes intuitive to understand and to study the two other methods of SRC.

Duality and Transposition: Duality is a property of many signal processing operations, where one operation performs the complementary of the other. For example, modulation and de-modulations are duals, as well as for up-sampling and down-sampling. The transposition theorems were developed in [Cla78], and are also presented in section 5 of chapter 3 in [Cro83]. Simply stated, a structure can be obtained from its dual through transposition, following three steps:

1. Reversing the direction of all signal flow branches
2. Replacing the nodes with summations and vice versa
3. Interchanging the input and output ports

Transposition of a linear time-invariant system (LTI) is the same system itself, while transposition of linear time-variant systems results in their dual. For SRC filters, transposition has the effect of interchanging F_{in} with F_{out}, and U with D. Therefore the design of a decimation operation of factor R can be found from an interpolation SRC implementation through two steps. First, the interpolation SRC operation of factor $R' = 1/R$ is developed. Then transposition on the resulting structure is applied to find the final SRC implementation. In this chapter, the interpolation SRC filters are developed, and their decimation counterparts

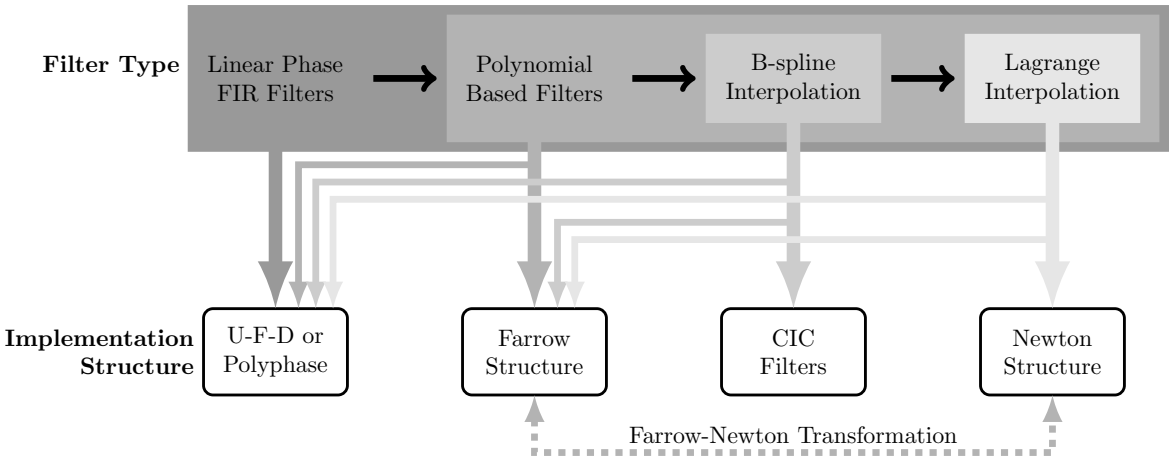


Figure 2.5: The Unified Vision of SRC FIR Filters

can be found through transposition. Appendix A provides further details on this subject, and develops the transposed structures of the filters presented in this chapter.

2.1.3 Unified Vision of SRC FIR Filters

Studies in the area of FIR based SRC techniques examined the different analytic and implementation aspects in independent ways, and usually shortcuts are made that are not adequately justified or explained when the proposed structures are presented. This makes it complicated to learn about these techniques, to understand what is the option better suited for each case of SRC, and to know what kind of filter responses can be achieved by using a certain structure. A very large number of structures have been proposed, however they can all be related to at least one of the five principal FIR-based SRC structures: the U-F-D, Polyphase, Farrow, CIC, and Newton structures. Everything proposed in the FIR based SRC literature revolves around a modification or a combination of these five structures. We offer in this chapter a unified derivation, starting from one common root, to find these structures and to explain clearly their respective correspondences and differences. As shown in Figure 2.5, starting from the linear phase FIR filter type, we proceed by taking more particular filter types, and then by exploiting the characteristics of the impulse response of these filters, we optimize and find the different implementation structures. For the linear phase FIR filters, we present the U-F-D and the polyphase structures. Then we take particular cases of linear phase FIR filters, which are the polynomial based, the B-spline, and the Lagrange interpolation impulse responses. By exploiting the particular properties of each response, we derive the Farrow, CIC, and Newton structure respectively. This presentation offers an easy way to grasp the relations between the different structures, clarifying thereby how to select the most appropriate structure for a certain SRC operation, how to compare the different structures, and how they can be combined together to perform any SRC operation.

We begin by considering the SRC FIR filters of linear phase response. The linearity of the phase response means that the phase delay is constant over all the frequency components, avoiding any distortions to the signal's phase. The amplitude response of the filter may introduce unwanted attenuation to some frequency components, however these can be handled through compensation techniques when necessary (mostly used with CIC filters due to their important droop in the passband). A criterion for the FIR filter to have a linear phase is that the impulse response must be symmetrical or anti-symmetrical [Paq18], which is a property that can be exploited to optimize the implementation structures, as it was done in [Bre11][Ves96][Bab02], and as it will be presented later in this chapter.

2.2 U-F-D and Polyphase Filters: Fundamental SRC Solutions

R. E. Crochiere and L.R. Rabiner were pioneers in the SRC domain, and were among the first to present the general theory of SRC, and to develop the up-sampling/filtering/down-sampling (U-F-D) structure implementation. They also showed how this structure can be further optimized by breaking it into multiple stages in [Cro75; Cro83]. One year later, the application of the polyphase structure for SRC of integer factor was presented by M. G. Bellanger *et al.* in [Bel76]. This work was extended in [Hsi87] to enable rational SRC factors with the polyphase implementations. Some further optimized structures for rational SRC were introduced in [Bre11], that benefit from the coefficient symmetry of linear phase FIR filters to reduce the hardware complexity. This section starts with a simple theoretical derivation of these filter structures in Section 2.2.1, it then develops the important points to be considered for their practical implementation in Section 2.2.2.

2.2.1 Filter Structure Derivation

The U-F-D structure is a direct implementation of the theoretical model presented in Figure 2.4. It is theoretically capable of implementing any SRC operation using any linear phase FIR filtering response, however practically it does not offer an efficient implementation. The polyphase structure improves on the U-F-D one, while allowing the implementation of any linear phase FIR filter response. The derivation of these two structures can be obtained as presented below.

U-F-D Structure: In this case, the impulse response is directly implemented as an FIR low pass filter as shown in Figure 2.4. Since the input of the filter $x_U[n]$ is the output of the up-sampling of $x[m]$, then we know that for each U input samples to the filter, only one is non-zero. Therefore if we require a filtering operation of order $N-1$ (meaning to use N inputs to calculate each output), the number of taps of the FIR filter has to be equal to $N \times U$. The transfer function of this filter is written as:

$$H(z) = \mathcal{Z}\{h[n]\} = \sum_{n=-\infty}^{+\infty} h[n]z^{-n} = \sum_{n=0}^{NU-1} h[n]z^{-n} \tag{2.12}$$

Then the output of the filter $X_{UF}(z)$ is found as:

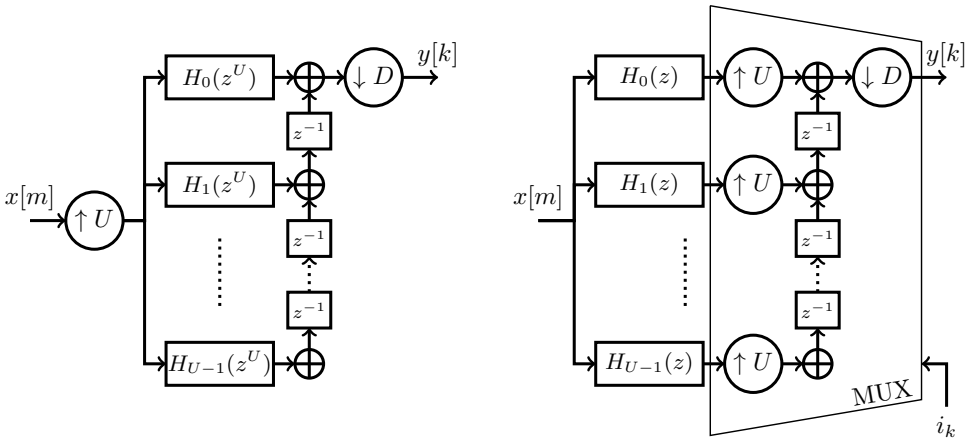
$$X_{UF}(z) = H(z)X_U(z) = H(z)X(z^U) \tag{2.13}$$

Since $X_U(z) = X(z^U)$ that is found by applying the Z-transform on $x_U[n]$. In many applications we require a finely-tuned SRC factor, where U and D will get large in value. The low pass filter in the U-F-D structure operates at the sampling frequency of $F_U = UF_{in}$. Having large values of U and D results in two problems that lead to the inefficiency of this structure:

1. The filter operates at a very high frequency UF_{in}, making the implementation inefficient, and possibly non realizable when the digital circuit has operating frequency limits.
2. A great number of the filter's output samples will be dropped by the following down-sampling block of factor D, meaning that the filter is inefficient since it calculates unneeded samples.

Polyphase Structure: This structure addresses the two disadvantages mentioned above. The classical polyphase structure is deduced from the U-F-D structure by rearranging the filter coefficients, through re-writing $X_{UF}(z)$ in Equation (2.13) as:

$$X_{UF}(z) = \left[\sum_{n=0}^{NU-1} h[n]z^{-n}\right] X(z^U) = \left[\sum_{i=0}^{U-1} z^{-i}H_i\left(z^U\right)\right] X\left(z^U\right) \tag{2.14}$$



(a) Polyphase Filter Structure (b) Polyphase SRC Using the Noble Identity

Figure 2.6: The Polyphase Structure before and after Applying the Noble Identity

In this expression, the FIR filter is broken into U filters $H_i(z)$ of order N in parallel, with each filter containing the coefficients used to find a certain output:

$$H_i(z) = \sum_{j=0}^{N-1} h[i + jU]z^{-j} \tag{2.15}$$

The matrix representation of $H(z)$ offers a clear description of the filter structure:

$$\underset{U\times 1}{H(z)} = \underset{U\times N}{H} \times \underset{N\times 1}{Z(z)} = \begin{bmatrix} h_0 & h_{0+U} & \dots & h_{0+(N-1)U} \\ h_1 & h_{1+U} & \dots & h_{1+(N-1)U} \\ \vdots & \vdots & \ddots & \vdots \\ h_{U-1} & h_{2U-1} & \dots & h_{NU-1} \end{bmatrix} \times \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N+1} \end{bmatrix} \tag{2.16}$$

With $h_k = h[k]$. Each row of the matrix H contains the coefficients of the filter $H_i(z)$. This shows an equivalent implementation to the U-F-D structure, known as the polyphase structure, that is realized through U filters $H_i(z)$ in parallel with a delay element z^{-i} in series with each filter i, and then the output of all the branches in parallel summed together, as shown in [Bel76][Hsi87]. This structure is illustrated in Figure 2.6-a.

However not all outputs of the polyphase filter are required, where the down-sampling by D eliminates many calculated samples. In the case of finely tuned SRC factors, this problem can get very critical when $U \approx D$, and approximately $U - 1$ samples of U calculated ones will be dropped. An optimization is possible by benefiting from the second Noble identity, stating that a delay line of length z^U operating at a frequency UF_s is equivalent to a single delay register z operating at F_s. This can be applied to the filter expression in Equation (2.14), by pulling the filters before the up-sampling operation, resulting in:

$$H_i(z^U)X(z^U) \text{ operating at } UF_{in} \quad \Rightarrow \quad H_i(z)X(z) \text{ operating at } F_{in} \tag{2.17}$$

Applying this property on the polyphase structure results in the SRC architecture illustrated in Figure 2.6-b. At the output of the filters, the network containing the combination of the up-sampling, delay and down-sampling blocks behaves like a commutator with increment steps D, that selects the appropriate filter i_k corresponding to the output instant k. It is then possible to replace this network by a multiplexer combined with a controller operating at the output sampling frequency F_{out} that updates the value of i_k for each output. This results in the polyphase structure shown in [Har97], that operates using the sampling frequencies F_{in} and F_{out} only, without the need of the UF_{in} sampling domain, that can be problematic in the

case of practical implementation. However when U is large, the structure still requires having U branches in parallel, which is still a problem concerning the efficiency of the structure. This can be solved by using polynomial based filters (PBF) presented in the next Section 2.3.

2.2.2 Practical Implementation

To implement the derived filter structures, certain practical aspects need to be addressed first. In this section, the hardware implementation aspects are first presented. Then the SRC controller algorithm is developed. The SRC factor quantization is then discussed. Finally, the half-band filters that allow very efficient implementations for SRC factors of two and half are introduced.

Hardware Implementation: The implementation of the derived structures in hardware requires the definition of a number of design preferences. First, any hardware implementation is qualified by a number of criteria, such as the operation speed, the calculation precision, or the hardware complexity. Each implementation scenario requires a compromise between these criteria, and for each scenario we find a certain implementation strategy and approach that is more efficient than the others. These hardware implementation aspects of this PhD work are developed in Chapter 6. The second design preference concerns the quantization of the implementation. Quantization is the operation of defining the number of bits used to represent the filter coefficients and signals in the hardware implementation. Coefficients quantization is a deterministic problem, where limiting the precision of the filter coefficients will result in a well-defined modification of the filter response. Many approaches are developed in the literature to quantize a digital filter coefficients [Cro75][Nie89][Won91][Jia02][Qia06]. The second part of quantization concerns the signals being processed, in which case the resulting error is random. This quantization task is more complicated and it is addressed in depth in Chapter 5, where an optimal signal quantization method is proposed.

SRC Controller: It is often supposed in the literature that for the implementation of finely-tuned SRC, F_{in} and F_{out} are actually available digital clocks. However we often do not have the exact two clock frequencies at the input and output, where the system operates using one given digital clock. The multi-rate operation is then made possible through an SRC controller, and by adding sample buffers at the input and output. These buffers are required because when dealing with SRC, the number of inputs to the filter is not necessarily equal to the number of outputs. For $R > 1$, we are increasing the sample rate, and therefore there will be more output samples than input ones. The opposite is true for $R < 1$. It is also important that the digital clock is faster than both F_{in} and F_{out} in order to avoid saturating the buffers. For a given output instant k, two indexes are needed to control the state of the polyphase filter. The first being m_k that indicates the inputs used by the filters, where every filter $H_i(z)$ has the inputs $\{x[m_k], x[m_k - 1], \ldots, x[m_k - N + 1]\}$ in its delay line. The other index is i_k that specifies the branch of the polyphase filter that calculates the output $y[k]$. These two indexes are managed by the SRC controller to implement the desired SRC operation defined by U and D. These two indexes can be found recursively as following:

$$\begin{cases} m_k - m_0 = \lfloor k/R \rfloor & \Rightarrow m_{k+1} = m_k + \left\lfloor \frac{i_k + D}{U} \right\rfloor \\ i_k - i_0 = kD \mod (U) & \Rightarrow i_{k+1} = [i_k + D] \mod (U) \end{cases} \tag{2.18}$$

With m_0 and i_0 being initialization values. The variable i_k is simply the result of an overflowing accumulator by D. This can be translated into the following Algorithm 1 that describes the AI SRC controller operation. It can be noticed in this algorithm that for $R > 1 \Rightarrow U > D$, it is possible sometimes to skip the while loop at line 4, meaning that the same inputs will be used to find multiple outputs. While for $R < 1 \Rightarrow U < D$, it is possible to stay in the while

Algorithm 1 Algorithm of the SRC Controller

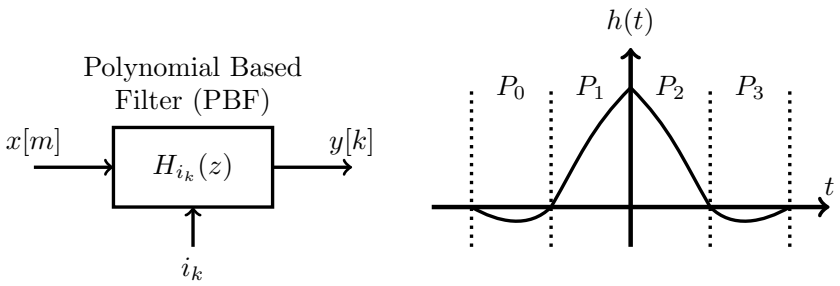
Input: U, D
Output: m_k, i_k, k
1: $m_k = 0$, $i_k = 0$, $k = 0$
2: **while** True **do**
3: $i_k = i_k + D$
4: **while** $i_k \geq U$ **do**
5: m_k++ {Get next input from $FIFO_x$}
6: $i_k = i_k - U$
7: **end while**
8: $k++$ {Calculate output k and insert in $FIFO_y$}
9: **end while**

loop at line 4 for multiple iterations, meaning that between two outputs, two or more inputs may be necessary.

Half-Band filters: When performing SRC of large factors, it is often desirable to break the U-F-D structure into multiple stages as it was proposed in [Cro75], that offers important improvements in efficiency. This is also advantageous when it is possible to divide the SRC factor into factors of two. In this case, we know that we will have two set of filter coefficients which are samples of the impulse response distant by $T_{in}/2$. Then the FIR filter design tries to approximate the ideal $sinc(t/T_{in})$ response, in a way that the value of the impulse response is zero at multiples of T_{in}. Doing this results in the first set of coefficients corresponding to zero fractional delay, with this set containing only one non-zero element, while the second set consists of the impulse response sampled at $(k + \frac{1}{2})T_{in}$ such that $k \in \mathbb{Z}$. Therefore for an interpolation by a factor of two, the SRC filter can be implemented with 25% of the coefficients, where the fact that half of the coefficients are zeros offer the first reduction by 50%, and then the symmetry of the impulse response is the source of a further reduction by 50%. This implementation approach is known as the half-band filter [Cro83, p.155].

2.3 Farrow Structure: An Efficient Solution for Arbitrary SRC

An efficient solution for finely-tuned SRC implementation was the structure developed by C. W. Farrow in [Far88]. This structure implements polynomial based filters, and is known in the literature under many names, most notably as the Farrow structure, variable digital filter (VDF), or variable fractional delay filter (V-FDF) [Laa96]. Before this structure, efficient SRC implementations were limited to SRC operations with simple rational factors. The introduction of this structure enabled implementing arbitrary SRC factors, allowing high precision finely-tuned SRC implementations far more efficiently than it was possible with the previously existing structures. This structure is used as a main block of many SRC systems, where we give for example the audio SRC system proposed in [Raj00] and the complex-valued SRC system in [Tsu05], that combine the Farrow structure with the U-F-D structure to build the complete SRC systems. A modification of the structure was developed in [Ves96], that offers the possibility to benefit from the symmetry of the coefficients due to the symmetrical impulse response. The transposed Farrow structure (TFS) presented by T. Hentschel and G. Fettweis in [Hen01], has the advantage of having a better performance for the case of decimation. The TFS can also benefit from the symmetry of the coefficients as discussed in [Bab02]. The transposed structure is related to the original Farrow structure through concepts of networks transposition presented in Section 2.1.2. It was shown in [Hen00] how this transposed structure can be used to implement a combined filter for SRC, matched filtering,



(a) Polynomial Based Filter Structure

(b) Implulse Response Constructed Using Polynomial Pieces

Figure 2.7: Polynomial Based Filter Structure and its Impulse Response

and symbol synchronization. Another application is in [Bab05], where the Farrow structure and its transpose were used in combination of the CIC filter structure to implement systems offering both anti-imaging and anti-aliasing capabilities simultaneously. Other applications for the transposed structure were proposed in [Bab04] and [Li04]. This section derives the Farrow structure as an implementation of polynomial based filters in Section 2.3.1, and it shows its relation to the developed polyphase structure in Section 2.2.1. Section 2.3.2 develops then the practical implementation concerns for the Farrow structure, most notably the quantization of its fractional delay parameter.

2.3.1 Filter Structure Derivation

It was shown in Section 2.2.1 that for each output of the polyphase structure, only one filter is selected by the multiplexer. Therefore it would be interesting that instead of having U filters in parallel, to have only one filter with its coefficients being calculated dynamically for each output using the index i_k. This can be made possible by defining the impulse response as being constructed using N polynomial pieces $P_j(\mu)$ with $j \in \{0, 1, \ldots, N-1\}$, as shown in Figure 2.7 for $N = 4$. Each polynomial piece is of order $M-1$ and is defined as:

$$P_j(\mu) = \sum_{l=0}^{M-1} c_l(j)\mu^l \tag{2.19}$$

Then P_j is used to calculate the filter's coefficient $h_{i_k}[j]$ corresponding to the output $y[k]$. The variable μ is called the fractional delay in the literature, since it corresponds to the delay between the output sample and its corresponding input sample, that is a fractional of the sampling period.

Polynomial Based Filter: Using Equation (2.19), it is possible to rewrite the filter expression in Equation (2.15) as:

$$H_{i_k}(z) = \sum_{j=0}^{N-1} h_{i_k}[j]z^{-j} = \sum_{j=0}^{N-1} \left[\sum_{l=0}^{M-1} c_l(j)\mu(i_k)^l \right] z^{-j} \tag{2.20}$$

With $\mu(i_k)$ indicating the position of the coefficients of the filter i_k on the different polynomial pieces. Since we are dealing with linear phase FIR, then we know that the impulse response is symmetric, and therefore the polynomial pieces will be symmetric with respect to the Y-axis as well. This was not exploited in the classical Farrow structures proposed in [Far88; Eru93]. The structures that did exploit this property later came to be known as the modified Farrow structure [Ves96; Bab02]. To benefit from this symmetry, the polynomials should have

their support centered at zero. This is why we define the variable $\mu(i_k)$ used to develop the polynomials as:

$$\mu(i_k) = \frac{i_k}{U} - \frac{1}{2} \in [-0.5; 0.5[\tag{2.21}$$

Through this, the polynomials are developed to be symmetric such that:

$$P_j(\mu) = P_{N-j-1}(-\mu) \tag{2.22}$$

This means that the coefficients of the even monomials of P_j and P_{N-j-1} are equal, and the odd monomials coefficients are opposite in signs. Using this type of impulse response, the matrix representation of the filter transfer function $H(z)$ in Equation (2.16) is written as:

$$\begin{aligned} \underset{U\times 1}{H(z)} &= \left(\underset{U\times N}{H}\right) \times \underset{N\times 1}{Z(z)} = \left(\underset{U\times M}{\boldsymbol{\mu}} \times \underset{M\times N}{P}\right) \times \underset{N\times 1}{Z(z)} \\ &= \begin{bmatrix} 1 & \mu_1 & \mu_1^2 & \dots & \mu_1^{M-1} \\ 1 & \mu_2 & \mu_2^2 & \dots & \mu_2^{M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mu_U & \mu_U^2 & \dots & \mu_U^{M-1} \end{bmatrix} \begin{bmatrix} c_1(0) & c_2(0) & \dots & c_N(0) \\ c_1(1) & c_2(1) & \dots & c_N(1) \\ \vdots & \vdots & \ddots & \vdots \\ c_1(\text{M-1}) & c_2(\text{M-1}) & \dots & c_N(\text{M-1}) \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N+1} \end{bmatrix} \end{aligned} \tag{2.23}$$

In Equation (2.23), we have $\mu_k = \mu(k-1)$. Then the transfer function of the filter $H_{i_k}(z)$ can be expressed in matrix form using only the row k of the matrix $\boldsymbol{\mu}$:

$$H_{i_k}(z) = \underset{1\times 1}{H(z, \mu_k)} = \underset{1\times M}{\boldsymbol{\mu}_k^T} \times \underset{M\times N}{P} \times \underset{N\times 1}{Z(z)} \tag{2.24}$$

Farrow Structure: Through a factorization by μ in Equation (2.20), we can find the definition of the well-known Farrow structure [Far88] given as:

$$H_{i_k}(z) = \sum_{l=0}^{M-1} G_l(z)\mu(i_k)^l \quad \text{where} \quad G_l(z) = \sum_{j=0}^{N-1} c_l(j) z^{-j} \tag{2.25}$$

The implementation of this structure is shown in Figure 2.8, which consists of M FIR filters $G_l(z)$, each of N taps. The outputs of the G_l filters are then evaluated following the Horner's scheme for polynomial evaluation with $\mu(i_k)$. In this structure the coefficients of the filters G_l are constant and symmetrical, allowing an efficient implementation that uses only half of the multipliers. Moreover, having constant coefficients multipliers provides further possible implementation optimizations. The *H&S* blocks are hold and sample blocks, that represent the interface between the different sampling rate domains. In some cases, the same inputs may be held and used to find multiple outputs. These blocks must be necessarily taken into consideration when applying the transposition techniques to find the transposed Farrow structure.

2.3.2 Practical Implementation

The Farrow structure implementation takes into consideration the same points presented for the U-F-D and polyphase structures implementations in Section 2.2.2. When using only one clock domain, the same SRC controller and configuration is required, the only difference is the added circuit needed to find the value of $\mu(i_k)$ as shown in Equation (2.21). The Farrow structure quantization is carried with the same considerations presented in the previous section, however with the added complexity of quantizing the fractional delay μ. This section first presents how to define the polynomial coefficients that construct the impulse response, and it then develops the fractional delay quantization effects to better understand how to select the appropriate number of bits to represent the variable μ.

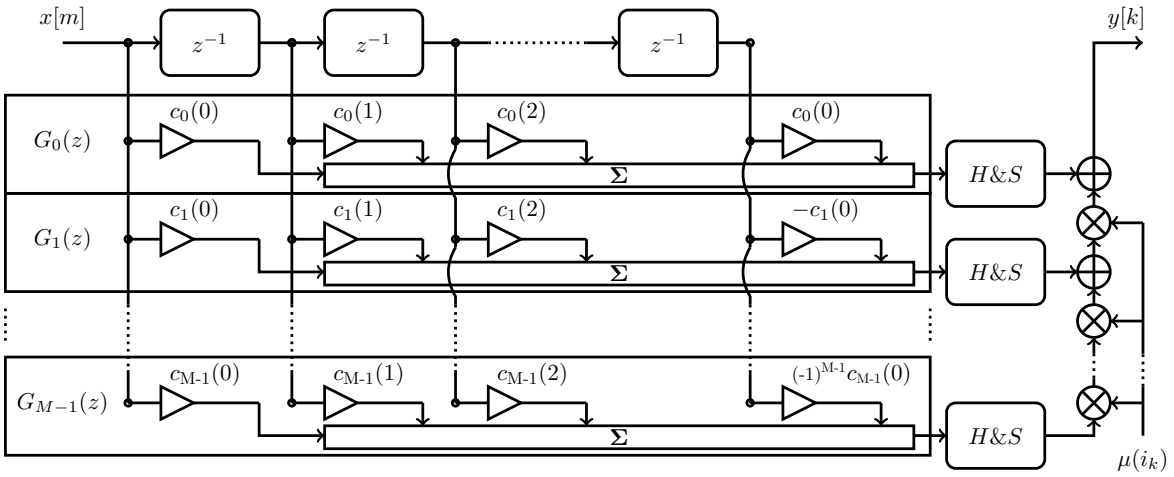


Figure 2.8: The Farrow structure consisting of M filters $G_l(z)$ of N taps each

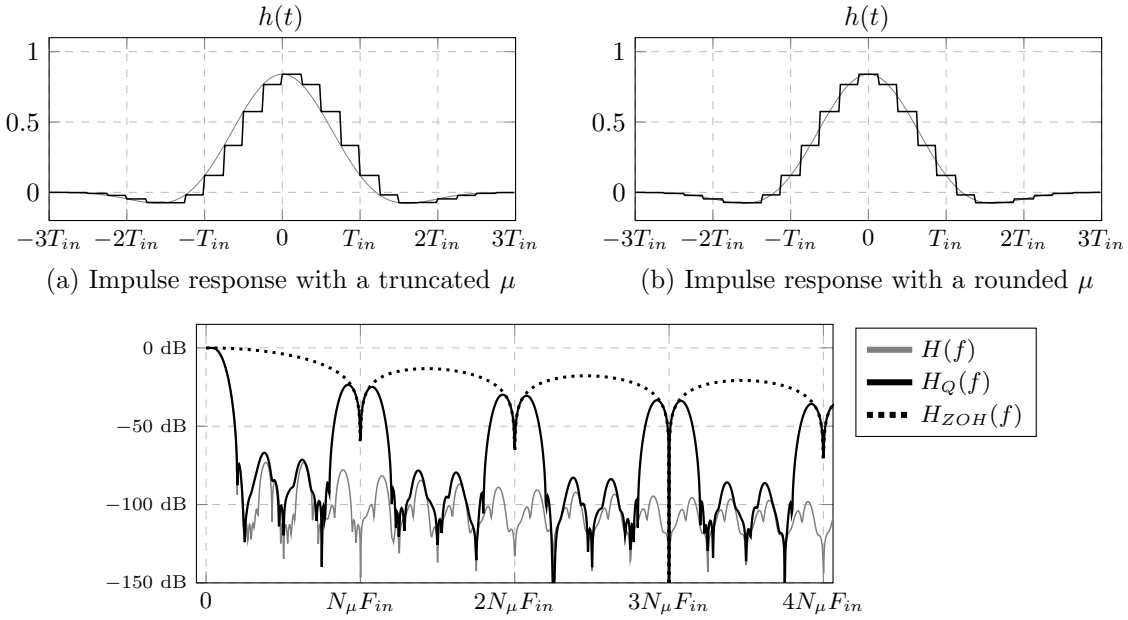
Polynomial Coefficients: The Farrow structure has $M \times N$ degrees of freedom to define the filter response, that are the coefficients $c_l(j)$. When the response is limited to that of linear phase filters, the degrees of freedom are divided by two due to the symmetry of the response, and therefore the polynomials and their coefficients. These coefficients can be found generally in two ways: using well-known polynomial interpolation methods, or applying filter optimization techniques.

In the domain of polynomial interpolation, there are three types that are primarily used. Due to their particular mathematical definition, these interpolation types can be implemented more efficiently using other structures that the Farrow structure as it will be developed later in this thesis. The first well-known type is the B-spline interpolation explained in [Uns99], and implemented in [Doo99] and [Lam16]. The algorithm developed in [Mil10] allows finding the coefficients of the cascaded polynomial pieces, then each piece should be centered around zero before being used with the Farrow structure. B-spline interpolation is detailed in Section 2.4 to find the CIC filter structure. The second type is Lagrange interpolation, which has the advantage of having a maximally flat passband [Her92]. The Farrow structure implementation of Lagrange interpolation is developed in [Eru93][Val95][Den07]. Section 2.5 studies this type of interpolation that results in the Newton structure implementation. The third type is Hermite interpolation, which generalizes Lagrange interpolation by applying equality constraints to samples derivatives as well as to their values. The implementation using the Farrow structure is studied in [Soo11] and [Tse12]. Hermite interpolation is used in Chapter 4 to develop new efficient implementation structures. The work of [Mei02] presents a concise overview of the interpolation methods in the literature, and their use for signal reconstruction.

To correctly profit from the Farrow structure filtering performance, the polynomial coefficients should be defined using filter optimization methods. In the literature we find different techniques, such as the least-square methods applied in [Far88; Lu99], the minmax optimization presented in [Ves97], or the optimization applied separately to each sub-filter as developed in [Joh03]. The thesis of M. T. Hunter [Hun08] develops in detail the Farrow structure optimization problem, and provides a software implementation of the developed optimization algorithm.

Fractional Delay Quantization: Quantization of the fractional delay $\mu(i_k)$ affects two implementation aspects: the output sampling rate precision, and the filtering performance. This subject has been addressed in different works [Ves00][Lóp02][Hun08]. The fractional delay quantization solutions are summed-up below.

For any SRC operation, a certain output sampling frequency F_{out} is desired. In hardware,



(a) Impulse response with a truncated μ

(b) Impulse response with a rounded μ

(c) ZOH sampling effects on the frequency response

Figure 2.9: Fractional delay quantization effects on the impulse and frequency responses

the division by U in Equation (2.21) is implemented as a multiplication by a factor of $1/U$, and it is the quantization of this factor that defines the precision of the SRC operation. The approximation error of $1/U$ results in a jitter on the output sampling period. Supposing that the quantization of $1/U$ is fixed-point with a fractional part of $F_{(1/U)}$ bits. For a certain given tolerated error at the output sampling frequency ΔF_{out}, it is possible to find the corresponding quantization of $1/U$ as:

$$\Delta F_{out} = \frac{F_{in}}{2^{F_{(1/U)}}} \quad \Rightarrow \quad F_{(1/U)} = \left\lceil \log_2 \left(\frac{F_{in}}{\Delta F_{out}} \right) \right\rceil \tag{2.26}$$

Parts-Per-Million (ppm) is a more useful normalized measure of this error for designs that will use multiple sampling frequencies. Depending on the application, the quantization for a given tolerated *ppm* is found similarly as:

$$\Delta F_{out} = \frac{1}{2^{F_{(1/U)}}} \quad \Rightarrow \quad F_{(1/U)} = - \left\lceil \log_2 \left(ppm \right) \right\rceil \tag{2.27}$$

Attention should be given that $F_{(1/U)}$ is at least equal to or larger than F_U, which is the bitwidth of U. Since if this is not respected, the value of $\mu(i_k)$ will be always stuck at zero.

The value of $\mu(i_k)$ resulting from Equation (2.21) is usually transmitted to the V-FDF with a different quantization bit width, that is usually smaller in the V-FDF than in the controller. This allows a reduction in the V-FDF implementation complexity. The primary effect of this second quantization is on the impulse response of the V-FDF, and thereby its filtering response. As shown in Figure 2.9, truncation and rounding of the fractional delay result in a zero-order-hold (ZOH) sampled impulse response. If F_μ bits are used to quantize the fractional part of μ, each polynomial piece of the impulse response will have $N_\mu = 2^{F_\mu}$ samples. The difference between truncation and rounding is that the former results in a constant phase shift of $T_{in}/2N_\mu$, that may be tolerated or corrected in some implementations. The effects of ZOH sampling are shown in Figure 2.9-c. Sampling results in aliasing images in the filter frequency response $H_Q(f)$, that will appear on multiple of $N_\mu F_{in}$. While the ZOH effect will modify the original frequency response $H(f)$ by introducing the effect of a ZOH filter $H_{ZOH}(f)$. If the useful signal band B is known, the aliasing images maximum level

I_{max} can be defined as:

$$I_{max} = sinc\left(\frac{N_\mu - \frac{B}{F_{in}}}{N_\mu}\right) \quad \Rightarrow \quad I_{max}(dB) = 20\log_{10}\left(\frac{B}{F_{in}}\right) - 6.02F_\mu \tag{2.28}$$

For a given tolerated level of aliasing images, the quantization of the fractional delay used by the V-FDF is then deduced as:

$$F_\mu = \left\lceil \frac{1}{6.02}\left[20\log_{10}\left(\frac{B}{F_{in}}\right) - I_{max}(dB)\right]\right\rceil \tag{2.29}$$

2.4 CIC Filter: A Multiplier Free Solution for Coarse SRC

In 1981, E. B. Hogenauer presented the cascaded-integrator-comb (CIC) filters [Hog81] which are a class of SRC FIR filter with an implementation that doesn't require multipliers. These filters have a slightly modified U-F-D structure resulting from the application of the Noble identities presented in Section 2.2.1. Their advantage lies in their low implementation complexity and their reconfigurability in terms of order and SRC factor. However these filters possess two limitations. First, their practicality is limited to implementing coarse rational SRC operations. And second, they are limited to implementing a B-spline interpolation response, that suffers from a high droop in the passband. A solution to improve the frequency response was proposed in [Kwe97] that uses filter sharpening techniques resulting in the sharpened CIC filter structure. Another SRC structure based on B-spline interpolation is the one proposed in [Hua12], that is developed as a polynomial based filter, however the difference from the Farrow structure is in that the polynomial pieces can be taken longer than the sampling period. This is done to allow defining the positions of the zeros of the frequency response to reject any unwanted signals in the received spectrum. In this section, the CIC filter structure is developed first in Section 2.4.1, and then the practical implementation aspects are presented in Section 2.4.2.

2.4.1 Filter Structure Derivation

The Farrow structure presented in Section 2.3 offers an SRC solution with a high degree of freedom permitting the implementation of different filter responses. In this section, we consider the B-spline interpolation method, which corresponds to a set of fixed coefficients defining the polynomial based filter.

B-spline Interpolation: Considering the most simple case of a polynomial filter, where the impulse response consists of only one polynomial piece ($N = 1$) of degree zero ($M = 1$), having a value of one. This corresponds to the Pi or rectangular function normalized to T_{in} noted as $\Pi(t/T_{in})$. Used for interpolation, the Pi function results in a zero-order-hold (ZOH) reconstruction. An interesting property of the Pi function is its frequency response $H_\Pi(f)$, that is found through the Fourier transform as:

$$\Pi\left(\frac{t}{T_{in}}\right) \quad \rightarrow \quad H_\Pi(f) = sinc\left(\frac{f}{F_{in}}\right) \tag{2.30}$$

This frequency response is of interest since its zeros are located at multiples of F_{in}, which is the optimal zero position for images suppression, since the up-sampling images are located on multiples of F_{in} as shown in Figure Figure 2.3-a. The filtering performance of a single Pi function may not be sufficient, however it can be improved by cascading N filters. This results in the impulse response of the B-spline interpolation of order $N-1$:

$$\beta^N(t) = (\underbrace{\Pi * \Pi * \ldots * \Pi}_{N})(t) \tag{2.31}$$

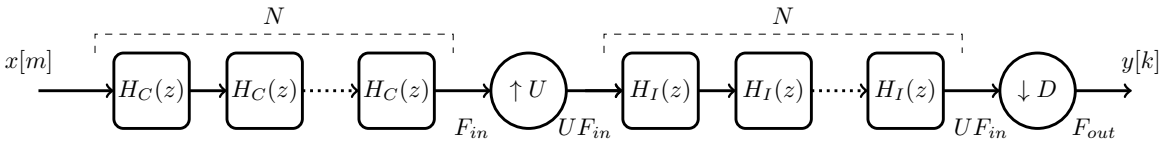


Figure 2.10: The interpolation cascaded-integrator-comb (CIC) filter structure of order N

The Fourier transform of convolution in the time domain results in multiplication in the frequency domain. Therefore the frequency response $\boldsymbol{B}^N(f)$ of $\beta^N(t)$ is found as:

$$\boldsymbol{B}^N(f) = sinc\left(\frac{f}{F_{in}}\right)^N \tag{2.32}$$

The B-spline function $\beta^N(t)$ is piecewise polynomial with N pieces of degree $N-1$ each [Uns99]. Therefore this interpolation can be implemented using the Farrow structure as it was developed in [Doo99]. However by exploiting the particularity of the impulse response definition as a cascade of rectangular functions, a more efficient implementation structure is found.

CIC Filter Structure: It was presented in Section 2.2.1, that for an SRC filter of order $N-1$ we need NU filter taps. In the case of the rectangular function, we have U coefficients taken from the causal rectangular function (defined for $t > 0$), with all coefficients having a value of one, which can be expressed through the Z transform as:

$$H(z) = \sum_{n=0}^{U-1} z^{-1} = \left(1 - z^{-U}\right)\left(\frac{1}{1 - z^{-1}}\right) = H_{C_U}(z)H_I(z) \tag{2.33}$$

This is the non-normalized moving average filter, which can be implemented in a recursive running-sum structure, consisting of two construction blocks, the integrator having the transfer function $H_I(z)$ and the comb with a delay z^{-U} having the transfer function $H_{C_U}(z)$ [Lyo05]. Then by cascading N rectangular function filters, the B-spline interpolation of order $N-1$ is implemented. We know that the comb and the integrator are linear time-invariant (LTI) systems, therefore their order of placement relative to one another in the structure can be exchanged. Then benefiting from the second Noble identity, the comb blocks can be placed before the up-sampling operation optimizing the implementation by reducing the delay line of the comb blocks z^{-U} to only one delay element z^{-1}. This results in the structure presented by Hagenauer in [Hog81], and shown in Figure 2.10, called the interpolation cascaded-integrator-comb (CIC) filter, where a CIC filter of order N having N comb and integrator blocks, implements the B-spline interpolation of order $N-1$.

2.4.2 Practical Implementation

In the case of CIC filters, there is no need for an SRC controller since the filter implements a U-F-D structure. However the same hardware implementation considerations discussed in Section 2.2.2 are used for the CIC filters as well. This section develops certain particular implementation aspects for CIC filters, such as output re-scaling, filter order choice, frequency response compensation, and non-recursive implementations.

Output Re-scaling: The CIC filter is multiplier free because it does not normalize its filtering operation, resulting in a large gain at its output. In the case of interpolation CIC filter, this gain is found as $G_{CIC} = U^{N-1}$. Certain implementations may require that the output of the CIC filter to be normalized to its input, requiring thereby an output re-scaling. This is

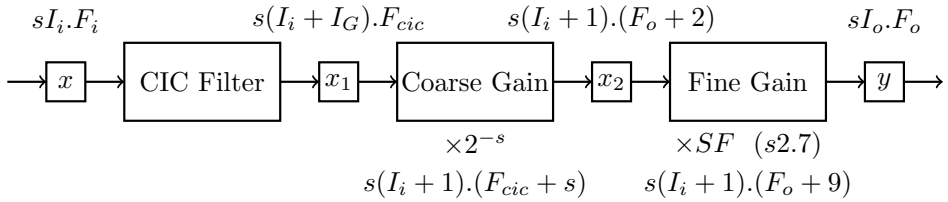


Figure 2.11: Normalization of the CIC output gain through coarse and fine adjustments

done in two steps: coarse and fine gain adjustments. We propose the approach illustrated in Figure 2.11. Using the fixed-point quantization scheme presented in Chapter 5, we consider the filter input is quantized on $sI_i.F_i$ bits. After performing the optimal quantization method developed in the same chapter, the output is quantized on $s(I_i + I_G).F_{cic}$. The added integer part bits $I_G = \lceil N \log_2(U) \rceil$ are needed due to the filter gain. The coarse gain adjustment consists then of normalizing the important integer gain, by right shifting the quantization by $s = \lfloor \log_2(G_{CIC}) \rceil$, and rounding the result to $s(I_i + 1).(F_o + 2)$, where $sI_o.F_o$ is the desired output quantization. Fine gain adjustment then finalizes the normalization by multiplying the samples with the correction factor $SF = 2^s / G_{CIC}$ that is quantized on $s2.7$ bits. The output of this multiplication is then rounded to $sI_o.F_o$.

Selecting the Filter Order: Three measures may be used to select the appropriate CIC filter order for a given SRC operation. A general measure is that of the first side lobe attenuation level, which is equal to $13.64 \times N$ dB with N being the filter order. This estimation is only valid for an up-sampling or down-sampling factor larger than eight in the case of interpolation or decimation CIC filters respectively. The attenuation level gets smaller as this factor goes below eight. Two other specific measures that depend on the useful signal band B are images attenuation level, and the passband droop. The images attenuation level measures the weakest attenuation of the signal images, while the passband droop measures the strongest attenuation in the signal band. These measures can be calculated using the filter frequency response expression, however for simplifying the design process, Hogenauer proposed in the original article two tables that resume these measures for certain example cases [Hog81]. The choice of the CIC filter order is a delicate task that compromises between the images attenuation level and the passband droop. However the former is more important than the latter, since aliasing errors due to weak images attenuation cannot be corrected afterwards, while it is possible to compensate the passband droop as explained below.

CIC Compensation Filter: The frequency response of B-spline interpolation is known to have an important droop in its passband. However it is possible to improve this response by using a compensation filter that aims to correct the droop of the CIC filter. Then the combined frequency responses of the CIC and its compensation filter has a more flat passband. The most straight forward way to build a compensation filter is through implementing an FIR filter approximating the inverse of the B-spline response over the desired passband [Alt07c]. Other methods for designing the CIC compensation filter rely on the maximally flat criterion, where the compensation filter coefficients are found through a linear equation system [Mol11][Fer12]. Multiplier free CIC compensation filters for both narrow-band and wide-band signals were developed and presented in [Fer12][Dol08][Dol09]. This compensation is usually integrated in the FIR filter that follows the CIC filter for a more efficient implementation (see the DFE architecture in Chapter 1).

Non-Recursive CIC Filter Structures: Another downside of the classical CIC filter is the recursive structure of the integrator blocks, that limits the maximum speed of the

hardware implementation. While the direct implementation of the structure presented in Figure 2.10 may meet the speed requirements for certain SRC operations, other applications may not be possible with a recursive CIC structure. A non-recursive structure was proposed in [Gao00] offering lower power consumption and higher operating speeds compared to the recursive CIC structure. Further improvements were achieved in [Sha07] through a polyphase non-recursive CIC implementation.

2.5 Newton Structure: A Low Cost Solution for Arbitrary SRC

The Newton interpolation structure presented by V. Lehtinen and M. Renfors in [Leh09] offers a low-cost arbitrary SRC filter solution based on Lagrange interpolation. The structure modifies what was proposed by S. Tassart and Ph. Depalle in [Tas96][Tas97] to accommodate for a variable fractional delay. The state-of-the-art study developed in chapter 3 of A. Franck thesis [Fra11] confirms that the Newton structure implements Lagrange interpolation using the lowest number of arithmetic operations. However, in [Fra11] this structure was not optimal for software implementation, but this work considers direct hardware implementations, in which case the Newton structure is the implementation of Lagrange interpolation that uses the least amount of hardware resources. To extend the use of the Newton structure, D. Lamb *et al.* developed the transformation matrices that show the relation between the Farrow implementation of the Lagrange interpolation and the Newton interpolation structure [Lam16]. These transformations were applied to B-spline interpolation of order 3 using the Farrow structure (proposed in [Doo99]) to get a modified Newton structure implementation. In this section, the Newton structure is derived first in Section 2.5.1, then the Farrow to Newton transformation is presented in Section 2.5.2. Finally, the practical implementation aspects are discussed in Section 2.5.3.

2.5.1 Newton Structure Derivation

Lagrange interpolation is a widely used interpolation technique for V-FDF implementation due its simplicity and the maximally flat passband of its frequency response. This section develops the definition of this interpolation, and then derives the corresponding Newton structure V-FDF implementation.

Lagrange Interpolation: Lagrange interpolation fits a polynomial $y(t)$ of degree $N-1$ through N points $x[m]$ such that:

$$y(t) = \sum_{m=0}^{N-1} P_m\left(\frac{t}{T_{in}}\right) x[m] \quad / \quad t \in [0, (N-1)T_s] \tag{2.34}$$

Where $P_m(t)$ are the Lagrange polynomials that can be used to find the coefficients of the interpolation filter:

$$P_m(\tau) = \prod_{\substack{k=0 \\ k \neq m}}^{N-1} \frac{\tau - k}{m - k} \quad / \quad \tau = \frac{t}{T_{in}} \tag{2.35}$$

It's clear that the Lagrange interpolation filter is piecewise polynomial by definition, and therefore can be implemented using the Farrow structure. It is always important to correctly center the development of the polynomial pieces in order to benefit from the symmetry of coefficients [Den07]. The filter response was developed in relation to the B-spline function in [Fra09], where the frequency response $H_L^N(f)$ of Lagrange interpolation of order $N-1$ is given with respect to that of B-spline interpolation of order $N-1$ as:

$$H_L^N(f) = G(f)\boldsymbol{B}^N(f) \tag{2.36}$$

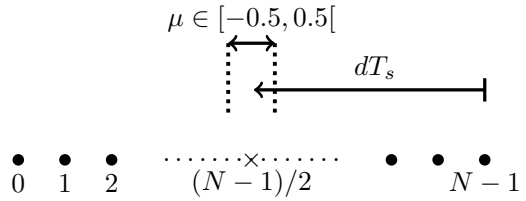


Figure 2.12: The relation between the centered and the causal fractional delays

The function $G(f)$ in this expression has no zeros. This shows that the zeros of the Lagrange interpolation are the same as the ones for the B-spline one, hence sharing the role of image suppression. To better understand the properties of the frequency response of Lagrange interpolation, one should consider the ideal interpolation filter response, where for a fractional delay $d \in \mathbb{R}$, the ideal filter is defined as:

$$h^d(t) = sinc\left(\frac{t}{T_{in}} + d\right) \Rightarrow H^d(f) = e^{j2\pi f d T_{in}} \quad \text{with} \quad d < 0 \tag{2.37}$$

The ideal filter is non realizable, however it can be approximated. One way to approximate the ideal as an FIR filter is through the maximally flat criterion. This criterion states that for an FIR filter of order $N-1$ to be maximally flat at $f = 0$ Hz, the difference between the FIR and the ideal filters should be zero for the frequency response and its derivatives up to degree $N-1$ at $f = 0$ Hz. This results in a linear system of N equation with N unknowns that has a unique solution, which are the coefficients of Lagrange interpolation, as it was proven in [Her92]. This means that the frequency response of Lagrange interpolation focuses on approximating the ideal filter at $f = 0$ Hz the best way possible, by having the maximally flat response possible in the passband.

Newton Structure: The Newton structure is a particular way of implementing the maximally flat FIR filter at $f = 0$ Hz. To have the best linear phase response possible, a centered interpolation range is required, therefore the delay factor $\mu \in [-0.5, 0.5[$ is transformed into the causal version of the delay referenced from the last input as illustrated in Figure 2.12. The causal fractional delay noted d is expressed as:

$$d(i_k) = \mu(i_k) - \frac{N-1}{2} \tag{2.38}$$

In the rest of this chapter we express $d(i_k)$ using the constant notation d to keep a concise presentation. However the value of d is always a function of the index i_k. The Z-domain transfer function of the ideal filter definition in Equation (2.37) can be written as:

$$H^d(z) = z^d \quad \text{with} \quad z = e^{-j2\pi f T_{in}} \quad \text{and} \quad d < 0 \tag{2.39}$$

The problematic point in this expression is that in the Z-domain, the terms z are used to express delays by an integer number of samples. However in this case d is a real value, and therefore evaluating $H^d(z)X(z)$ in this form will result in an ambiguous output expression. This was addressed in [Ram90], where the transfer function $z^{-p/q}$ was approximated as a partial series of terms z^{-i}, with p and q being real numbers, and i being integer. This is done using the generalized binomial theorem, where it is possible to re-write $H^d(z)$ as:

$$\begin{aligned} H^d(z) &= \left[1 + \left(z^{-1} - 1\right)\right]^{-d} \\ &= \sum_{k=0}^{+\infty} \frac{-d(-d-1)\dots(-d-k+1)}{k!} \left(z^{-1} - 1\right)^k \end{aligned} \tag{2.40}$$

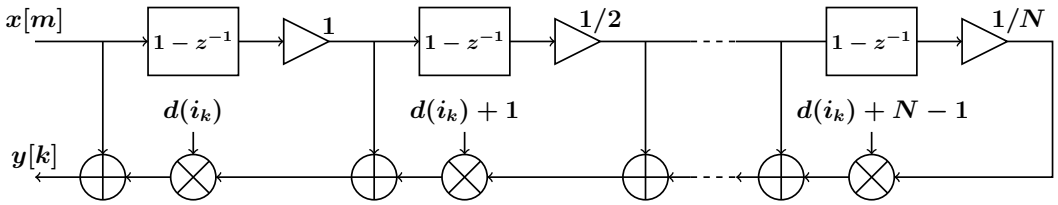


Figure 2.13: The Newton Structure implementing Lagrange interpolation of order N

This series converges for $|z^{-1}-1| < 1$. Truncating this expression to the first N terms is known to result in the partial sum that satisfies the maximally flat criterion presented previously [Tas97][Pei01][Sam04]. And since this criterion has a unique solution, then the partial sum of N terms corresponds to the Lagrange interpolation of order $N-1$ [Her92][Koo96]. Developing the partial sum in term of powers of $(1-z^{-1})$ results in:

$$H_N^d(z) = \sum_{k=0}^{N-1} \frac{d(d+1)\dots(d+k-1)}{k!} \left(1-z^{-1}\right)^k \tag{2.41}$$

This is the expression of the Newton backward difference formula of order $N-1$, that implements the Lagrange interpolation. Developing this expression using Horner's scheme results in the modular Lagrange interpolation filter presented in [Tas96][Tas97]. This filter structure is to be modified as it is shown by [Leh09] in order to support variable fractional delays, resulting in the Newton interpolation structure, presented in Figure 2.13, having a complexity of $O(N)$ compared to that of the Farrow structure implementing Lagrange interpolation with a complexity of $O(N^2)$.

2.5.2 Farrow to Newton Transformation

The matrix representation of the Farrow structure transfer function presented in Equation (2.24) uses a monomial polynomial base vector μ_k^T, and a cumulative delay base vector $Z(z)$. The work developed in [Lam16] shows that it is possible to transform these base vectors to those of the Newton structure, it is then possible to deduce the corresponding Newton structure implementation of any Farrow structure. The Newton structure is based on the Newton backward difference formula, which is based on differentials between samples and their derivatives. In this case, the polynomial evaluation is done with the Newton polynomial base vector d^T expressed as:

$$\underset{1\times M}{\boldsymbol{d}^T} = \left[1, d, d(d+1), \dots, \Pi_{i=0}^{M-2}(d+i)\right] \tag{2.42}$$

With d being the causal fractional delay expressed in Equation (2.38). For the delay vector, the Newton structure uses a differential delay base vector $\bigtriangledown Z(z)$ expressed as:

$$\underset{N\times 1}{\bigtriangledown Z(z)} = \left[1, (1-z^{-1}), \dots, (1-z^{-1})^{N-1}\right] \tag{2.43}$$

Then the matrix representation of the Newton structure transfer function is written as:

$$H_{i_k}(z) = \underset{1\times M}{\boldsymbol{d}^T} \times \underset{M\times N}{Q} \times \underset{N\times 1}{\bigtriangledown Z(z)} \tag{2.44}$$

With the matrix Q containing the coefficients describing the Newton structure implementation. By transforming the base vectors in Equation (2.24) to those of the Newton structure, the Newton structure coefficients matrix Q corresponding to the matrix P of the Farrow

structure can be found as follows:

$$\begin{aligned} H(z,\mu) &= \boldsymbol{\mu}^T P\ Z(z) \\ &= \boldsymbol{\mu}^T \left(T_d^T\ T_d^{-T}\right)\ P\ \left(T_z^{-1}\ T_z\right)\ Z(z) \\ &= (T_d\ \boldsymbol{\mu})^T \left(T_d^{-T}\ P\ T_z^{-1}\right)\ (T_z\ Z(z)) \\ &= \boldsymbol{d}^T\ Q\ \bigtriangledown Z(z) \end{aligned} \tag{2.45}$$

With $T_d^{-T} = \left(T_d^T\right)^{-1}$ and the matrices T_d and T_z are the ones transforming the vectors $\boldsymbol{\mu}$ and $Z(z)$ to $\boldsymbol{d}$ and $\bigtriangledown Z(z)$ respectively. The analytical expressions of these matrices are developed later in Chapter 4. Applying this transformation to a Farrow structure implementation of order 3 results in the matrix Q_{L3}:

$$P_{L3} = \frac{1}{48}\begin{pmatrix} -3 & 17 & 27 & -3 \\ -2 & 54 & -54 & 2 \\ 12 & -12 & -12 & 12 \\ 8 & -24 & 24 & -8 \end{pmatrix} \quad \Rightarrow \quad Q_{L3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/6 \end{pmatrix} \tag{2.46}$$

This is the description of the classical Newton structure of order 3. The work in [Lam16] proposes then to apply this transformation to the B-spline interpolation of order 3, resulting in the matrix Q_{S3}:

$$P_{S3} = \frac{1}{48}\begin{pmatrix} 1 & 23 & 23 & 1 \\ 6 & 30 & -30 & -6 \\ 12 & -12 & -12 & 12 \\ 8 & -24 & 24 & -8 \end{pmatrix} \quad \Rightarrow \quad Q_{S3} = \begin{pmatrix} 1 & 0 & 1/6 & 1/6 \\ 0 & 1 & 0 & 1/6 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/6 \end{pmatrix} \tag{2.47}$$

The resulting modified Newton structure for B-spline interpolation requires only 3 extra additions compared to the Newton structure of order 3 for Lagrange interpolation. This thesis further explores this transformation in Chapter 4 to develop new low-cost arbitrary SRC solutions based on the Newton structure.

2.5.3 Practical Implementation

The Newton structure does not have any different practical implementation aspects than those already presented in this chapter. The same hardware implementation considerations and the same coefficients quantization guidelines presented in Section 2.2.2 apply for the Newton structure. For the quantization of the signal, refer to Chapter 5 for a practical quantization example of the Newton structure of order 5. The quantization of the fractional delay follows the same approach presented for the Farrow structure in Section 2.3.2. The implementation of the SRC controller is also identical to that presented in Section 2.2.2, with the only difference being the added circuitry to calculate the value of the causal fractional delay d. Finally, The hardware implementations on FPGA and ASIC targets are developed in Chapter 6 using different strategies of implementation.

2.6 Comparison and Applications

The different SRC filters presented in this chapter offer a set of SRC solutions that allows the efficient implementation of any SRC application in the DFE. In Section 2.6.1, the different SRC filters are first compared to better understand the characteristics of each filter, highlighting thereby the advantages and inconveniences of each solution. With this understanding, Section 2.6.2 then discusses how to efficiently implement any SRC application by using the appropriate cascade of SRC filters.

Table 2.1: Comparison of SRC FIR Filters of Order N

$R = U/D$	**U-F-D/Polyphase**	**Farrow Structure**	**CIC Filters**	**Newton Structure**
Impulse Response	Symmetrical FIR	Symmetrical Piecewise Polynomial FIR	B-Spline Interpolation	Lagrange Interpolation
Highest Operating Sampling Frequency	$R > 1:\ F_{out}$ $R < 1:\ UF_{in}$	$R > 1:\ F_{out}$ $R < 1:\ F_{in}$	UF_{in}	$R > 1:\ F_{out}$ $R < 1:\ F_{in}$
Order of Multiplications	$R > 1:\ O(UN)$ $R < 1:\ O(DN)$	$O(N^2)$	Zero!	$O(N)$
Order of Additions	$R > 1:\ O(UN)$ $R < 1:\ O(DN)$	$O(N^2)$	$O(N)$	$O(N)$

2.6.1 Characteristics Comparison

Three main characteristics are used to qualify an SRC filter structure: the type of impulse response defining the filtering performance, the highest operating sampling frequency present in the implementation, and the calculation complexity reflected by the number of additions and multiplications required. These characteristics are shown in Table 2.1 for each structure developed in this chapter. Each structure makes compromises between the implementation complexity and the possible filtering performances.

The U-F-D structure can implement any symmetrical FIR response with a linear phase. However for fine-tuned SRC factor R where U and D get very large, the implementation is very inefficient if not impossible. This is because having an operating frequency of UF_{in} may present a calculation complexity beyond what the implementation target can support. The polyphase structure is more efficient than the U-F-D structure in the case of interpolation, since it can be implemented with a maximum sampling frequency of F_{out} by using the structure with the SRC controller as explained in Section 2.2.2. However for fine-tuned SRC factors R, the complexity of the implementation is still high, where a very large number of coefficients should be stored.

The Farrow structure allows building the impulse response through polynomial pieces, resulting in an implementation complexity that is independent of the parameters U and D. This justifies the compatibility of this structure with finely-tuned SRC, where the complexity of the implementation (number of additions and multiplications) does not increase as the parameters U and D take very large values. The compromise to be made in this case is that the impulse response is limited to polynomial based filters. However this still allows building customized filter responses using optimization techniques for filters synthesis presented in Section 2.3.2.

The CIC filters implement the B-spline interpolation only, through a modified U-F-D structure. However no multiplications are required, offering a very low cost implementation solution. The CIC filter is presented as capable of implementing rational factor SRC, however attention should be given if the system can support the CIC maximal sampling frequency of UF_{in}. This is the desirable effect when performing an interpolation by U, however for decimation by a rational factor when $D > U$, the CIC filter may not always be practical. For decimation applications of integer factors when $U = 1$, the maximal sampling frequency is that of the input which does not present a problem. This is why practically the CIC filters are most often used for SRC operations of large SRC factors that are integer.

The newton structure is a more efficient form of the Farrow structure with a complexity of $O(N)$ in place of $O(N^2)$ for Lagrange interpolation. The compromise for this structure is that its implemented impulse response is restricted to Lagrange interpolation. The modified Newton structure developed in [Lam16] and the new structures proposed in Chapter 4 modify the classical Newton structure to implement different impulse responses with the same order of complexity. This structure can be an interesting solution when low complexity is the main

Table 2.2: The most adapted SRC filters for different SRC applications

Coarse SRC	Interpolation: $R_i > 8$ ($U_i \in \mathbb{N}, D_i = 1$) Decimation: $R_i < 1/8$ ($U_i = 1, D_i \in \mathbb{N}$)	CIC Filter
	Interpolation: $1 < R_i < 8$ (R_i rational) Decimation: $1/8 < R_i < 1$ (R_i rational)	U-F-D / Polyphase
	Interpolation: $R_i = 2$ ($U_i = 2, D_i = 1$) Decimation: $R_i = 1/2$ ($U_1 = 1, D_i = 2$)	Half-Band Filter
Fine SRC	High Filtering Performance	Farrow Structure
	Low-Cost Implementation	Newton Structure

design goal, and the filtering performance of the possible impulse responses is adequate for the desired application.

2.6.2 Implementing an SRC Application

As presented in Section 2.1.2, SRC applications are often implemented as a cascade of multiple SRC filters. In this case the SRC factor can be written as:

$$R = R_1 \times R_2 \times \cdots \times R_m = \frac{U_1}{D_1} \times \frac{U_2}{D_2} \times \cdots \times \frac{U_m}{D_m} \quad \text{such that} \quad (U_k, D_k) \in \mathbb{N}^2 \tag{2.48}$$

Each factor R_i is implemented using the appropriate SRC filter structure. Table 2.2 sums-up the SRC types and the most adapted SRC filter corresponding to each SRC application. To partition an SRC operation into a multistage implementation, a simple guide is proposed below for a partition into 3 SRC operations:

1. For SRC application with large changes in the sampling frequency, extract the factor R_1 representing an interpolation or decimation operation by an integer factor larger than eight. The CIC filters then offer a low-cost implementation with good filtering performances. However attention should be given to the passband droop, that may require correction as explained in Section 2.4.2. If the filtering response of the CIC filters is not satisfying for the design criteria, a U-F-D or polyphase implementation allows defining a custom filter response at the cost of increased implementation complexity. If the factor R_1 can be broken into multiple interpolation or decimation factors by 2 or 1/2 respectively, the half-band filters are a particular case that allow improving the filtering performances relatively to the CIC filters while keeping a low implementation complexity.
2. After extracting the factor R_1, identify if there exists a residual coarse rational or integer SRC factor R_2. Usually when a CIC filter is used, this second stage combines further SRC with the CIC filter response compensation. The implementation choice of R_2 largely depends on the system. For example, if it is expected that the SRC factor will be variable, a re-programmable polyphase or U-F-D filter might be needed. Another example, for coarse rational factor SRC (e.g. $R_2 = 7/6$) it is possible to use the Farrow or Newton structures when their filtering responses are acceptable.
3. Finally, the fine adjustment of the output sampling frequency is represented by the factor $R_3 \approx 1$. If the application requires minimal degradation of the signal quality, the Farrow structure is privileged that can achieve an attenuation of 60 dB to 70 dB in the filter response stopband for an order 5 implementation. When less strict filtering performance is required, the Newton structure or one of its modified structures offer a solution with much lower implementation complexity, with a stopband attenuation level between 35 dB and 45 dB for structures of order 5.

2.7 Conclusion

This chapter gives an overview of SRC FIR filtering and proposes a new unified vision of this issue. This presentation started with the definition of SRC and the discussion of important theoretical and practical implementation aspects. Then the four main types of SRC FIR were derived from the common root of linear phase FIR filters. U-F-D and polyphase structure offer a very flexible SRC solution, however with the drawback of a very high implementation complexity. The Farrow structure exploits the definition of polynomial based impulse response to offer a finely-tuned SRC solution with a very low complexity relatively to the previous classical solutions. CIC filters implement the B-spline interpolation impulse response without using any multipliers, offering a very low-cost implementation for coarse SRC. The last solution is the Newton structure, which is a low-cost solution for finely-tuned SRC, however limited to Lagrange interpolation impulse response. The recently proposed transformation that links the Farrow and Newton structures was also presented. Finally, all these solutions were compared and a simplified guide was provided to help in implementing an SRC application efficiently by using the correct filters in an appropriate multi-stage configuration. The next two chapters address in more details the Newton structure and its generalization, in order to design new SRC filter structures of lower complexity and improved performance.

Chapter 3

Theoretical Insights on the Newton Structure

Contents

3.1 Equivalence Between Lagrange and Newton Interpolations . . . 58
3.1.1 Lagrange Interpolation . . . 58
3.1.2 Newton's Forward Difference Formula . . . 59
3.1.3 Newton's Backward Difference Formula . . . 60
3.2 Newton Structure Transfer Function Convergence . . . 61
3.2.1 Fractional Delay Definition . . . 61
3.2.2 The Newton Structure Transfer Function . . . 63
3.2.3 Proof of convergence . . . 65
3.3 Conclusion . . . 68

The classical Newton structure presented in Chapter 2 implements the Newton backward difference formula (NBDF). This chapter identifies two theoretical aspects concerning this structure that deserve further development: the equivalence between the NBDF and Lagrange interpolation, and the Newton structure convergence.

The equivalence between the NBDF and the Lagrange interpolation is well known and accepted in the literature. The proof that is commonly used, and that was presented in Chapter 2, is based on the fact that the partial series in Equation (2.41) satisfies the maximally flat criterion at $f = 0$ Hz [Tas96][Tas97][Her92]. However the direct relation between the NBDF expression and the one for Lagrange interpolation is not explicitly developed in the literature. Section 3.1 develops in details the definition of Lagrange interpolation and the Newton interpolation expressions, and it explicitly shows how one expression can be directly deduced from the other. This relation is useful for the demonstration of the Newton structure transfer function convergence.

An important point overlooked in the literature is the convergence of the Newton structure to the ideal filter as the order of interpolation increases to infinity. This is important to justify that as the implementation order increases, the filtering performance is improved. The convergence was only discussed in [Tas97], and proposed to be true using a conjecture. The delicacy of this convergence lies in the fact that the expression of the transfer function (as shown in Equation (2.41)) is of a causal filter, and the fractional delay d is taken at the center of the interpolation interval. This means that when the order of the filter tends to infinity, the factor d will tend to infinity as well. To circumvent this, the filter response should be centered in time, and as a consequence the fractional delay as well. To our knowledge, there is no rigorous proof that demonstrates that the Newton structure performance improves at its order increases. Section 3.2 develops the proof that the transfer function of the Newton structure

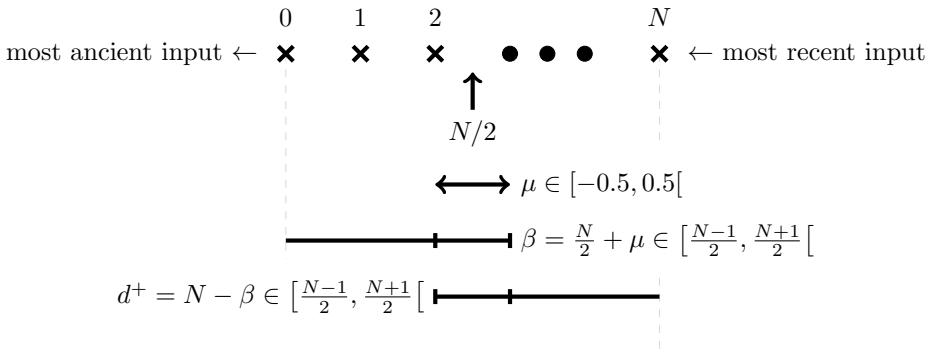


Figure 3.1: Interpolation Indexing

converges to the Shannon ideal interpolator as its order tends to infinity, and therefore its frequency response is improved as the order increases.

3.1 Equivalence Between Lagrange and Newton Interpolations

The equivalence between Lagrange interpolation and the Newton Backward difference formula is well known and widely accepted, and has been proven by Gauss in 1783 [Eul83, p.275-276], and noted in [Gau86][Mei02]. However the direct relation between the polynomials constructed through the different interpolation formulas could not be found in the literature. In this section the interpolation formulas are developed in a way to show the direct relation between them. This development is used in the next section to prove the convergence of the Newton structure to the ideal filter.

3.1.1 Lagrange Interpolation

In this section we're interested in the case of equidistant interpolation samples. To define Lagrange interpolation we use the indexing shown in Figure 3.1. Let $x(t)$ be a signal defined over $t \in [0; N]$. Lagrange interpolation of order N aims to interpolate the value of x at instant $\beta \in [\frac{N-1}{2}; \frac{N+1}{2}]$ using $N+1$ samples $x(k)$ with $k \in \{0, 1, \ldots, N\}$. We can write $x(\beta)$ as:

$$x(\beta) = \sum_{k=0}^{N} L_k(\beta) x(k) \tag{3.1}$$

With $L_k(\beta)$ being the Lagrange polynomials defined as:

$$L_k(\beta) = \prod_{\substack{i=0 \\ i \neq k}}^{N} \frac{\beta - i}{k - i} \tag{3.2}$$

Let us suppose now that we have a signal $x(t)$ defined over $t \in]-\infty; +\infty[$. Lagrange interpolation of order N can be now expressed in two manners:

Referenced by the Most Ancient Input: Referencing the signal to the most ancient input instant $n \in \mathbb{Z}$ gives:

$$x(n + \beta) = \sum_{k=0}^{N} L_k(\beta) x(n + k) \tag{3.3}$$

$L_k(\beta)$ is then developed as:

$$\begin{aligned} L_k(\beta) &= \frac{\beta(\beta-1)\dots(\beta-N)}{(\beta-k)} \times \frac{1}{(k)(k-1)\dots(2)(1)(-1)(-2)\dots(k-N)} \\ &= \frac{\beta!}{(\beta-N-1)!} \times \frac{1}{(\beta-k)} \times \frac{(-1)^{N-k}}{k!(N-k)!} \\ &= (-1)^{N-k}\frac{\beta!}{k!(\beta-k)!} \times \frac{(\beta-k-1)!}{(N-k)!(\beta-N-1)!} \end{aligned} \tag{3.4}$$

From which we can write Equation (3.3) as:

$$x(n+\beta) = \sum_{k=0}^{N}(-1)^{N-k}\binom{\beta}{k}\binom{\beta-k-1}{N-k}x(n+k) \tag{3.5}$$

Referenced by the Most Recent Input : If the interpolation is referenced by $n \in \mathbb{Z}$ corresponding to the most recent input of the signal, Equation (3.1) would be written as:

$$\begin{aligned} x(n-N+\beta) &= \sum_{k=0}^{N} L_k(\beta)x(n-N+k) \\ &= \sum_{k=0}^{N} L_{N-k}(\beta)x(n-k) \end{aligned} \tag{3.6}$$

Replacing β by $N-d^+$ gives:

$$x(n-d^+) = \sum_{k=0}^{N} L_{N-k}(N-d^+)x(n-k) \tag{3.7}$$

Then by developing L_{N-k} we get:

$$\begin{aligned} L_{N-K}(N-d^+) &= \prod_{\substack{i=0 \\ i\neq N-K}}^{N} \frac{N-d^+-i}{N-K-i} \\ &= \prod_{\substack{i=0 \\ i\neq K}}^{N} \frac{-d^++i}{-K+i} = L_k(d^+) \end{aligned} \tag{3.8}$$

It is possible now to write Equation (3.7) as:

$$x(n-d^+) = \sum_{k=0}^{N}(-1)^{N-k}\binom{d^+}{k}\binom{d^+-k-1}{N-k}x(n-k) \tag{3.9}$$

3.1.2 Newton's Forward Difference Formula

Newton's forward difference formula is defined as:

$$x(n+\beta) = \sum_{j=0}^{N} \frac{\beta(\beta+1)(\beta+2)\dots(\beta+j-1)}{j!} \triangle^j [x(n)] \tag{3.10}$$

With $\triangle$ being the forward difference operator, defined as:

$$\triangle^j [x(n)] = \sum_{k=0}^{j}(-1)^k \binom{j}{k} x(n+j-k) \tag{3.11}$$

Equation (3.10) can now be written as:

$$x(n+\beta) = \sum_{j=0}^{N} \binom{\beta}{j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} x(n+j-k) \tag{3.12}$$

A needed property to prove the equivalence with Lagrange interpolation, which we call *Property* (α), is the following:

$$\sum_{k=0}^{N} \sum_{j=0}^{N-k} A(N-j,k)x(n-k) = \sum_{k=0}^{N} \sum_{j=k}^{N} A(j,k)x(n-k) = \sum_{j=0}^{N} \sum_{k=0}^{j} A(j,k)x(n-k) \tag{3.13}$$

This property can be verified by developing the summation in a matrix form, and it can be seen that for an index j, only the terms of $x(n-k)$ with k up to j are evaluated by $A(j,k)$, which translates that for an index k only terms $A(j,k)$ with $j > k$ exist:

$$\begin{pmatrix} A(0,0)x(n) & 0 & 0 & \dots & 0 \\ A(1,0)x(n) & A(1,1)x(n-1) & 0 & \dots & 0 \\ A(2,0)x(n) & A(2,1)x(n-1) & A(2,2)x(n-2) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A(N,0)x(n) & A(N,1)x(n-1) & A(N,2)x(n-2) & \dots & A(N,N)x(n-N) \end{pmatrix} \tag{3.14}$$

From Equation (3.5), we can now develop the following:

$$\begin{aligned}
& & x(n+\beta) &= \sum_{k=0}^{N} (-1)^{N-k} \binom{\beta}{k} \binom{\beta-k-1}{N-k} x(n+k) \\
& \left[\sum_{m=0}^{n} (-1)^{n-m} \binom{r}{m} = \binom{r-1}{n}\right] & &= \sum_{k=0}^{N} \binom{\beta}{k} (-1)^{N-k} \sum_{j=0}^{N-k} (-1)^{N-k-j} \binom{\beta-k}{j} x(n+k) \\
& j \to j-k & &= \sum_{k=0}^{N} \sum_{j=k}^{N} (-1)^{j-k} \left[\binom{\beta-k}{j-k} \binom{\beta}{k}\right] x(n+k) \\
& \left[\binom{a}{b}\binom{b}{c} = \binom{a-c}{b-c}\binom{a}{c}\right] & &= \sum_{k=0}^{N} \sum_{j=k}^{N} (-1)^{j-k} \binom{\beta}{j} \binom{j}{k} x(n+k) \\
& \text{Property } (\alpha) & &= \sum_{j=0}^{N} \binom{\beta}{j} \sum_{k=0}^{j} (-1)^{j-k} \binom{j}{k} x(n+k) \\
& k \to j-k & &= \sum_{j=0}^{N} \binom{\beta}{j} \sum_{k=0}^{j} (-1)^{k} \binom{j}{j-k} x(n+j-k) \\
& \left[\binom{a}{b} = \binom{a}{a-b}\right] & &= \sum_{j=0}^{N} \binom{\beta}{j} \sum_{k=0}^{j} (-1)^{k} \binom{j}{k} x(n+j-k)
\end{aligned} \tag{3.15}$$

Then Equation (3.12) can be found from Equation (3.5) as shown in Equation (3.15). This proves the equivalence of Lagrange interpolation and the Newton's forward difference formula.

3.1.3 Newton's Backward Difference Formula

Newton's backward difference formula is defined as:

$$x(n+d) = \sum_{j=0}^{N} \frac{d(d+1)(d+2)\dots(d+j-1)}{j!} \bigtriangledown^j [x(n)] \tag{3.16}$$

Where $\bigtriangledown$ is the backward difference, defined as:

$$\bigtriangledown^j [x(n)] = \sum_{k=0}^{j} (-1)^k \binom{j}{k} x(n-k) \tag{3.17}$$

By taking $d = -d^+$, Equation (3.16) can be written as:

$$x(n - d^+) = \sum_{j=0}^{N} (-1)^j \binom{d^+}{j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} x(n-k) \tag{3.18}$$

From Equation (3.9), we can develop the following:

$$\begin{aligned}
x(n-d^+) &= \sum_{k=0}^{N} \binom{d^+}{k} (-1)^{N-k} \left[\binom{d^+ - k - 1}{N-k} \right] x(n-k) \\
\left[\sum_{m=0}^{n} (-1)^{n-m} \binom{r}{m} = \binom{r-1}{n} \right] \quad &= \sum_{k=0}^{N} \binom{d^+}{k} (-1)^{N-k} \sum_{j=0}^{N-k} (-1)^{N-k-j} \binom{d^+ - k}{j} x(n-k) \\
j \to N-k-j \quad &= \sum_{k=0}^{N} (-1)^{N-k} \sum_{j=0}^{N-k} (-1)^j \left[\binom{d^+ - k}{N-j-k} \binom{d^+}{k} \right] x(n-k) \\
\left[\binom{a}{b}\binom{b}{c} = \binom{a-c}{b-c}\binom{a}{c} \right] \quad &= \sum_{k=0}^{N} \sum_{j=0}^{N-k} (-1)^{N-j+k} \binom{d^+}{N-j} \binom{N-j}{k} x(n-k) \\
\text{Property } (\alpha) \quad &= \sum_{j=0}^{N} (-1)^j \binom{d^+}{j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} x(n-k)
\end{aligned} \tag{3.19}$$

Then Equation (3.18) can be found from Equation (3.9) as shown in Equation (3.19). This proves the equivalence of Lagrange interpolation and the Newton's backward difference formula.

3.2 Newton Structure Transfer Function Convergence

This section develops a rigorous demonstration of the uniform convergence of Lagrange interpolation transfer function to the ideal fractional delay, and as a consequence the convergence of the Newton structure as well. This demonstration also provides an alternate approach to define the ideal fractional delay as the digital filter implementing Lagrange interpolation of an infinite order. This result is primordial for any analytical development that aims to qualify the fractional delay error of the approximation implemented by the digital filter. This development starts by developing the fractional delay definition in Section 3.2.1. In Section 3.2.2, the problematic of finding the limit of Lagrange interpolation is introduced. The solution is then developed in Section 3.2.3.

3.2.1 Fractional Delay Definition

3.2.1.1 Ideal Fractional Delay

Let $x[n] = x(nT_s)$ with $n \in \mathbb{Z}$ be uniformly spaced samples taken with a sampling frequency $F_s = 1/T_s$ from a continuous-time signal $x(t)$ defined over $t \in \mathbb{R}$, and sampled at instants nT_s. In order to reconstruct the signal $x(t)$ from its samples, the signal must be band-limited, having a bandwidth B in the frequency domain, and the sampling frequency must be equal to at least twice the signal's bandwidth $F_s \geq 2B$. This is commonly known in the literature as the Nyquist-Shannon sampling theorem [Jer77; Sha49]. The ideal signal reconstruction is then performed through the Whittaker-Shannon interpolation formula or sinc interpolation given as:

$$x(t) = \sum_{m \in \mathbb{Z}} sinc\left(\frac{t}{T_s} - m \right) x[m] \tag{3.20}$$

where $sinc(x) = sin(\pi x)/\pi x$. This series is known to uniformly converge for $t \in \mathbb{R}$ [Whi35]. In this chapter, we are interested in finding $y[n]$ defined as:

$$\begin{aligned} y[n] = y(nT_s) &\triangleq x(nT_s - \mu T_s) \\ &= \sum_{m\in\mathbb{Z}} sinc(n-\mu-m)x[m] \end{aligned} \tag{3.21}$$

with $0 \leq \mu < 1$. The expression in Equation (3.21) is the discrete convolution $(h_{id}^{\mu} * x)[n]$, which is equivalent to a filtering operation. The filter impulse response corresponds to the time shifted cardinal sine function $h_{id}^{\mu}[n] = sinc(n-\mu)$. The discrete-time Fourier transform (DTFT) of $h_{id}^{\mu}[n]$ can be found by applying Equation (3.20) to the signal $x(t) = e^{2j\pi ft}$ with $t = (n-\mu)T_s$:

$$e^{2j\pi(n-\mu)fT_s} = \sum_{m\in\mathbb{Z}} sinc(n-\mu-m)e^{2j\pi mfT_s} \tag{3.22}$$

then

$$\begin{aligned} e^{-2j\pi\mu fT_s} &= \sum_{m\in\mathbb{Z}} sinc(n-\mu-m)e^{-2j\pi(n-m)fT_s} \\ &= \sum_{m\in\mathbb{Z}} sinc(m-\mu)e^{-2j\pi mfT_s} \end{aligned} \tag{3.23}$$

The series in Equation (3.23) converges uniformly for $\mu \in \mathbb{R}$, and in particular for $0 \leq \mu < 1$. By fixing μ and taking f as the variable, we recognize two Fourier transforms. On the left, the DTFT of $\delta(. - \mu T_s)$, and on the right, the DTFT of $h_{id}^{\mu}[n]$. Therefore the fractional delay $\delta(. - \mu T_s)$ is defined as the all-pass digital filter with a constant group delay of μT_s.

3.2.1.2 Z-transform of the Fractional Delay

In this section, we introduce the Z-transform used to express a series in a way that each element n is indexed by z^{-n} with $z \in \mathbb{C}$ and $n \in \mathbb{Z}$:

$$X(z) = \mathcal{Z}\{x[n]\} \triangleq \sum_{n\in\mathbb{Z}} x[n]z^{-n} \tag{3.24}$$

The formal relation between $y[n]$ and $x[n]$ can be found through the Z-transform of $y[n]$ as:

$$Y(z) = \sum_{n\in\mathbb{Z}} y[n]z^{-n} \triangleq \sum_{n\in\mathbb{Z}} x(nT_s - \mu T_s)z^{-n} \tag{3.25}$$

the ideal filter $H_{id}^{\mu}(z)$ finds the signal $Y(z)$ by filtering the input $X(z)$, which can be expressed as:

$$Y(z) = H_{id}^{\mu}(z)X(z) \tag{3.26}$$

Equation (3.26) is a formal expression since the term $z^{-\mu}$ is not well defined for $z \in \mathbb{C}$. First, the Z-transform is defined for integer delay indexes $n \in \mathbb{Z}$ for z^{-n}, while the delay index is real with $\mu \in [0;1[$ for $z^{-\mu}$. Therefore direct evaluation of $z^{-\mu}X(z)$ in Equation (3.26) results in an ambiguous expression. Second, $z^{-\mu}$ with $\mu \in \mathbb{R}$ is not a well-defined complex function. Taking $z^{-\mu}$ as the exponential of its own complex logarithm results in:

$$z^{-\mu} = e^{-\mu \times log_e(z)} = e^{-\mu[ln|z| + j[\mathrm{Arg}(z) + 2k\pi]]} \tag{3.27}$$

where log_e and ln are the complex and real natural logarithms respectively, $\mathrm{Arg}(z)$ is the principal argument value of z, and $k \in \mathbb{Z}$. It is clear that when μ is non-integer, the term $e^{-j2k\pi\mu}$ will not vanish, resulting in multiple output values for the same term $z^{-\mu}$. Nonetheless, we find this term used sometimes in the literature without any definition to perform formal calculation of transfer functions.

To give a rigorous definition for $z^{-\mu}$, we consider Equation (3.26) representing a filtering operation using a transfer function equal to $z^{-\mu}$. Using the filter coefficients $h_{id}^{\mu}[n]$, the discrete convolution filtering operation is written as following:

$$y[n] = \sum_{m\in\mathbb{Z}} h_{id}^{\mu}[n-m]x[m] \tag{3.28}$$

by identification between Equation (3.21) and Equation (3.28), and using the identity in Equation (3.23), the logical definition of $H_{id}^{\mu}(z)$ would be:

$$H_{id}^{\mu}(z) = \sum_{n\in\mathbb{Z}} sinc(n-\mu)z^{-n} = z^{-\mu} \quad , \quad \text{for} \quad z \in \mathcal{U} \tag{3.29}$$

the infinite sum in Equation (3.29) only converges on subset $\mathcal{U} \subset \mathbb{C}$, defined by $\mathcal{U} = \{z \in \mathbb{C}, |z| = 1, -\pi < arg(z) < \pi\}$. The proof is developed later in Section 3.2.3.1. Hence this infinite sum uniformly converges to $z^{-\mu}$ for all $0 \leq \mu < 1$ and $z \in \mathcal{U}$. This provides a rigorous definition that extends the z-transform to fractional delay indexes. First, this series converges on the unit circle, meaning the frequency response is well defined. Second, it is clear that when μ takes integer values, both sides of Equation (3.29) are equal to $z^{-\mu}$. And finally, an important property to verify is the accumulation of delay. Mathematically, the accumulation can be expressed as:

$$z^{-\mu_1} z^{-\mu_2} = z^{-(\mu_1+\mu_2)} \tag{3.30}$$

the above property can be verified by proving that:

$$\sum_{m\in\mathbb{Z}} sinc(m-\mu_1)sinc(n-m-\mu_2) = sinc(n-\mu_1-\mu_2) \tag{3.31}$$

replacing $sinc(x)$ by its Taylor series in Equation (3.31) proves the property shown in Equation (3.30). Hence, the definition of $z^{-\mu}$ in Equation (3.29) is rigorous with no ambiguities on the unit circle, which is the domain of interest for filter response analysis.

3.2.2 The Newton Structure Transfer Function

3.2.2.1 Demonstration Context

An ideal filter is by definition not realizable since it is not causal. Fractional delay filter implementations can only approximate the ideal filter. Many different approximations are proposed in the literature [Laa96], however, we focus in this chapter on the approximation proposed in [Tas96][Tas97]. It is important to note that in these two references it is not the ideal filter response that is approximated, but it is rather the response of a causal fractional delay filter of infinite order. This is because the Newton's generalized binomial theorem is used to develop the expression of $z^{-\delta}$, where δ in this case is the fractional delay at the center of the interpolation interval (*i.e.* $\delta \in \left[\frac{N}{2} - \frac{1}{2}; \frac{N}{2} + \frac{1}{2}\right[$). In [Tas97], the term $z^{-\delta}$ is expressed as:

$$\begin{aligned} H^{\delta}(z) &= \left[1 + \left(z^{-1} - 1\right)\right]^{\delta} \\ &= \sum_{k=0}^{+\infty} \binom{\delta}{k} \left(z^{-1} - 1\right)^{k} \end{aligned} \tag{3.32}$$

where $\binom{\delta}{k}$ is the generalized binomial coefficient. This series converges for $|z^{-1} - 1| < 1$, and therefore its frequency response is only defined for frequencies $|f| < F_s/3$. The approximated digital filter transfer function is then taken as the partial series of order N of $H^{\delta}(z)$:

$$H_N^{\delta}(z) = \sum_{k=0}^{N} \binom{\delta}{k} \left(z^{-1} - 1\right)^{k} \tag{3.33}$$

Since the transfer function $H_N^{\delta}(z)$ is causal, the delay δ taken at the center of the interpolation interval will tends to infinity when the order N tends to infinity. However it is always possible to break the causal fractional delay into an integer delay and a fractional delay $0 \leq \mu < 1$. In the case of a filter with an odd order $2N+1$, we have $\delta = N + \mu$, and we can write:

$$z^{-\delta} = z^{-N} z^{-\mu} \tag{3.34}$$

The same development can be done for an even filter order, however with a fractional delay shift by 1/2. Equation (3.34) shows that the relation between the expression of the causal fractional delay $z^{-\delta}$ developed in [Tas96; Tas97], and the ideal fractional delay $z^{-\mu}$, consists of a delay shift by z^{-N} having the role of centering the causal response with respect to the time reference. This delay shift is infinite for the ideal case, since by definition an ideal filter cannot be made causal. Hence the only approach to build a convergence proof consists of rewriting Equation (3.33) in the form of Equation (3.34), and then studying the convergence of the approximation of $z^{-\mu}$. In the case this is true, then a mathematical proof is provided to the statement done by [Tas97], that the Newton structure frequency response improves as the order of interpolation increases.

3.2.2.2 Useful Notations

By rewriting Equation (3.33) in the form of Equation (3.34), we can identify the corresponding approximation of $z^{-\mu}$. To do so, we start by developing $(z^{-1}-1)^k$ using the binomial theorem and then flipping the order of summation. We can then write $H_N^{\delta}(z)$ in Equation (3.33) as:

$$H_N^{\delta}(z) = \sum_{n=0}^{N} \sum_{k=n}^{N} \binom{\delta}{k} \binom{k}{n} (-1)^{k-n} z^{-n} \tag{3.35}$$

then using some combinatorial properties, we get:

$$H_N^{\delta}(z) = \sum_{n=0}^{N} \left[(-1)^{N-n} \binom{\delta}{n} \binom{\delta - n - 1}{N - n} \right] z^{-n} \tag{3.36}$$

taking a filter order of $2N$ and making two consecutive change of variables: $m = n - N$ and $i' = N - i$, the digital filter transfer function can be expressed as:

$$H_{2N}^{N+\mu}(z) = z^{-N} f_N(z) \tag{3.37}$$

where $f_N(z)$ is given as

$$f_N(z) = \sum_{m=-N}^{N} \left[\prod_{\substack{i=-N \\ i \neq -m}}^{N} \frac{i+\mu}{i+m} \right] z^{-m} \tag{3.38}$$

Using Equation (3.37) and Equation (3.34), we identify that the corresponding finite order approximation of $z^{-\mu}$ is exactly the Lagrange interpolation defined by $f_N(z)$.

Since the $z^{-\mu}$ definition proposed earlier converges for $z \in \mathcal{U}$ only, we let $z = e^{j\theta}$ such that $\theta = 2\pi f T_s$ and $\theta \in I = [a, b] \subset]-\pi, \pi[$. Then the approximation of $z^{-\mu}$ with a filter of order $2N$ is written as:

$$f_N(\theta) = \sum_{n=-N}^{N} L_{-n}^{N}(\mu) e^{-nj\theta} \quad , \quad \theta \in I \tag{3.39}$$

where each $L_{-n}^{N}(\mu)$ is defined as:

$$L_{-n}^{N}(\mu) = \prod_{\substack{i=-N \\ i \neq -n}}^{N} \frac{i+\mu}{i+n} \tag{3.40}$$

3.2.2.3 Demonstration Objective

The objective is to demonstrate that $f_N(\theta)$ converges to $f(\theta) = e^{-j\mu\theta} = z^{-\mu}$, showing that Lagrange interpolation converges to the ideal fractional delay as the order tends to infinity. This is done by demonstrating first that $\lim_{N\to+\infty} f_N$ exists in the sense of $||.||$, where $||f||$ is the uniform norm of f on any interval I, i.e. $||f|| = \sup_{\theta\in I} |f(\theta)|$. Then the convergence of Lagrange interpolation to $z^{-\mu}$ is found as:

$$\lim_{N\to+\infty} f_N(\theta) = f(\theta) = \sum_{n=-\infty}^{+\infty} sinc(n-\mu)e^{-nj\theta} \tag{3.41}$$

3.2.3 Proof of convergence

In this section, we divide the proof into three steps. Firstly, we demonstrate that the function $f(\theta)$ is well defined for $-\pi < \theta < \pi$. Secondly, some needed properties concerning $f_N(\theta)$ are presented. Finally, we prove the convergence of $f_N(\theta)$ to $f(\theta)$.

3.2.3.1 Existence of $f(\theta)$

Let $S_{N^-}^{N^+}$ be the partial sums of $f(\theta)$:

$$S_{N^-}^{N^+}(\theta) = \sum_{n=N^-}^{N^+} sinc(n-\mu)e^{-nj\theta} \tag{3.42}$$

Proving the convergence of $f(\theta)$ consists of demonstrating that the partial sums converge following $||.||$. We consider at first the convergence of $S_0^N(\theta)$ for $N \to +\infty$. Then the convergence of S_N^{-1} as $N \to -\infty$ is directly concluded, and finally the convergence of $S_{N^-}^{N^+}(\theta) = S_0^{N^+}(\theta) + S_{N^-}^{-1}(\theta)$.

By taking $sinc(x) = sin(\pi x)/\pi x$ and expressing the sinus using Euler's formula, $S_0^N(\theta)$ is expressed as:

$$\begin{aligned} S_0^N(\theta) = \frac{1}{2\pi j}\Bigg(&\sum_{n=0}^{N} \frac{e^{j\pi(n-\mu)}}{n-\mu} e^{-nj\theta} \\ &- \sum_{n=0}^{N} \frac{e^{-j\pi(n-\mu)}}{n-\mu} e^{-nj\theta}\Bigg) \end{aligned} \tag{3.43}$$

we start by considering the first summation of $S_0^N(\theta)$:

$$S_N(\omega) = C\sum_{n=0}^{N} \frac{e^{nj\omega}}{n-\mu} \tag{3.44}$$

Where $C = e^{-j\pi\mu}/2\pi j$, and $\omega = \pi - \theta$. We note that the variable C is not dependent on n. To apply Abel convergence criterion, we express $S_N(\omega)$ as:

$$S_N(\omega) = C\sum_{n=0}^{+\infty} a_n b_n \quad \text{, with} \quad a_n = \frac{1}{n-\mu} \ , \ b_n = e^{nj\omega} \tag{3.45}$$

by expressing the series of the geometric sequence b_n as:

$$B_N = \sum_{n=0}^{N} b_n = \frac{e^{(N+1)j\omega} - 1}{e^{j\omega} - 1} \leq \frac{1}{|sin(\frac{\omega}{2})|} \tag{3.46}$$

a bound for B_N is found only if:

$$\omega \neq 0 + 2k\pi \Rightarrow \theta \neq (2k+1)\pi \ , \quad \text{with } k \in \mathbb{Z} \tag{3.47}$$

This condition is always respected since we supposed that $\theta \in I$ in Equation (3.39). On the other side, the sequence a_n converges to zero as $n \to +\infty$, and $\sum (a_{n+1} - a_n)$ is absolutely convergent. Therefore, using the Abel criterion, $S_N(\omega)$ converges following $||.||$ for all intervals I. Applying the same reasoning for the series with the conjugate general term, the series $S_0^N(\theta)$ is shown to converge following $||.||$.

Hence the function $f(\theta)$ is defined as the uniform limit of the partial series $S_{N^-}^{N^+}(\theta)$ on any compact interval $I \subset]-\pi, \pi[$.

3.2.3.2 Useful Properties

Property 1 Using the following Gamma function identities:

- Reflection: $\Gamma(1-z)\Gamma(1+z) = [sinc(z)]^{-1}$, where $z \notin \mathbb{Z}$.
- Addition: $\Gamma(n+z) = \prod_{k=1}^{n}(z+k-1)\Gamma(z)$, where $z \neq 0, -1, -2, ..$ and $n \in \mathbb{N}$.

then $L_n^N(\mu)$ in Equation (3.40) is rewritten as:

$$L_n^N(\mu) = c_n^N(\mu) sinc(n+\mu) \ , \quad \text{for } |n| \leq N \tag{3.48}$$

where:

$$c_n^N(\mu) = c_{-n}^N(\mu) = \frac{\Gamma(N+1+\mu)\Gamma(N+1-\mu)}{\Gamma(N+1+n)\Gamma(N+1-n)} \tag{3.49}$$

Property 2 For all $n \in N$, the limit of $c_n^N(\mu)$ when N tends to infinity is equal to one.

Property 3 For $N \geq 0$, $c_n^N(\mu)$ is a sequence of positive decreasing numbers, and $\lim_{|n| \to +\infty} c_n^N(\mu) = 0$.

3.2.3.3 Convergence of $f_N(\theta)$ to $f(\theta)$

The last proof comes back to proving:

$$\lim_{N \to +\infty} ||f_N - f|| = 0 \tag{3.50}$$

as it was proven above, the symmetric partial sums $S_{N^-}^{N^+}(\theta)$ converge. We consider the symmetric remainder $R_N(\theta)$ expressed as:

$$R_N(\theta) = \sum_{|n|>N} sinc(n-\mu) e^{-nj\theta} \tag{3.51}$$

the remainder $R_N(\theta) = f(\theta) - f_N(\theta)$ is well defined, and $\lim_{N \to +\infty} \sup_{n \geq N} ||R_N|| = 0$. In the following, let M denotes an integer such that $0 \leq M \leq N$, and we define:

$$\begin{aligned} \Delta_M^N(\theta) &= \sum_{|n| \leq M} (c_n^N(\mu) - 1) sinc(n-\mu) e^{-nj\theta} \\ A_M^N(\theta) &= \sum_{M<|n| \leq N} c_n^N(\mu) sinc(n-\mu) e^{-nj\theta} \\ V_n &= \sum_{|k| \leq n} sinc(k-\mu) e^{-kj\theta} \end{aligned} \tag{3.52}$$

For $0 \leq M \leq N$, we want to verify that:

$$||f_N - f|| \leq ||\Delta_M^N|| + ||A_M^N|| + ||R_M|| \tag{3.53}$$

Using the Abel's Lemma [Nic17], and the fact that $c_n^N = c_{-n}^N$, the partial sum A_0^N can be rewritten as:

$$A_0^N = c_N^N V_N - \sum_{0 \leq n < N} (c_{n+1}^N - c_n^N) V_n \tag{3.54}$$

then, we subtract all terms from $-M$ to M to get:

$$A_M^N = A_0^N - A_0^M = c_N^N V_N - c_M^N V_M - \sum_{M \leq n < N} (c_{n+1}^N - c_n^N) V_n \tag{3.55}$$

then writing Equation (3.55) in terms of $R_n = f - V_n$ results in:

$$A_M^N = c_M^N R_M - c_N^N R_N + \sum_{M \leq n < N} (c_{n+1}^N - c_n^N) R_n \tag{3.56}$$

this allows bounding $||A_M^N||$ for $0 \leq M \leq N$ as:

$$\begin{aligned} ||A_M^N|| \leq \big(& c_N^N + c_M^N \\ & + \sum_{M \leq |n| < N} |c_{n+1}^N - c_n^N| \Bigg) \sup_{M \leq n \leq N} ||R_n|| \end{aligned} \tag{3.57}$$

As stated in the property 3, the terms c_n^N are strictly decreasing for $M \leq n \leq N$, then:

$$||A_M^N|| \leq 2 c_M^N \sup_{n \geq M} ||R_n|| \tag{3.58}$$

hence, by replacing Equation (3.58) in Equation (3.53), we find:

$$||f_N - f|| \leq ||\Delta_M^N|| + 2 c_M^N \sup_{n \geq M} ||R_n|| + ||R_M|| \tag{3.59}$$

which can be simplified for $0 \leq M \leq N$ to:

$$||f_N - f|| \leq ||\Delta_M^N|| + \big(2 c_M^N + 1\big) \sup_{n \geq M} ||R_n|| \tag{3.60}$$

Let's now define $\varepsilon > 0$:

1. $\exists\ N_s \in \mathbb{N}$, such that, $M \geq N_s \Rightarrow \sup_{n \geq M} ||R_n|| < \varepsilon/8$ since we proved that $f(\theta)$ converges uniformly. Let us fix $M = N_s$.

2. $\exists\ N_s' \in \mathbb{N}$, such that, $N \geq N_s' \Rightarrow ||\Delta_M^N|| < \varepsilon/2$, since $||\Delta_M^N||$ tends to zero by applying the property 2 on Equation (3.52) with a fixed M.

3. $\exists\ N_s'' \in \mathbb{N}$, such that, $N \geq N_s'' \Rightarrow 0 < c_M^N < 3/2$ using the property 2. Then $0 < 2c_M^N + 1 < 4$.

Hence for $N \geq \max(N_s', N_s'') \Rightarrow ||f_N - f|| < \varepsilon$. Therefore, Equation (3.50) is proven, and the convergence is demonstrated.

3.3 Conclusion

Two theoretical aspects of the Newton structure were addressed in this chapter. The first part provided the direct relation between the transfer function expressions of Lagrange interpolation and the Newton difference formulas. This relation completes the proof of equivalence between the two interpolation found in the literature, by showing explicitly how both interpolations are linked. The second part provided a rigorous proof for the convergence of the Newton structure transfer function using the relation developed in the first part. This convergence proves that the filter response improves and approaches the ideal filter as the order of interpolation is increased. This was necessary to rigorously define the Newton structure for any order, before tackling the generalization of the Newton structure to all polynomial interpolations of any order in the next Chapter 4.

Chapter 4

Generalized and Reconfigurable Newton Structures

Contents

4.1 Generalized Newton Structure 70
4.1.1 Generalization of the Newton Structure 70
4.1.2 Farrow to Newton Transformation 71
4.1.3 Vector Form Transformation 72
4.2 Modified Newton Structure - Hermite Interpolation 72
4.2.1 V-FDF Based on Hermite Interpolation 73
4.2.2 Proposed Low Complexity Hermite Interpolation Based V-FDF . . . 74
4.2.3 Performance and Complexity 75
4.3 Re-configurable Newton Structure 78
4.3.1 Derivation of the Transfer Function 78
4.3.2 Complexity Comparison 81
4.3.3 Application Examples 81
4.4 Optimized Newton Structure 82
4.4.1 Generalized Newton Structure Frequency Response 82
4.4.2 Filter Optimization Application 83
4.5 Conclusion 86

The proposition to use the Newton structure for sample rate conversion in [Leh09] offered a low-cost solution that implements the Lagrange interpolation filtering response. The transformation linking the Farrow and Newton structure proposed by [Lam16] made possible the implementation of any polynomial based filter response using a modified Newton structure. The work proposing this transformation only developed one example implementation using B-spline interpolation of order 3. This chapter further investigates the possibilities of this transformation to build more efficient SRC filters based on the Newton structure. To do so, the transformation expressions for arbitrary polynomial based filter parameters are first developed in Section 4.1. These expressions then provide a definition for the generalized Newton structure. Section 4.2 then presents Hermite interpolation, and applies the transformation to obtain modified Newton structures implementing this type of interpolation. Two examples of structures are developed, and their complexity and performance are compared with the solutions based on Lagrange and B-spline interpolation. With the different available filters based on the Newton structure, Section 4.3 explores the possibility to linearly combine two interpolation methods in a single newton structure. This combination has the objective of building a structure with a reconfigurable filter response that is controlled with one variable parameter.

Application examples of this reconfigurability are also discussed in this section. Finally, using the closed form expression of the Farrow to Newton transformation, Section 4.4 builds the generalized Newton structure frequency response expression. This expression allows the application of filter optimization methods, which are then used to develop an example optimized SRC filter based on the Newton structure. Section 4.5 concludes the results developed in this chapter.

4.1 Generalized Newton Structure

In this section, the generalization of the Newton structure is elaborated based on the work done in [Lam16], where it was shown that there exists a transformation between the Farrow and the Newton structure. The closed form expressions of this transformation are developed, alongside the definition of the generalized Newton structure.

4.1.1 Generalization of the Newton Structure

The Farrow to Newton transformation is made possible by expressing the impulse responses of the two structures in matrix form. The Farrow structure is the efficient implementation of the polynomial based filter developed in [Far88], that consists of M FIR filters in parallel of N taps each, with the outputs of these filters evaluated with the fractional delay μ following Horner's scheme. Its transfer function can be written in matrix form as presented in eq:DynPBFdef in Chapter 2:

$$H^{\mu}(z) = \underset{1\times M}{\boldsymbol{\mu}^T} \times \underset{M\times N}{P} \times \underset{N\times 1}{Z(z)} \tag{4.1}$$

Where $\boldsymbol{\mu}^T = [1, \mu, \ldots, \mu^{M-1}]$ is the monomial based polynomial evaluation vector with $\mu \in [-0.5; 0.5[$, and the vector $Z(z)^T = [1, z^{-1}, \ldots, z^{-N+1}]$ is the z^{-1} based delay vector. The matrix P is what defines the Farrow implementation, where the element p_{ij} of row i and column j contains the j^{th} coefficients of the i^{th} FIR filter.

On the other hand, the Newton structure also implements a polynomial based filter, however using different evaluation bases:

$$H^{d}(z) = \underset{1\times M}{\boldsymbol{d}^T} \times \underset{M\times N}{Q} \times \underset{N\times 1}{\bigtriangledown Z(z)} \tag{4.2}$$

Where $\boldsymbol{d}^T = [1, d, d(d+1), \ldots, \prod_{i=0}^{M-2}(d+i)]$ with $d = \mu - \frac{M-1}{2}$ is the Newton based polynomial evaluation vector, and $\bigtriangledown Z(z)^T = [1, (1-z^{-1}), \ldots, (1-z^{-1})^{N-1}]$ is the $(1-z^{-1})$ based delay vector. The matrix Q describes the Newton structure, where the element q_{ij} is the coefficient that will scale the output of the block $(1-z^{-1})^j$, before multiplying the result by $(1)(d)(d+1)\ldots(d+i-1)$. This can be implemented in different forms, however the classical form of the Newton structure stays the most efficient, where the reuse of calculated values is maximized.

The generalized Newton structure is defined by Equation (4.2), where the matrix Q has $M \times N$ coefficients to be defined. Both the Farrow structure and the generalized Newton structure have a calculation complexity of the same order $O(N \times M)$. However a great advantage of the Newton structure in terms of complexity is visible when implementing filter responses having zeros on multiples of the input sampling frequency F_{in}, such as the case for Lagrange [Leh09], Spline [Lam16] or Hermite (c.f. Section 4.2) interpolations. Where in these cases, most of the coefficients describing the Newton structure are zeros, thereby making the implementation efficient with a complexity of the order $O[N \times f(M)]$ with $f(M) < M$ while keeping the exact same response of the Farrow implementation. As an example, the implementation of the generalized Newton structure with $M = N = 4$ is shown in Figure 4.1, with $\bar{q_i} = q_{00} \times q_{11} \times \ldots \times q_{ii}$. From the figure, it is seen that the diagonal elements of Q

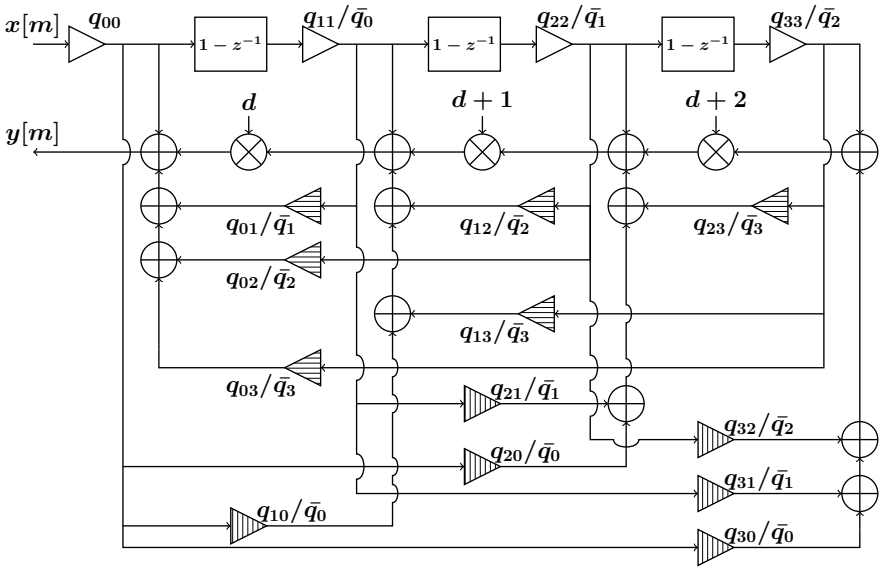


Figure 4.1: The generalized Newton structure of order 3 ($M = N = 4$)

are found on the $1 - z^{-1}$ delay line, the lower elements with respect to the diagonal form feed-forward loops (hashed vertically), and the upper elements form feed-back loops (hashed horizontally).

4.1.2 Farrow to Newton Transformation

Every Farrow implementation has a Newton structure equivalent, where each structure implements the same transfer function, however with different bases. The matrices P and Q can be related through the following matrix transformation [Lam16]:

$$\begin{aligned} H(z,\mu) &= \boldsymbol{\mu}^T P\ Z(z) \\ &= \boldsymbol{\mu}^T \left(T_d^T\ T_d^{-T}\right)\ P\ \left(T_z^{-1}\ T_z\right)\ Z(z) \\ &= \left(T_d\ \boldsymbol{\mu}\right)^T \left(T_d^{-T}\ P\ T_z^{-1}\right)\ \left(T_z\ Z(z)\right) \\ &= \boldsymbol{d}^T\ Q\ \bigtriangledown Z(z) \end{aligned} \tag{4.3}$$

With $T_d^{-T} = \left(T_d^T\right)^{-1}$ and the matrices T_d and T_z are the ones transforming the vectors $\boldsymbol{\mu}$ and $Z(z)$ to $\boldsymbol{d}$ and $\bigtriangledown Z(z)$ respectively. To develop the relation between the vector $\boldsymbol{d}$ and $\boldsymbol{\mu}$, the vector d^i is defined as:

$$\underset{1\times M}{d^i}^T = \begin{bmatrix} 1 & d & d^2 & \dots & d^{M-1} \end{bmatrix} \tag{4.4}$$

Then $\boldsymbol{d}$ can be developed as:

$$\begin{aligned} \underset{M\times 1}{\boldsymbol{d}} &= \underset{M\times M}{T_d^2} \times \underset{M\times 1}{d^i} \\ &= \underset{M\times M}{T_d^2} \times \underset{M\times M}{T_d^1} \times \underset{M\times 1}{\boldsymbol{\mu}} \end{aligned} \tag{4.5}$$

Where T_d^1 is the matrix with each row i containing the coefficients of $\left(\mu - \frac{M-1}{2}\right)^{i-1}$:

$$T_d^1[i,j] = \binom{i-1}{j-1} \left(-\frac{M-1}{2}\right)^{i-j} \tag{4.6}$$

And T_d^2 is the matrix with each row i containing the coefficients of the development of the rising power of d, that is $\prod_{n=0}^{i-1}(d+n)$. The relation between the rising power and the monomial elements of d can be developed as:

$$T_d^2[i,j] = S_j^{(i)} \tag{4.7}$$

With $S_j^{(i)}$ being the Stirling numbers of the first kind. On the other hand, the delay matrix $Z(z)$ is re-written in a backward difference base, resulting in $\bigtriangledown Z(z)$. The relation between the two vectors is then developed as:

$$\underset{N\times 1}{\bigtriangledown Z(z)} = \underset{N\times N}{T_z} \times \underset{N\times 1}{Z(z)} \tag{4.8}$$

Where T_z is the matrix with each row i containing the coefficients of $(1-z^{-1})^{i-1}$:

$$T_z[i,j] = \binom{i-1}{j-1}(-1)^{j+1} \tag{4.9}$$

The interest here is to find the coefficients of P using the ones of Q, to be used later in developing the frequency response of the generalized Newton structure in Section 4.4. Then the coefficients p_{ij} of the Farrow structure are related to the coefficients of the Newton structure q_{ij} through:

$$p_{ij} = \sum_{k=0}^{M-1}\sum_{p=0}^{N-1} T(i,j,k,p)q_{kp} \tag{4.10}$$

Where:

$$T(i,j,k,p) = T_z[p,j]\sum_{r=0}^{k} T_d^1[r,i]T_d^2[k,r] \tag{4.11}$$

4.1.3 Vector Form Transformation

The transformation presented above is linear, and therefore can be also written in a matrix linear transformation form as:

$$\underset{MN\times 1}{V_P} = \underset{MN\times MN}{T} \times \underset{MN\times 1}{V_Q} \tag{4.12}$$

Where V_P and V_Q are the vectors obtained from reshaping P and Q respectively, with the reshaping operation done in the column direction (meaning the elements of the same matrix column are consecutive in the vector). The matrix T is then found as:

$$T[i,j] = T_z\left[\left\lfloor\frac{j}{N}\right\rfloor, \left\lfloor\frac{i}{N}\right\rfloor\right] T_d^T[i\%M, j\%M] \tag{4.13}$$

With $T_d = T_d^2 \times T_d^1$, and $\lfloor x \rfloor$ being the biggest integer smaller or equal to x, and % being the modulo operator. The transformation form given in Equation (4.13) is more convenient than then one given in Equation (4.3), for developing the relation between the coefficients of Farrow and Newton structures in vectorial form. This transformation form is used later in Section 4.4 to apply filter optimization techniques.

4.2 Modified Newton Structure - Hermite Interpolation

In this section, the focus is on achieving the lowest possible hardware computational complexity by adapting the Newton structure to Hermite interpolation. Section 4.2.1 presents the definition of Hermite interpolation and presents the available implementations in the literature using variable fractional delay filters (V-FDF). Section 4.2.2 then develops how to derive

low complexity structures based on the Newton structure, implementing this interpolation method. Two example filter structures are also developed for different given interpolation parameters. Finally in Section 4.2.3, the new structures and their performance in terms of filtering and implementation complexity are compared with Lagrange and B-spline interpolation of the same order as references.

4.2.1 V-FDF Based on Hermite Interpolation

Before developing the new proposed structures, the Hermite interpolation is presented. This interpolation can be seen as a generalized case of Lagrange interpolation, where constraints of equality for the sample j, are applied not only to the function value, but also to the derivatives of order $i \in \{0, 1, 2, \ldots, p\}$, with p being the highest order derivative for which the equality constraints is applied. Therefore the Hermite interpolation constraints are:

$$y^{(i)}[j] = x^{(i)}[j] \quad / \quad i \in \{0, 1, 2, \ldots, p\} \tag{4.14}$$

Where $y^{(i)}[j]$ is the derivative of order i of the j^{th} sample of the function $y(t)$. This function represents the polynomial that is fitted through the $N = n+1$ input samples taken from $x(t)$, having a degree of $N_H = (p+1)(n+1) - 1$:

$$y(t) = \sum_{i=0}^{N_H} a_i t^i \tag{4.15}$$

Applying the $p+1$ equality constraints of Equation (4.14) to the N sample points, results in $N_H + 1$ equations with $N_H + 1$ unknown coefficients a_i. This forms a linear system, that can be solved to find the values of a_i as a function of $x^{(i)}[j]$. Then through a factorization by $x^{(i)}[j]$, the function $y(t)$ can be written as:

$$y(t) = \sum_{i=0}^{p} \sum_{j=0}^{N-1} \alpha_{i,j}(t) x^{(i)}[j] \tag{4.16}$$

where $\alpha_{i,j}(t)$ are polynomials of degree N_H. Therefore, the Hermite interpolation can be implemented as proposed in [Tse12], as p Farrow structures in parallel with their outputs summed up together. Each Farrow structure i implements the filtering of the N samples of derivative i. However this implementation requires the pre-calculation of the derivatives of the signal at the input samples. This adds to the complexity of the implementation by requiring more hardware.

A solution for low complexity V-FDF based on Hermite interpolation was proposed in [Soo11], where the derivatives were approximated using the central finite difference (CD) and the backward finite difference (BD). In [Soo11] the case of $n = 1$ and $p = 1$ was considered, which is the definition of the cubic Hermite interpolation. It was shown that using CD to find the derivatives $x^{(1)}[j]$ results in a better performance compared to using the BD expression. The 2^{nd} order central finite difference (CD2) is defined as:

$$x^{(1)}[j] = \frac{1}{2}(x^{(0)}[j+1] - x^{(0)}[j-1]) \tag{4.17}$$

while the 4^{th} order central finite difference (CD4) is:

$$\begin{aligned} x^{(1)}[j] = \frac{1}{12}(&-x^{(0)}[j+2] + 8x^{(0)}[j+1] \\ &- 8x^{(0)}[j-1] + x^{(0)}[j-2]) \end{aligned} \tag{4.18}$$

Equation (4.17) and Equation (4.18) show that sample points other than the ones for which the constraints are applied will be needed. Where for example to find $x^{(1)}[0]$ the sample

$x^{(0)}[-1]$ is required, even though equality constraints are not applied to $x^{(0)}[-1]$. Then the derivative approximation will have the effect of increasing the number of taps of the filter, where two and four extra points are needed, for CD2 and CD4 approximations respectively. Now by replacing Equation (4.17) or Equation (4.18) in Equation (4.16), it is possible to have an implementation using only one Farrow structure defined through:

$$y(t) = \sum_{j=0}^{N_\beta} \beta_j(t) x^{(0)}[j] \tag{4.19}$$

where $\beta_j(t)$ are polynomials of degree N_H, and N_β is the total number of points including the ones needed for derivative approximations. These polynomials define the implementation of Hermite interpolation using only one Farrow structure.

4.2.2 Proposed Low Complexity Hermite Interpolation Based V-FDF

Up to this point, the lowest complexity structure implementing Hermite interpolation is the Farrow structure presented in [Soo11]. The application of the Farrow-Newton transformation to the Hermite interpolation is investigated below. This transformation is applied to the matrix P_β containing the coefficients of $\beta_j(t)$.

The implementation developed in [Soo11] implements the classical Farrow structure [Far88]. However, the impulse response of the filter implementing Hermite interpolation is symmetric, and has a linear phase response. Then it is possible to reduce the complexity of the structures proposed by half through using the modified Farrow structure presented in [Ves96]. This is achieved by centering the polynomials $\beta_j(t)$ around zero though a time axis translation, where instead of evaluating $\beta_j(t)$ in the range of $t \in \left[\frac{N}{2} - 1; \frac{N}{2}\right[$, the polynomials are evaluated for $t \in [-1/2; 1/2[$. This results in having the matrix P_β with symmetric coefficients. This is an important condition to be respected in order to apply the transformations of [Lam16], where the matrix P used in Equation (4.1) has symmetric columns as it contains the coefficients of the symmetric polynomials defined over $t \in [-1/2; 1/2[$.

On another hand, the definition of Hermite interpolation is more complicated than the Lagrange or B-spline interpolations ones. For the latter interpolations, the number of polynomial pieces N that construct the impulse response is equal to the number of coefficients of each polynomial M. Then the matrices describing the Farrow implementation of these interpolations are always square matrices. This is not always the case for the matrix P_β containing the coefficients of Hermite interpolation, where the number of polynomial coefficients $(p+1)(n+1)$ can be easily different than N_β. This should be taken in consideration when building the transformation matrices, where in the previous cases presented in [Lam16], the matrices proposed had both dimensions of the same size. In the case of Hermite interpolation, the transformation of $\boldsymbol{\mu}$ to $\boldsymbol{d}$ can have different dimensions than that of $Z(z)$ to $\bigtriangledown Z(z)$, as it will be shown in the examples below.

Then applying the Farrow-Newton transformation on P_β results in the matrix Q_β containing the coefficients describing the Newton implementation of the Hermite interpolation. The definition of the Hermite interpolation is flexible, where three parameters are to be chosen: n, p, and the derivative approximation method. Since the interest is to find low complexity structures, the two implementations of the lowest complexity are presented below. Albeit their very low complexity, these two structures achieve a superior performance compared to B-spline and Lagrange interpolations of the same order.

4.2.2.1 Example 1 ($n = 1$, $p = 1$, CD2)

The first example considers the definition of an order 3 interpolation using $n = 1$, $p = 1$, and CD2 derivative approximation. In this case, the matrix P_β is a square matrix, since

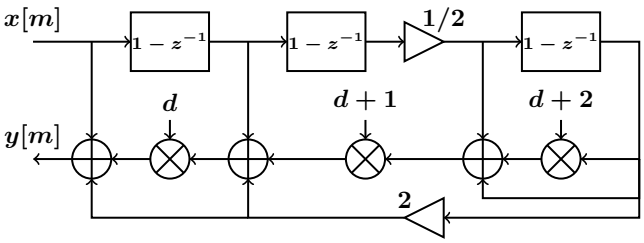


Figure 4.2: Modified Newton structure adapted to Hermite interpolation of order 3.

$(p+1)(n+1) = N_\beta = 4$. Following the development presented in Section 4.2.1, the matrix P_β is found as:

$$\underset{4\times4}{P_\beta} = \frac{1}{16}\begin{pmatrix} -1 & 9 & 9 & -1 \\ -2 & 22 & -22 & 2 \\ 4 & -4 & -4 & 4 \\ 8 & -24 & 24 & -8 \end{pmatrix} \tag{4.20}$$

Applying the $\boldsymbol{\mu}$ to $\boldsymbol{d}$ and $Z(z)$ to $\bigtriangledown Z(z)$ transformations, both of dimension 4, results in Q_β:

$$\underset{4\times4}{Q_\beta} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 \end{pmatrix} \tag{4.21}$$

Interpreting this matrix and maximizing the re-use of calculated values, results in the Newton structure adapted to Hermite interpolation of order 3 shown in Figure 4.2.

4.2.2.2 Example 2 ($n = 1$, $p = 1$, CD4)

The second case keeps $n = 1$, $p = 1$, but considers the CD4 approach for derivative approximation in place of CD2, resulting in an interpolation of order 5. In this case the resulting matrix P_β is no longer square, and has a dimension of 4×6:

$$\underset{4\times6}{P_\beta} = \frac{1}{96}\begin{pmatrix} 1 & -9 & 56 & 56 & -9 & 1 \\ 2 & -14 & 128 & -128 & 14 & -2 \\ -4 & 36 & -32 & -32 & 36 & -4 \\ -8 & 56 & -128 & 128 & -56 & 8 \end{pmatrix} \tag{4.22}$$

Now the $\boldsymbol{\mu}$ to $\boldsymbol{d}$ transformation is of dimension 4 while the $Z(z)$ to $\bigtriangledown Z(z)$ one has a dimension 6. This results in Q_β:

$$\underset{4\times6}{Q_\beta} = \begin{pmatrix} 1 & -1 & 0 & 0 & -1/6 & -1/6 \\ 0 & 1 & -1 & 0 & -1/6 & -1/6 \\ 0 & 0 & 1/2 & -1/2 & -1/12 & -1/12 \\ 0 & 0 & 0 & 1/6 & -1/6 & -1/12 \end{pmatrix} \tag{4.23}$$

The interpretation of this matrix in the same manner used for Equation (4.21), results in the structure shown in Figure 4.3.

4.2.3 Performance and Complexity

In this section, the performance and complexity of the proposed structures with the existing state-of-the-art implementations are compared.

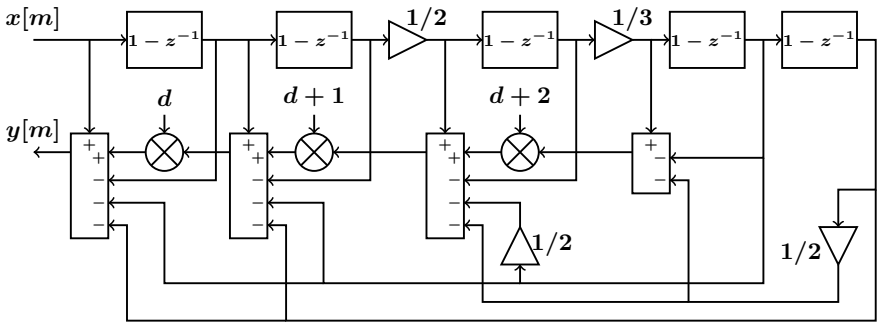
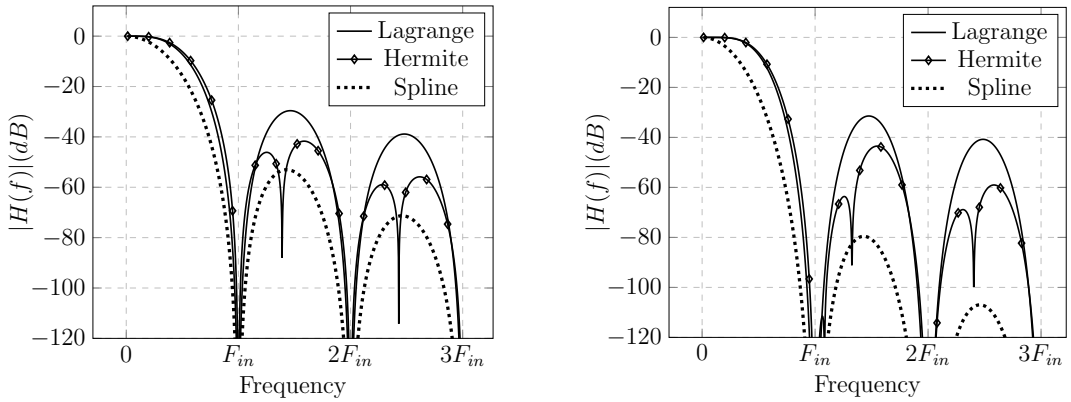


Figure 4.3: Modified Newton structure adapted to Hermite interpolation of order 5.



(a) Order 3 (b) Order 5

Figure 4.4: Frequency response of the discussed interpolation methods for order 3 and 5.

4.2.3.1 Performance Comparison

The previous examples taken above were both studied in [Soo11], where the amplitude and the phase delay responses of the discrete-time Hermite filter were developed and compared. It was shown that the Hermite filter has a better amplitude response compared to the Lagrange and B-spline interpolations based V-FDF filters, while having a phase delay response that is slightly inferior. Here, the continuous-time frequency response of the different interpolation methods is used, that is useful to study the ability of the filter to reject the interpolation images. The interpolation error studied from the continuous frequency domain can be developed by considering the continuous spectrum of the input discrete-time signal, that is periodic by F_{in} (the sampling frequency). The better the filter response can eliminate the signal images on multiples of F_{in}, the lower the interpolation error will be. In the case of the ideal filter response, these images are completely eliminated, and therefore there is no interpolation error. In Figure 4.4-a the frequency response of the Hermite interpolation can be seen compared to the ones of Lagrange and B-spline interpolations. The characteristics of these responses are detailed in Table 4.1.

Table 4.1: Frequency Response for order 3 interpolations

	Passband at -3 dB	Sidelobe Attenuation
Lagrange	0.384 F_{in}	-29.83 dB
B-spline	0.228 F_{in}	-53.91 dB
Hermite	0.403 F_{in}	-41.69 dB

The role of variable fractional delay filters is interpolation and not filtering signal compo-

nents. Therefore having the widest passband possible is important (ideally equal to $0.5F_{in}$) in order to be able to apply this interpolation correctly to all the frequency components of the signal. On the other hand maximal out of band attenuation is equally important in order to eliminate all signal images. The previous comparison table shows that Hermite interpolation offers the widest passband of $0.403F_{in}$, while having a better attenuation compared to the one of Lagrange interpolation, making it the better SRC solution. One may notice that B-spline offers the best side lobe attenuation, however this is at the cost of largely decreasing the passband, limiting the application of this type of interpolation.

The same analysis applies for the different interpolation methods of order 5 shown in Figure 4.4-b, where the differences are more visible than in the previous case. The frequency response characteristics for order 5 interpolations are shown in Table 4.2. It is rare to find a V-FDF implementing interpolation of an order higher than 5, which provides a sufficient performance for most of the interpolation applications.

Table 4.2: Frequency Response for order 5 interpolations

	Passband at -3 dB	Sidelobe Attenuation
Lagrange	0.410 F_{in}	-31.44 dB
B-spline	0.187 F_{in}	-79.56 dB
Hermite	0.422 F_{in}	-43.49 dB

Therefore in the general case, Hermite interpolation offers the maximum flatness in the frequency response passband, while improving the attenuation of the secondary lobes compared to Lagrange interpolation. For maximal out-of-band attenuation, the B-spline interpolation might be of interest in the case of narrowband signals or a high oversampling factor, due to the smaller passband of the frequency response. Hence for Hermite interpolation, the filter frequency responses of order 3 and 5 offer a compromise between the advantages of the two previous interpolation methods, by maximizing both passband flatness and side lobes rejection, thereby reducing the interpolation error when processing wideband signals. This section presented the advantage of the Hermite interpolation filtering response, next the complexity of the structures implementing Hermite interpolation is compared to that of the available structures in literature.

4.2.3.2 Complexity Comparison

What makes Hermite interpolation even more interesting is the low-complexity of its Newton implementation. Table 4.3 shows the hardware resources required to implement each of the three interpolation methods of order 3. Three types of multiplication are identified, and presented below in the increasing order of implementation complexity:

1. Shifters, that are multiplication by 2^i with $i \in \mathbb{Z}$.
2. Scalers, that are multipliers with a constant coefficient.
3. Multipliers, general case with two variable inputs.

The low complexity of the Hermite Newton implementation can be seen in the fact that a shift is needed in place of the scaler by 1/3, where the latter is a lot more costly to implement than a simple shift operation. Compared to the structure developed in [Soo11] based on the Farrow structure, our proposed structure implements the exact same interpolation while eliminating the need for any scalers. Next, the hardware complexity of the implementations of order 5 interpolation are presented in Table 4.4.

A clear advantage of the Hermite interpolation of order 5 in terms of complexity can be seen, with only 3 multipliers and 1 scaler needed, making it the lowest complexity option,

Table 4.3: Complexity of Order 3 Newton Implementations

	Register	Adder	Shifter	Scaler	Multiplier
Newton Implementation					
Lagrange	3	6	1	1	3
B-spline	3	9	1	1	3
Hermite	3	9	2	0	3
Farrow Implementation based on [Soo11]					
Hermite	3	10	5	2	3

Table 4.4: Complexity of Order 5 Newton Implementations

	Register	Adder	Shifter	Scaler	Multiplier
Newton Implementation					
Lagrange	5	10	3	2	5
B-spline	5	20	8	5	5
Hermite	5	18	3	1	3
Farrow Implementation based on [Soo11]					
Hermite	5	15	8	7	3

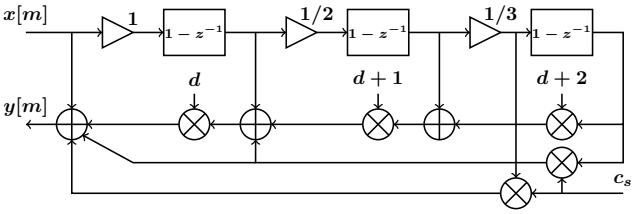
while keeping advantageous performance in terms of amplitude response. Compared to the equivalent structure based on the Farrow implementation, the advantage is more pronounced in this case, where 6 scalers are replaced with 3 adders, resulting in a more efficient SRC solution.

4.3 Re-configurable Newton Structure

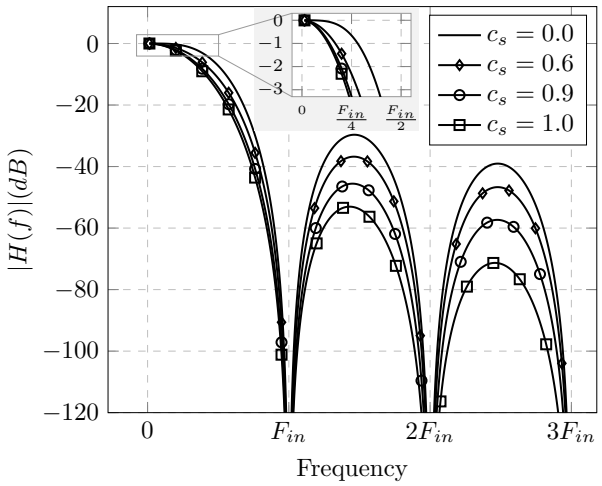
The work developed in [Lam16], and the development presented in the previous section, propose modifying the classical Newton structure to get different kinds of interpolation. However, each structure is limited to only one type of interpolation, since they are defined with fixed coefficients. In the next section, a new design approach is proposed to develop dynamically reconfigurable Newton structures, allowing the implementation of different interpolation methods using the same fixed structure. This has been made possible through the results of the previously developed generalization. The advantage of this reconfigurability is that it is controlled using only one input variable. Section 4.3.1 presents how by using the already developed modified Newton structures, it is possible to to obtain a reconfigurable structure by adding a variable parameter to the definition of the transfer function. Two concrete example structures are developed, and then the generalization of this method is discussed. In Section 4.3.2, the complexity of the new reconfigurable structures is compared with the fixed response ones. Finally in Section 4.3.3, two practical application examples of the reconfigurability of these structure for SRC are developed, where it is shown how the new structures allow adapting the filter response to the processed signal, maximizing thereby the SRC filtering performance.

4.3.1 Derivation of the Transfer Function

This section begins by presenting two examples of reconfigurable structures based on Spline and Hermite interpolation of order 3 respectively. Then the generalization for all kind of interpolation with any order is presented.



(a) The filter structure with the variable parameter c_s



(b) Different frequency responses for different values of c_s

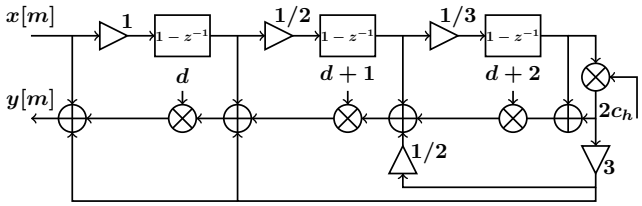
Figure 4.5: The Spline based reconfigurable Newton structure.

Reconfigurable Spline Based Newton Structure: The reconfigurability of the structure based on Spline interpolation is achieved through a linear combination of the matrices defining Lagrange and Spline interpolation. The Lagrange matrix Q_{Lag}, corresponding to the classical Newton structure, is considered as the base matrix. Let the matrix Q_{Spl} represent to the modified Newton structure for Spline interpolation. Taking the difference between Q_{Spl} and Q_{Lag} weighted by the coefficient c_s, and adding it to the base matrix, results in the matrix $Q_{Spl}^{c_s}$ defining the reconfigurable Newton structure based on Spline interpolation:

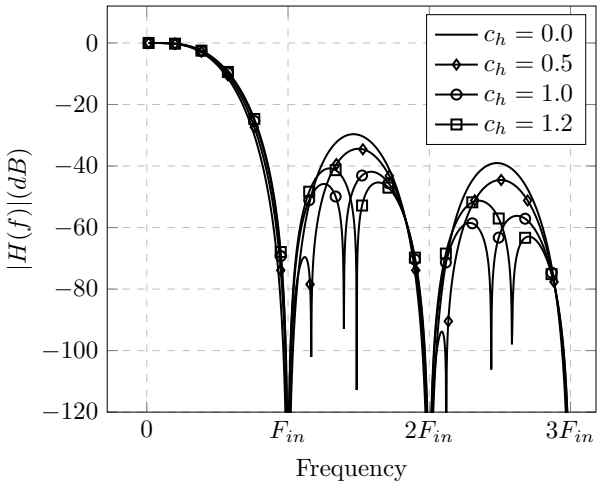
$$\begin{aligned} Q_{Spl}^{c_s} &= Q_{Lag} + c_s(Q_{Spl} - Q_{Lag}) \ , \quad c_s \in [0\ ;1] \\ &= \begin{pmatrix} 1 & 0 & c_s/6 & c_s/6 \\ 0 & 1 & 0 & c_s/6 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/6 \end{pmatrix} \end{aligned} \tag{4.24}$$

It can be seen in Equation (4.24) that only the off-diagonal elements are made variable. Figure 4.5-a shows the implementation structure of the transfer function $Q_{Spl}^{c_s}$ in Equation (4.24). The coefficient c_s can be set to any value between zero and one. Setting c_s to zero results in Lagrange interpolation, while a value of one results in Spline interpolation. Any value in between will result in having a different frequency response related to the responses of Lagrange and Spline interpolation. This is illustrated in Figure 4.5-b, with $c_s \in \{0, 0.6, 0.9, 1\}$. The parameter F_{in} is the sampling frequency of the input $x[m]$.

Reconfigurable Hermite Based Newton Structure: In a similar way to what is done above, the matrix $Q_{Her}^{c_h}$ defining the reconfigurable Newton structure based on Hermite in-



(a) The filter structure with the variable parameter c_h



(b) Different frequency responses for different values of c_h

Figure 4.6: The Hermite based reconfigurable Newton structure.

terpolation is developed as:

$$\begin{aligned} Q_{Her}^{c_h} &= Q_{Lag} + c_h(Q_{Her} - Q_{Lag}) \ , \quad c_h \in [0\ ; 1.25] \\ &= \begin{pmatrix} 1 & 0 & 0 & c_h \\ 0 & 1 & 0 & c_h \\ 0 & 0 & 1/2 & c_h/2 \\ 0 & 0 & 0 & 1/6 + c_h/3 \end{pmatrix} \end{aligned} \tag{4.25}$$

The implementation following Equation (2.44) results in the structure presented in Figure 4.6-a. For a coefficient c_h that is zero, the filter implements Lagrange interpolation. For a c_h with a value of one, the filter implements Hermite interpolation. In the case of Hermite based reconfigurable Newton structure, it is possible to allow c_h to go beyond the value of one, where this value dictates the position of the zeros in the side lobes. Starting from $c_h = 0$ the zeros of the side lobes will move from multiples of F_{in} toward the center of the side lobes when $c_h = 1.25$. This is shown in Figure 4.6-b for the case of $c_h \in \{0, 0.5, 1, 1.2\}$. Both structures presented above are dynamically reconfigurable, where it is possible to change the value of c_s or c_h at any moment.

Generalization: In the examples above, two most commonly used polynomial interpolation methods were considered. However, this design approach can be applied to any other types of polynomial interpolation. Moreover, the reconfigurability can be generalized for interpolations of a higher order in a similar way to what was presented previously, using Equation (4.24) and Equation (4.25). It is noteworthy that, the higher is the order of interpolation, the higher is the order of complexity of the reconfigurable structure. Regardless, the reconfigurable Newton structure presents a great advantage over the classical Farrow structure. This is because the reconfigurability is achieved through one input variable and a small

number of extra computations. This is not possible with the Farrow structure that requires storing a large number of coefficients in order to use different interpolation methods.

4.3.2 Complexity Comparison

In the last section, two low complexity reconfigurable V-FDF solutions based on the Newton structure are presented. The developed examples based on Spline and Hermite interpolation of order 3 have structures with a comparable complexity to the one of the non-reconfigurable variants. Table 4.5 shows a comparison of the hardware elements needed for the different implementations being the Newton structure of order 3. It can be seen that making Spline Newton implementation reconfigurable only requires two additional multipliers. This results in a structure implementing both Spline and Lagrange interpolations, in addition to any related response to these two. For Hermite interpolation, reconfigurability is achieved by adding one adder, one shifter, two scalers, and only one multiplier. This shows the advantage of such structures, which offer reconfigurability with a minor increase in complexity.

Table 4.5: Complexity comparison of reconfigurable Newton structures

	Register	Adder	Shifter	Scaler	Multiplier
Lagrange	3	6	1	1	3
Spline	3	9	1	1	3
Reconfigurable Lagrange/Spline	3	9	1	1	5
Hermite	3	9	2	0	3
Reconfigurable Lagrange/Hermite	3	10	3	2	4

4.3.3 Application Examples

The two examples of presented structures have different application interests. Each structure is better adapted to a certain application scenario that is explained below. To judge the filtering performance, the reference digital SRC operation model U-F-D (c.f. Chapter 2) is considered. The objective of the filtering step is to maximize the rejection of the up-sampled signal images on multiples of F_{in} [Cro83, Chapter 2]. Therefore, the higher is the attenuation of the side lobes of the frequency response, the better is the filtering performance. Attention should also be given to have a sufficiently large filter passband, to avoid altering the signal.

For the reconfigurable Newton structures based on Spline interpolation, the case of Lagrange interpolation ($c_s = 0$) presents the extreme case of maximal platitude in the passband [Her92]. While Spline interpolation ($c_s = 1$) presents the extreme case of maximal rejection by the side lobes [Fra09]. In fact, the choice of c_s lies in a compromise between the width of the passband and the rejection of the side lobes. Therefore to maximize the filtering performance, the coefficient c_s is increased as the band of the signal gets narrower (to maximize the side lobes rejection level). As a practical example, a multi-standard operation between Sigfox and LoRa technologies is considered. If the system is processing a wideband LoRa signal of 250 kHz, then the coefficient c_s is set to be close to zero in order to have a passband that is sufficiently large. While in the case of a narrowband Sigfox signal of 100 Hz bandwidth, the coefficient c_s is then set to be close to one, thus maximizing the side lobes attenuation.

For the reconfigurable Newton structures based on Hermite interpolation, the application is different. In this case, all the variations of the frequency response achievable with this structure have the property of maximal flatness at $f = 0$ Hz. Therefore, the choice of the response does not depend on the signal's bandwidth. The interest of this structure is in the ability to position the zeros of the side lobes, and most notably for the first side lobe that

has the lowest attenuation. This structure can be very useful in cases where multiple signals exist in the filtered bandwidth, where the secondary signals have higher frequency components than the main signal. In this case the aliasing effect caused by the secondary signals can be minimized by positioning the zero to fall on top of the up-sampling images of these secondary components.

4.4 Optimized Newton Structure

The closed form expressions of the Farrow to Newton transformation were developed in Section 4.1. These expressions are used in this section to develop the closed form expression of the frequency response of the generalized Newton structure. This latter expression then enables the application of filter optimization methods to design generalized Newton structures with an optimized frequency response. The application of the optimization is discussed, and it is shown through a practical implementation example how it is possible to develop optimized Newton structures having comparable performance to the optimized Farrow structures while reducing the implementation complexity by about 50%.

4.4.1 Generalized Newton Structure Frequency Response

Both the Farrow and the Newton structures implement a polynomial based filter response $h(t)$, that is defined with N polynomial pieces $f_n(t)$ with $n \in \{0, 1, \ldots, N-1\}$, each of length T_s and degree $M-1$, where T_s is the input sampling period. The development of the polynomial pieces normalized with respect to T_s using the term $\tau = t/T_s$ gives:

$$f_n(\tau) = \begin{cases} \sum_{m=0}^{M-1} p_{mn} \left(\tau - \frac{1}{2}\right)^m & 0 \leq \tau < 1 \\ 0 & Otherwise \end{cases} \tag{4.26}$$

These pieces are then used to construct the normalized impulse response:

$$h(\tau) = \sum_{n=0}^{N-1} f_n(\tau - n) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p_{mn} \psi_m(n, \tau) \tag{4.27}$$

Where:

$$\psi_m(n, \tau) = \begin{cases} \left((\tau - n) - \frac{1}{2}\right)^m & n \leq \tau < (n+1) \\ 0 & Otherwise \end{cases} \tag{4.28}$$

Applying the Fourier transform on $h(t)$ as it was done in [Hun08] results in the normalized frequency response $H(\omega)$ with $\omega = 2\pi f T_s$:

$$\begin{aligned} H(\omega) &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p_{mn} \left[\int_n^{n+1} \psi_m(n, \tau) e^{-j\omega\tau} d\tau \right] \\ &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p_{mn} \Psi_m(n, \omega) \end{aligned} \tag{4.29}$$

Where:

$$\Psi_m(n, \omega) = \frac{m!}{e^{j\omega n}} \sum_{k=0}^{m} \frac{(-1)^{m-k} - e^{-j\omega}}{(2^{m-k})\ (m-k)!} \left(\frac{1}{j\omega}\right)^{k+1} \tag{4.30}$$

Replacing Equation (4.10) in Equation (4.29) results in the closed form expression of the frequency response of the generalized Newton structure:

$$H(\omega) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \Psi_m(n, \omega) \sum_{k=0}^{M-1} \sum_{p=0}^{N-1} T(m, n, k, p) q_{kp} \tag{4.31}$$

Through this expression, it is possible now to apply filter optimization techniques on the frequency response $H(\omega)$ of the generalized Newton structure as it will presented in the next sections.

4.4.2 Filter Optimization Application

The advantage of the Newton structure implementation over the Farrow structure was presented in Section 4.1, which considers the case of common interpolation methods such as Lagrange, Spline and Hermite interpolations. This section investigates the application of filter optimization methods to develop generalized Newton structures with an optimized transfer function. Through the optimization techniques, it is possible to add constraints on the values that the matrix coefficients can take, allowing thereby the customization of the optimized filter structure. This section starts by presenting the Farrow structure optimization method proposed in [Hun08], before developing how it is possible to extend this method to apply optimization on the generalized Newton structure.

Farrow Structure Optimization: Filter optimization consists of finding the filter coefficients minimizing the error between the ideally desired and the practically implemented frequency responses. Coefficients optimization for FIR filters is well developed in the literature, and the case of optimization for digital interpolation filters was studied in [Laa96]. In the case of polynomial based filters, such as the Farrow structure, an added complexity arises from the way the transfer function is defined. Since the filter coefficients are calculated using polynomial pieces, the optimization method should be adapted to build optimized polynomial impulse response instead of fixed filter coefficients. A proposed solution for this problem was developed in [Hun08] using a minmax optimization method. Approximation in the minmax context is defined as minimizing the differential error between the filter to be designed with the frequency response $H(f)$, and the desired filter response $D(f)$. The ideal response is normally chosen as an ideal low-pass filter response with a cut-off frequency f_c. The differential error between the ideal and the designed response is written as:

$$E(f) = W(f)\,[D(f) - H(f)] \tag{4.32}$$

Where $W(f)$ is the weight function, which defines the importance of optimizing the approximation at a certain frequency f. This method gets its name from how the error is minimized, where in this case the objective is to perform a minmax minimization of $E(f)$ with respect to the filter coefficients:

$$|E(f)| < \delta \tag{4.33}$$

With delta being a threshold of when the optimization should stop. Minmax being a linear optimization problem, it is possible to use a linear program in order to solve the problem for the filter coefficients $c_m(n)$. Where the linear program is defined as:

$$\min_x \; g^T x \quad / \quad \begin{cases} Ax \leq b \\ A_{eq}x = b_{eq} \\ lb \leq x \leq ub \end{cases} \tag{4.34}$$

With $A, A_{eq}, b, b_{eq}, x, g$ are matrices defining the optimization problem. lb and ub are limits on the values that x can take in order to accelerate the optimization. A general explanation of the roles of each matrix or vector is briefly presented below:

- x: A vector, containing the coefficients to be optimized $c_n(m)$ and the threshold δ
- g: A vector, containing only one non-zero element, which specify that δ is the parameter to minimize

- A: A matrix, presenting the relation between the coefficients $c_n(m)$ and the frequency response $H(f)$ defined in Equation (4.29)
- b: A vector, holding the values of the desired frequency response $D(f)$
- A_{eq}: A matrix defining the continuity conditions of the impulse response $h(t)$ and its derivatives
- b_{eq}: A vector, with all zero elements

The linear program can be implemented in any programming language or tool that support solving such problems (LINDO^{TM} LINGO, Python CVXOPT module, MathWorks® Optimization Toolbox, ...). The main parameters of the program can be summarized as the following:

1. The desired response $D(f)$ which is defined over a limited discrete frequency range (i.e. passband: $f \in [0, f_c[$ and stopband: $f \in [f_c + \Delta f, 4F_{in}[$ with Δf being the transition band)
2. The weight function $W(f)$ which is defined over the support of $D(f)$ (i.e. $k_{pass} = 1$ over the passband and $k_{stop} = 10$ over the stopband)
3. The PBF parameters N and M (i.e. $N = 6$ and $M = 5$)

Executing the linear program after defining all the matrices and parameters, will give the coefficients that best approximate the desired frequency response $D(f)$. It is known that these coefficients correspond to the optimal solution, since a linear program can be solved for a global minimum.

Generalized Newton Structure Optimization: In the case of the generalized Newton structure, constraints may be imposed on most of the coefficients of the matrix Q to be equal to zero. Then the optimization is applied on the non-zero coefficients. This offers structures of a very low complexity that are capable of achieving relatively similar filtering performance to that of the more complex Farrow structure. The optimization method proposed below is an extension of the Farrow structure optimization presented in the last section. This extension is achieved by modifying the optimization parameters and variables following the closed form expression of the generalized Newton structure frequency response of Equation (4.31). Starting from the linear phase filter optimization method developed in [Hun08] for the Farrow structure, the optimization problem was formulated as follows:

$$\min_{x} g^T x, \quad \text{such that} \begin{cases} Ax \leq b \\ A_{eq}x = b_{eq} \end{cases} \tag{4.35}$$

In order to adapt this optimization method to the generalized Newton structure, three steps are required:

1. First, the coefficients of the Newton structure that are to be optimized are selected, defining thereby the resulting filter structure. Then Equation (4.13) is reduced to include only the non-zero elements in calculation and only the first half of the coefficients of V_P, resulting in the matrix T_{opt} that is of dimensions $\left(\frac{MN}{2} + 1 \times \frac{MN}{2} + 1\right)$.
2. Second, the matrix T_{opt} is used to modify the matrix A following Equation (4.31) to have the description of the generalized Newton structure frequency response.
3. Third, the matrix A_{eq} and the vector b_{eq} are used to impose constraints to have a symmetrical impulse response, and to add the zero constraints on the unneeded coefficients in Q.

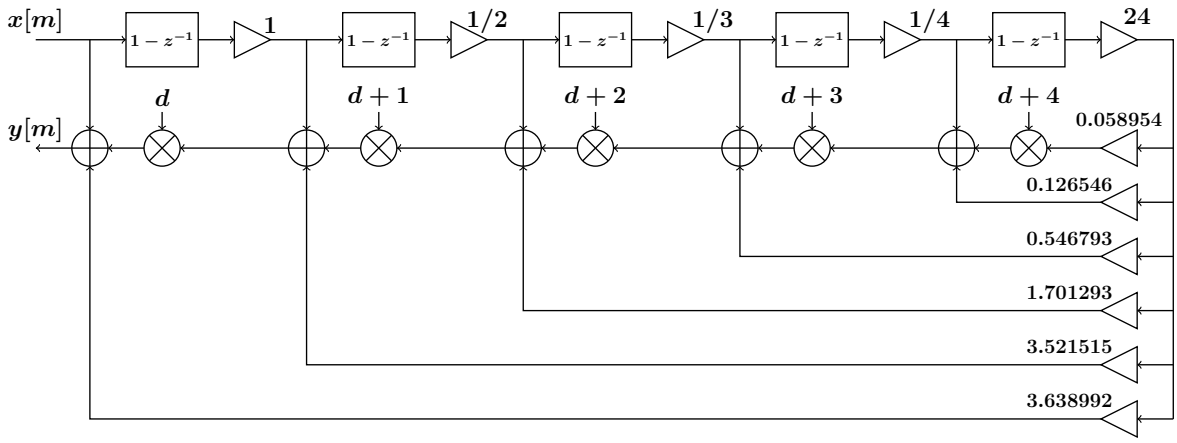


Figure 4.7: The optimized Newton structure with $N = M = 6$

In the optimization example below, only the coefficients of the matrix Q that are on its diagonal or its last columns are selected for optimization. Then the optimization method is applied for both cases of Farrow and Newton structures, with $M = N = 6$, a passband of $0.15F_{in}$, and a stopband starting from $0.85F_{in}$. The optimized Newton matrix Q found is:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 3.638992 \\ 0 & 1 & 0 & 0 & 0 & 3.521515 \\ 0 & 0 & 1/2 & 0 & 0 & 1.701293 \\ 0 & 0 & 0 & 1/6 & 0 & 0.546793 \\ 0 & 0 & 0 & 0 & 1/24 & 0.126546 \\ 0 & 0 & 0 & 0 & 0 & 0.058954 \end{pmatrix} \tag{4.36}$$

Following the implementation model presented in Figure 4.1, the resulting structure that is obtained from the optimized matrix Q is shown in Figure 4.7. The frequency responses of the filters obtained by optimization are shown in Figure 4.8. The optimized Newton structure keeps the advantage of having strong zeros at multiples of F_{in}. In addition, the side lobes rejection level achieved by optimization is around -60 dB, which is comparable to the rejection level of the side lobes of the optimized Farrow response. In terms of sample rate conversion applications, this means that the aliased images filtered by the side lobes are attenuated by around 60 dB in both cases. Complexity wise, Table 4.6 shows a comparison of the two optimized implementation structures, with the classical Newton structure for Lagrange interpolation of order 5 as a reference. It is seen that the optimized Newton structure has an added complexity around 50% compared to the classical Newton structure, while in the case of the optimized Farrow structure the added complexity is more than 200%. This shows the interests of applying optimization methods on the generalized Newton structure, which allow achieving high filtering performance with an important reduction of complexity relatively to the Farrow structure.

Table 4.6: Complexity comparison of the optimized structures

	Register	Addition	Multiplication
Newton Lagrange	5	10	7
Optimized Newton	5	15	12
Optimized Farrow	5	41	23

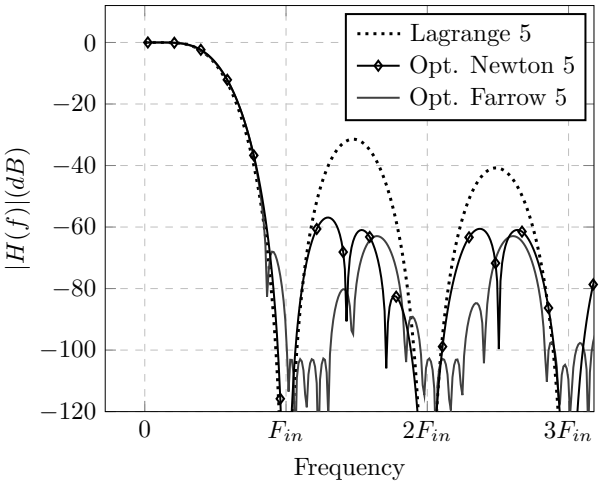


Figure 4.8: The frequency responses of Farrow and Newton optimized structures

4.5 Conclusion

We introduced in this chapter new SRC filters based on the Newton structure, by using the Farrow to Newton transformation. First, the modified Newton structure implementing Hermite interpolation possesses lower complexity than both the classical and the modified B-spline structure for a given order, while offering a better compromise between passband flatness and side lobes rejection. Second, by linearly combining the transfer functions of a modified Newton structure with the classical one, reconfigurable structures are developed that keep a low complexity implementation, while offering the possibility to adapt the filter response to the processed signal type. Finally, the application of filter optimization methods became possible with the development of the generalized Newton structure frequency response. This design approach allows designing low-cost implementations based on the Newton structure with a similar filtering performance offered by an optimized Farrow structure of the same order. The complexity of the newly developed filter structures were compared in this chapter using the number of calculation operations. To compare the practical real-world complexity of these SRC solutions, they need to be implemented in hardware. Before proceeding with the implementation, this thesis addresses first the problem of quantization, which is responsible of defining the bit width of the filter signals and parameters. The quantization problem does not only concern the SRC filters, but also concerns all of the hardware modules in the DFE. The next chapter investigates the state-of-the-art quantization methods, and then proposes a new quantization approach that is optimal yet simple to apply.

Chapter 5

Optimal Fixed-Point Quantization

Contents

5.1 Hardware Quantization 88
5.1.1 Quantization Fundamentals 88
5.1.2 Literature of Quantization Methods 94
5.2 Optimal Analytical Quantization 96
5.2.1 Integer Part Quantization 97
5.2.2 Fractional Part Quantization 98
5.3 Application Example 106
5.3.1 CIC Module Quantization 107
5.3.2 Newton Module Quantization 109
5.4 Conclusion 111

For a complete study of the SRC solutions for the DFE, it is important to develop the hardware implementation of the newly developed SRC filter structures, and to compare them with the currently available SRC solutions. However, before proceeding with the hardware implementation, the digital circuit signals and parameters need to be quantized. Quantization consists of allocating a certain number of bits to represent a value in the digital circuit. This step of hardware design needs special attention since it defines on one hand the calculation precision of the total implementation, and on the other hand it has an important impact on the implementation complexity. Therefore a bad quantization design of only one processing module in the DFE will have negative effects on the total system's efficiency. This is because any introduced quantization error will propagate through the system and cannot be filtered or compensated once it is mixed with the useful signal. This chapter aims at developing a quantization solution that is simple yet capable of guarantying a predefined calculation precision level. Section 5.1 starts by presenting the fixed point quantization scheme, and provides a quick discussion of the available quantization methods in the literature. Section 5.2 addresses the lack of a simple quantization method in the literature, and then proposes an optimal analytical quantization method, with a simple and clear guide on how to apply this method to quantize an arbitrary filter. Section 5.3 then develops a practical quantization example to demonstrate the capacity of the developed method to define quantization schemes with a guaranteed precision performance level. The method is applied to two SRC filter structures. The conclusion is finally provided in Section 5.4.

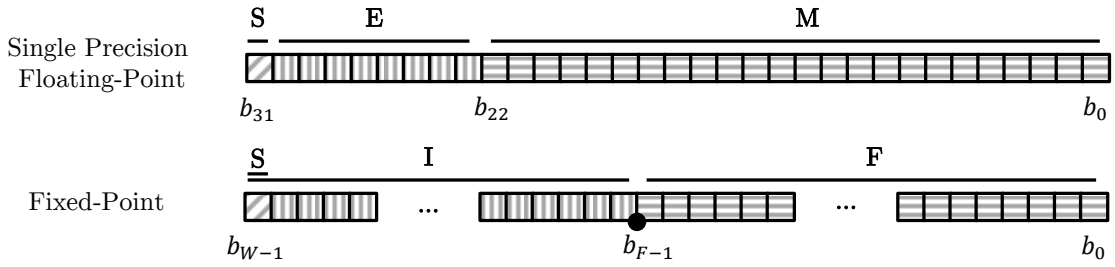


Figure 5.1: Floating-point and fixed-point quantization

5.1 Hardware Quantization

In a digital hardware implementation, the signals and parameters are presented with a finite precision. This precision depends on the number of bits used to represent the signal's value. This is called quantization in the signal processing literature. Choosing the appropriate number of bits to be used for each signal is a delicate task, whose objective is always to use the minimal hardware resources while respecting a given precision constraint. In some implementation, feasibility constraints may also apply, for example when certain hardware implemented operators have a limited number of bits at their inputs. In this section, the fundamentals of quantization are presented in Section 5.1.1. Section 5.1.2 then sums-up the literature of quantization methods, and shows what motivated the development of our own quantization method in this work.

5.1.1 Quantization Fundamentals

This section aims at providing a quick coverage of the quantization basics. More detailed presentations can be found in the mentioned references.

5.1.1.1 Schemes and Notations

The two most used quantization schemes in hardware implementations are fixed-point and floating-point quantization. Each scheme has its own advantages and drawbacks, that make it better adapted to certain applications. The two schemes are presented in Figure 5.1.

Floating-point : The floating-point quantization scheme presented here corresponds to the IEEE 754 single precision format using 32 bits. In this case, the value is presented using three variables: the sign (S, 1 bit), the exponent (E, 8 bits), and the mantissa (M, 23 bits). The quantized value is then found as:

$$x = (-1)^{b_{31}} \times 2^{(E-127)} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \quad \text{with} \quad E = \sum_{i=0}^{7} b_{23+i} 2^{i} \tag{5.1}$$

This quantization allows presenting values between $\pm 1.18 \times 10^{-38}$ and $\pm 3.4 \times 10^{38}$. The presentation precision is fixed at 24 bits, which is the mantissa length plus the implicit added one. The IEEE 754 standard specifies specific coding to represent special values such as zero, infinity, and others.

Fixed-point : The fixed-point quantization scheme can be interpreted in different ways, and we discuss here the two most common approaches. In the first case, all the W bit elements are used to represent the signal value that is always positive. The bits are separated by a fixed-point, with the I most significant bits (MSBs) representing the integer part, and the F

least significant bits (LSBs) representing the fractional part. This quantization is noted as $uI.F$. Then the quantized value and the dynamic range (DR) are found as:

$$x = \left\lfloor \sum_{i=0}^{W-1} b_i 2^{i-F} \right\rfloor \quad \in \left[0; 2^I - 2^{-F}\right] \tag{5.2}$$

In the second case, the MSB of the quantized value is considered to be a sign bit, and the negative values are considered to be coded using the two's complement approach. The notation used for this quantization is $sI.F$. In this case the quantized value and the DR are:

$$x = \left\lfloor \sum_{i=0}^{W-2} b_i 2^{i-F} \quad - \quad b_{W-1} 2^{(I-1)} \right\rfloor \quad \in \left[-2^{I-1}; 2^{I-1} - 2^{-F}\right[\tag{5.3}$$

In both cases, for a maximum value x_{max} and minimal value x_{min} quantized using W bits, the quantization step q if found as:

$$q = \lfloor x_{max} - x_{min} \rfloor 2^{-W} = 2^I \times 2^{-W} = 2^{-F} \tag{5.4}$$

Comparison : Floating-point representation can implement a larger DR than fixed-point as the number of bits increases, making it advantageous for general-purpose computing. However the floating-point quantization scheme is more complicated, increasing thereby the hardware implementation complexity of its arithmetic operators. This is tolerable for general purpose processors (GPP) implementation, however when implementing signal processing modules in hardware or using low-cost micro-controllers and GPPs, fixed-point quantization offers a much more efficient solution that is better adapted for low-cost implementations. And since the objective is to develop ASIC implementations, the two's complement fixed-point quantization scheme is used in this thesis for implementing the DFE hardware modules.

5.1.1.2 Quantization Rules

When performing two's complement fixed-point quantization, some rules should be defined first. The first rule considers if truncation or rounding to nearest value should be used when removing bits. The second defines the quantization behavior in the case of overflow that can be handled with saturation or wrapping.

Truncation and Rounding: For a given signal with a continuous amplitude level, two approaches are possible. In the first case, the value can be quantized to the first quantization level achievable by the quantization step q. In this case the quantization error is uniformly distributed in the range $[0; q]$, resulting in an average constant error $\mu_e = q/2$. This constant error can be eliminated by rounding the quantization to the nearest value instead of taking the nearest lower quantization value. In this case the operation is called rounding, and the error is uniformly distributed in the range $[-q/2; q/2]$. Truncation and rounding of signals with a discrete amplitude level are slightly different when the number of bits eliminated is small. For more details on this subject, the reader is referred to chapter one of [Roc06].

Saturation and Wrapping: When a signal's value goes beyond the range that can be represented by the quantization scheme, two options are possible to handle this overflow. The first option is to block the signal's value at its maximum or minimum level, which is known as saturation. The second option is to wrap-around the value, which is seen as a circular representation of the signals' range with the maximum and minimum levels being attached. For example going beyond the maximum level x_{max} by an amount α, the final value will be $\alpha + x_{min}$. In hardware implementation of two's complement quantization, the wrapping

behavior is intrinsic. To implement saturation extra logic resources are required to provide this protection against overflow. However, saturation is not always the solution, where some signal processing applications rely on wrapping for their correct operation (see CIC filters in Chapter 2). Therefore the choice between the two options depends on the operation in question.

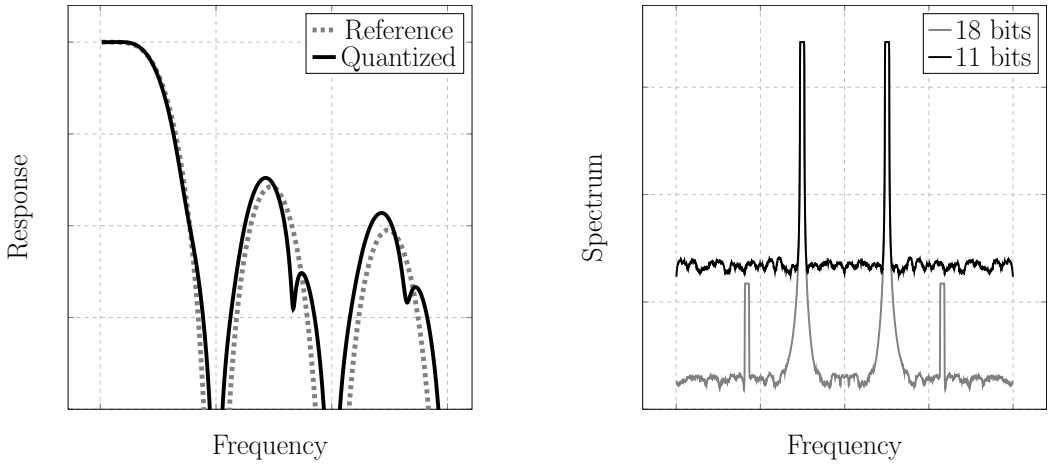
Bitwidth Evolution: In the majority of cases, we want to avoid any overflow beyond the quantization range. When performing arithmetic operations, the signal's value will evolve. To ensure that a sufficient number of bits is used to represent the signal, the designer should study in advance the module structure and the input signal dynamics. The methodology is studied in more details in Section 5.2. However, there are certain general rules that should be used in this study. When performing addition or subtraction of two signals x_1 and x_2 quantized on $sI_1.F_1$ and $sI_2.F_2$ respectively, the output of the operation needs to be presented on $s[max(I_1, I_2) + 1].[max(F_1, F_2)]$. In the case of $x_1 \times x_2$, the operation output is quantized on $s[I_1 + I_2].[F_1 + F_2]$. Division operations are transformed into multiplication by inverting the divisor with sufficient precision.

5.1.1.3 Quantization Error Modeling

To quantify the quantization performance, the effects of quantizing the parameters and signals should be first understood. This section presents these effects and models the quantization error in order to develop the signal to noise ratio (SNR) expression.

Parameters and Signals Quantization: When developing the quantization of a hardware implementation of an operator, two aspects of the circuit are to be considered: the parameters of the operation, and the signals to be processed by the operation. The parameters are the fixed coefficients that are used in the signal processing models (e.g. the coefficients of an FIR filter, or a constant offset coefficient). Quantizing the parameters mainly affects the response of the system. An example on a low-pass FIR filter is shown in Figure 5.2-a, where it is seen how the quantization of the filter coefficients modifies the frequency response. The quantization of filter parameters has been covered extensively in the literature [Cro75][Won91][Qia06]. The second type of quantization that concerns the signals is more delicate, and it is the interest of this chapter. The error generated by the quantization of the filter parameters is deterministic and it is managed depending on the tolerated distortion of the filter response. However the signal quantization error is random, and needs to be modeled in order to be studied. The effects of this random error are seen on the frequency spectrum of the signal, where by using fewer bit elements, the noise level caused by the quantization error will be higher, therefore reducing the SNR of the signal in question. In wireless telecommunication receivers, the noise level should be kept to a minimum in a DFE in order to manage to capture signals of low power. This is illustrated in Figure 5.2-b, where it can be seen that by quantizing the signal on 11 bits, the low power components are lost compared to an 18 bits quantization. The quantization of the parameters and signals are usually done separately. At first the parameters quantization is developed to respect certain criteria on the frequency response. The signals quantization is developed afterwards to respect precision criteria that depend on the application.

Quantization Models: To understand the effects of quantizing the signals values in a signal processing system, a model for the quantization process needs to be developed. In the literature many models that estimate the quantization noise have been proposed. Detailed models use a statistical approach to develop an expression of the power spectral density (PSD) of quantization noise [Osb90]. However for common signal processing applications, the



(a) The deformation of a filter response due to the quantization of its coefficients

(b) The quantization noise level is inversely proportional to the number of bits used

Figure 5.2: Effects of parameters and signals quantizations

Widrow model is widely adopted for its simplicity and practicality, while being sufficiently accurate [Wid56][Wid61]. This model states that the quantization process can be modeled as a linear operation, where the quantized output signal y is the sum of the input signal x and a random variable error e. This is true if the condition developed in [Wid96] is respected. In this model the quantization noise e follows a uniform distribution. The same model is developed in [Sri77] with more relaxed conditions. The correlation between the quantization noise and the input signal is studied in [Zöl08][Lip92], and the conditions for when both signals are considered uncorrelated are developed.

5.1.1.4 Measuring the Quantization Error

In our work we consider the Widrow model for its simplicity and practicality, while being sufficiently accurate. The quantization noise is supposed to be a random continuous signal with a uniform distribution. This section develops the theoretical expression of the quantization error power, the appropriate way of measuring it practically, and the performance measures used to qualify the quantization.

Quantization Error Power: The power spectral density (PSD) is used to evaluate the quantization noise level. The PSD is useful because it normalizes the quantization error power over the bandwidth (BW) it is measured on. Mathematically, the relation between the PSD $S_x(f)$ of a band limited signal x and its autocorrelation function $R_x(\tau)$ is given as:

$$R_x(0) = E\left\{x(nT)^2\right\} = \int_{F_s} S_x(f)df \tag{5.5}$$

With $F_s = 1/T$ being the sampling frequency of the signal x. By considering the quantization error signal e that follows a random uniform distribution:

$$e(nT) = y(nT) - x(nT) \ \sim \ \mathcal{U}\left[-\frac{q}{2}\ ;\ +\frac{q}{2}\right] \tag{5.6}$$

To find the quantization error PSD $S_e(f)$ we find the autocorrelation function first:

$$R_e(mT) = E\left\{e(nT)\bar{e}(nT - mT)\right\} = E\left\{e^2(nT)\right\}\delta(mT) = \frac{q^2}{12}\delta(m) \tag{5.7}$$

The first equality is possible because the quantization error is random and is only correlated with itself at any given instant. The second equality considers that $E\left\{e^2(nT)\right\} = q^2/12$ which is the variance of a random variable with a uniform distribution. Then the PSD $S_e(f)$ for the discrete error can be found using Equation (5.5) as:

$$R_e(0) = \frac{q^2}{12} = \int_{F_s} S_e(f)df \quad \Rightarrow \quad S_e(f) = \frac{q^2}{12F_s} \tag{5.8}$$

Statistical Measurement: To verify the exactitude of the quantization model used, or to simply measure the quantization noise, the appropriate measuring procedure should be defined that corresponds with the definitions presented previously. Measuring the quantization noise level follows the same procedures of measuring any other signal, which is based on the discrete Fourier transform (DFT). The presentation below is developed in more details by Heinzel et al. in [Hei02]. In summary, the steps to be followed are the following:

1. Acquiring data: Take N samples x_n of a certain signal $x(t)$.

2. Applying a window function: As the DFT assumes the periodicity of the signal, a window function W is needed to reduce spurious frequencies caused by non-periodicity.

$$\mathring{x}_n = x_n \times W_n \tag{5.9}$$

3. *Calculating the DFT:* Using the N samples $\mathring{x}_n$, the DFT is efficiently calculated by many FFT algorithms as:

$$X_k = \sum_{n=0}^{N-1} \mathring{x}_n e^{-j2\pi \frac{k}{N} n} \tag{5.10}$$

4. *Finding the PSD:* Using the DFT result, $S_x(k)$ can be found using:

$$S_x(k) = \frac{|X_k|^2}{F_s S_W} \qquad \text{with} \qquad S_W = \sum_{n=0}^{N-1} W_n^2 \tag{5.11}$$

   Where S_W takes the role of the normalization coefficient that takes into account the number of samples and the window function response.

It is possible to verify theoretically the relation between the measured noise and the theoretical estimation when considering the case of a uniformly distributed random variable. This can be done by considering the discrete Parseval theorem stating that:

$$\sum_{k=0}^{N-1} |X_k|^2 = N \sum_{k=0}^{N-1} |\mathring{x}_k|^2 \Rightarrow NE(|X_k|^2) = NS_W E(|x_k|^2) \tag{5.12}$$

Taking the expected value of Equation (5.11), and replacing $E(|X_k|^2)$, we find:

$$E(S_x(k)) = \frac{1}{F_s S_W} E(|X_k|^2) = \frac{E(|x_k|^2)}{F_s} = \frac{q^2}{12F_s} \tag{5.13}$$

By making the number of samples goes to infinity, the law of large numbers (LLN) shows that the mean of the measured PSD should tend to the estimated PSD:

$$\overline{S_x}(k) = \frac{1}{N} \sum_{i=0}^{N-1} S_x(i) \quad \xrightarrow[N\to\infty]{} \quad E(S_x(k)) = \frac{q^2}{12F_s} \tag{5.14}$$

5.1.1.5 Quantization Performance

The quantization performance is evaluated with several measures, with the ones most prominent to this study presented below.

Signal to Noise Ratio (SNR): The SNR represents the quality of the signal considering the noise introduced by quantization. This measure is defined as:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{5.15}$$

This measure is also called signal to quantization noise ratio (SQNR) in literature to differentiate it from SNR measures related to other sources of noise. It is common to study the SNR as if the signal was a sine wave (where every signal can be decomposed into sine waves by the Fourier transform), so by taking the input:

$$v(t) = 2^{I-1} sin(2\pi ft) \Rightarrow P_{signal} = P_v = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |v(t)|^2 dt = \frac{4^{I-1}}{2} \tag{5.16}$$

With the quantization error having the power given by:

$$P_{noise} = \int_{F_s} S_e(f) df = \frac{q^2}{12} = \frac{4^{-F}}{12} \tag{5.17}$$

Therefore the SNR can be found as:

$$SNR = \frac{P_{signal}}{P_{noise}} = 6 \times 4^{I+F-1} \Rightarrow SNR_{dB} = 10\log_{10}(4^W) + 10\log_{10}\left(\frac{6}{4}\right) \tag{5.18}$$

This results in the well-known formula:

$$SNR_{dB} = 6.02W + 1.76 \;\; dB \tag{5.19}$$

In this expression, we call W the effective number of bits (ENOB) actually representing the signal. The ENOB usually decreases through the processing chain, and it is the role of the hardware designer to limit the loss of the ENOB measure. This is the challenge tackled in this chapter.

Spectral Signal to Noise Ratio (SSNR): Similarly to the SNR, we define the SSNR as the ratio of the signal PSD $S_s(f)$ to the noise PSD $S_n(f)$:

$$SSNR = \frac{S_s(f)}{S_n(f)} \tag{5.20}$$

This measure concerns the best achievable SNR after complete filtering of the quantization noise which is outside the signal band. Considering a band B_s of sinusoids with the same signal power as in the last section, which is oversampled using F_s, we can find the expression of the SSNR where we can identify the processing gain (PG) resulting from filtering the quantization noise:

$$SSNR = 6 \times 4^{I+F-1} \times \frac{F_s}{B_s} \;\Rightarrow\; SSNR_{dB} = 6.02W + 1.76 + \underbrace{10\log_{10}\left(\frac{F_s}{B_s}\right)}_{PG} \tag{5.21}$$

Relative bit degradation (Δb): This measure is responsible for tracking the degradation of the optimal achievable SNR. This figure can be used to understand the theoretical loss in precision through a given hardware system. Note that Δb is practically positive, since it is not possible to improve the SSNR of a signal.

$$\Delta SSNR = \frac{SSNR_{out}}{SSNR_{in}} = 4^{-\Delta b} \quad \Rightarrow \quad \Delta SSNR_{dB} = -6.02\Delta b \text{ dB} \tag{5.22}$$

5.1.2 Literature of Quantization Methods

Prior to implementing a signal processing module in digital hardware, the different numbers of bits used to represent the parameters and signals in the modules must be determined. The process of finding these dimensions is known in the quantization literature as wordlength determination [Sun95] or floating-to-fixed point conversion [Men06]. This process consists of two main steps. The first step concerns the determination of the number of bits used to quantize the integer part, which should be sufficient to represent the signal's dynamic range. The second step then handles the fractional part that represents the precision used in calculation. This section sums-up the different approaches developed in the literature to perform these two steps.

5.1.2.1 Integer Part Quantization

The main objective of the first step is to protect from overflow, which is preventing the situation of having the signal value going beyond the range that can be represented by the number of bits used. Therefore this step requires information about the minimum and maximum value every signal may take. Determining this range of values is the core of this step, and it can be done with different approaches. The literature concerning this step can be separated over two categories: the statistical and the analytical approaches.

Statistical Approach: The first approach relies mainly on simulating the circuit's operation. The basic method consists of running multiple simulations of the circuit, while using inputs that are expected to be processed by the module after it is implemented [Aam01][Roy04]. Then the values of each signal in the circuit are tracked, then the minimum and maximum measured values are identified. The quantization then chooses the number of bits that are sufficient to represent the extreme values. The problem of this method is that it heavily depends on the inputs used for simulation, which will be different in real life implementation. Kim et al. proposed to improve this method by considering the inputs as random variables with a defined distribution function [Kim98a][Kim98b]. Then by using both the simulation results combined with the distribution function, a more credible estimation of the signal value range can be obtained.

Analytical Approach: The second approach uses the quantization rules that were presented in Section 5.1.1 to find the value ranges of the different signals in the circuit. Given an input with a defined dynamic range and quantization, this method consists of propagating the quantization through the circuit following the aforementioned rules. These rules are usually conservative since they consider the worse possible case [Cac02][Tou99]. We find propositions that try to improve on the pure analytical approach by combining it with statistical simulations [Cma99]. There exists also automated tools (e.g. FRIDGE tool [Wil97][Ked98]) that combine both approaches, by having certain signals quantized using the statistical approach, and then the rest of the signals are quantized based on quantization rule.

In summary, each approach offers a certain compromise. The statistical approach requires long simulation times to cover sufficient possible input configurations, and it is less reliable than the analytical approach. However the advantage of the statistical approach is to give a more efficient quantization scheme, since the analytical approach is conservative by considering the worst case, and many bit elements might be unused in the implemented circuit, which are considered as wasted resources.

5.1.2.2 Fractional Part Quantization

The second step of the quantization process aims at determining the number of bits for the fractional part to guarantee a certain precision performance constraint. As in the first step, both statistical and analytical approaches are proposed for this step. In the first case, the precision performance is measured from simulation results, while the second case consists of calculating the precision measure using defined quantization models. In both cases, different performance measures and quantization models may be used [Mor83][Car84][Can06][Zha09].

Statistical Approach: This approach consists of simulating the circuit, and then a performance measure is calculated using the resulting signals. Then depending on a given precision constraint and a threshold for the considered performance measure, the number of bits is either increased or decreased. The objective is always kept as respecting the precision constraint with the minimum hardware resources. One of the oldest wordlength optimization method was proposed in [Cat88] that uses simulated-annealing-based optimization to quantize both parameters and signals. Later work by Kum and Sung in [Sun95] and [Kum01] proposed to develop both performance and cost functions, and then an optimization algorithm is proposed that iteratively maximizes the performance function while minimizing the cost function. The work of [Cho96] build on this last method by attempting to group operations using similar quantization schemes in order to maximize the reuse of hardware. Further work by Kim et al. in [Kim94][Kim95][Kim98b] developed statistical methods using the C++ programming language, with a special focus on reducing the long simulation times required by the statistical methods. The FRIDGE tool combines both statistical and analytical approaches, but it relies on bit-true simulation to qualify the quantization performance making it closer to a statistical approach [Ked98].

Analytical Approach: In this case the first step is to develop an analytical expression that relates the performance measure to the quantization scheme used. The developed expression can vary depending on the quantization model and the performance measure selected for optimization. This technique dates back to 1970, where the analytical expression of quantization noise was developed for two IIR filters implementations in [Jac70], and it was shown that certain filter structures can be more efficient for hardware implementation. This work was applied to a specific filter, however later work started focusing on automating the wordlength determination process. The work presented in [Tou99] and [Mar01] proposes an algorithm that considers the circuit as a graph, that is then used to develop the analytical expression of quantization noise to be used for optimization. The limitation of this method was in its inability to process graph cycles, which represent recursive structures in terms of implementation. This was handled in the work of Menard who proposed an automated quantization algorithm capable of applying the analytical approach to recursive structure [Mén02][Men06][Men08]. However this last method was also limited since it couldn't handle non-linear operators efficiently. Later work presented in [Par10] proposed to combine both statistical and analytical approaches, with the quantization noise expression developed for linear and smooth operators, while the operators that are difficult to be presented analytically are simulated.

As in the case of integer part quantization, each approach has its own compromise. The statistical methods can be applied to any circuit, however they require implementing the simulation that usually takes very long time to execute. Some propositions addressed the problem of long optimization time, such as [Kim98b] that proposed to use the mantissa of the float variable for fixed-point simulation. Other algorithms also try to use the native data types in the machine used for simulation to minimize the simulation time [De 98][Ked98]. The analytical approach on the other hand eliminates the troubles of simulating the circuit, however the difficulty lies in building an expression that accurately represents the quantization error level introduced. Recursive structures and nonlinear operators also add complexity to applying the analytical method.

5.2 Optimal Analytical Quantization

The automation of the quantization process dominates the literature presented above. The objective of the different methods aims at building a quantization tool that can be used with any signal processing system. These studies proposed many complex software applications, with some statistical methods being adopted by commercialized hardware computer aided design (CAD) tools. In this thesis, the task is to optimize specific signal processing functions. Since the architecture design is part of this work, then the transfer function of the signal processing system is well known. In addition, for linear sample rate conversion and filtering functions, the system can be easily described with analytical expressions. Hence quantization with the analytical approach is privileged in this work, which will result in accurate and guaranteed precision performance levels, and also in a great reduction in design time by avoiding time-consuming statistical methods. However, the analytical quantization approaches presented in the literature focused on building an automated solution, with a very specific work flow to each approach. Moreover, the work of [Mén02] targeted specifically the implementations on digital signal processors with defined fixed point data types. In this work we are concerned with FPGA and ASIC implementation with complete freedom relatively to the data types definition. Deploying and adapting the software proposed in [Tou99][Mén02], and applying it to quantize our developed modules will require important effort and time.

In this thesis, we found it is important to revisit the quantization methods in order to develop an efficient solution. Our objective is to develop a quantization work flow that is simplistic and easy to use and apply, while offering the possibility of having a guaranteed precision performance.

5.2.1 Integer Part Quantization

5.2.2 Fractional Part Quantization

5.2.2.1 Quantization Criterion

5.2.2.2 Quantization of Multirate Systems

5.2.2.3 Internal Quantization

5.2.2.4 Optimal Quantization Solution

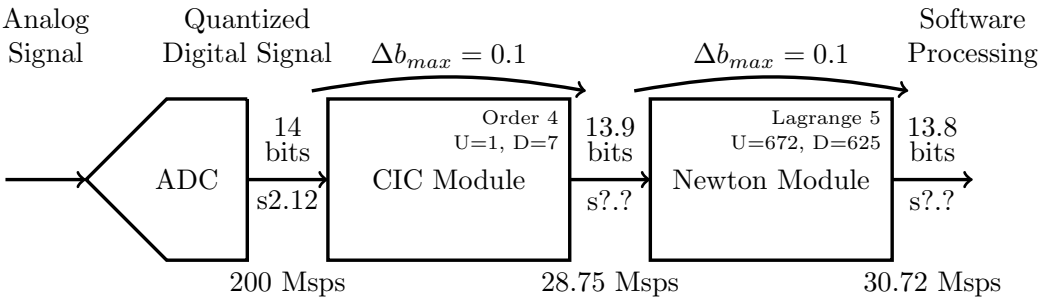


Figure 5.3: A simplistic NB-IoT DFE example

5.3 Application Example

This section presents a practical example of how the developed methods in this chapter can be applied. We consider for this example an NB-IoT receiver. NB-IoT is a recent cellular wireless standard conceived for IoT applications, proposed in [3GP15] and standardized in [3GP18]. One of the commonly used development and testing deployment targets are the universal software radio peripherals (USRPs), and in this example the X310 model is considered [Ett]. The analog to digital converters (ADC) of this USRP operates at 200 Mega samples per second (Msps), with each sample being quantized on 14 bits. We suppose that signed fixed-point quantization is used with two integer and twelve fractional bits. On the other hand, for the case of an NB-IoT signal in a 20 MHz LTE band, the useful baseband signal is required to be sampled at 30.72 Msps for further processing that is performed in software. The sampling rate is defined by the developed software with a given specific oversampling factor. Therefore an SRC operation is required in the DFE with a factor R that is written as:

$$R = \frac{U}{D} = \frac{30.72}{200} = \frac{1}{7} \times \frac{672}{625} = \frac{U_1}{D_1} \times \frac{U_2}{D_2} \tag{5.23}$$

We identify two SRC operations from this expression: a coarse decimation SRC of a factor 7, and a fine-tuned interpolation SRC of a factor 672/625. This is implemented as shown in Figure 5.3 using two SRC modules (this is a simplistic example that omits the filtering and compensation functions). The CIC module performs the decimation of a factor 7, using an implementation of order 4. This order is selected based on the required filtering constraints. The output at a sampling rate of 28.75 Msps is then up-sampled by the Newton module that implements the impulse response of Lagrange interpolation of order 5. The final output at a sampling rate of 30.72 Msps is then ready to be processed by the software section of the receiver.

The quantization method developed in this chapter is then applied to this example system as developed below. We suppose that we can tolerate an extremely stringent effective bits degradation of 0.1 bit through each of the two modules. Therefore the effective bits precision of the output used for software processing is equivalent to 13.8 effective bits.

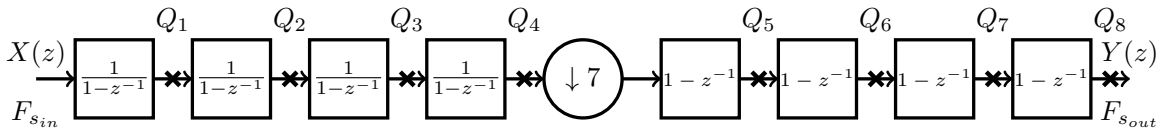


Figure 5.4: CIC Decimation Filter of order 4

5.3.1 CIC Module Quantization

Integer Part Quantization:

Fractional Part Quantization:

[...] results in the optimal F_i combination shown in Table 5.1. This optimization is run for $\Delta SSNR \geq 4^{-0.1}$, a useful signal bandwidth of $B = 8.29$ MHz that includes all sub-carriers of the LTE signal, and with two custom conditions for step 5: take the smallest output quantization parameter and take the maximum $\Delta SSNR$ value.

The number of bits removed from the fractional part for each quantization Q_i is expressed by $F_{in} - F_i$. At the output, 6 bits are eliminated from the fractional part, however 12 bits were added to the integer part. Therefore, for a maximum tolerated effective bits loss of $\Delta b_{max} = 0.1$ bit through the CIC module, the output of this filter is quantized on 20 bits (s2.18 after scaling), while its input is on 14 bits (s2.12).

Table 5.1: Fractional Part Quantization Parameters for the CIC Filter

i	1	2	3	4	5	6	7	8
F_i	12	11	9	8	8	7	7	6
$F_{in} - F_i$	0	1	3	4	4	5	5	6

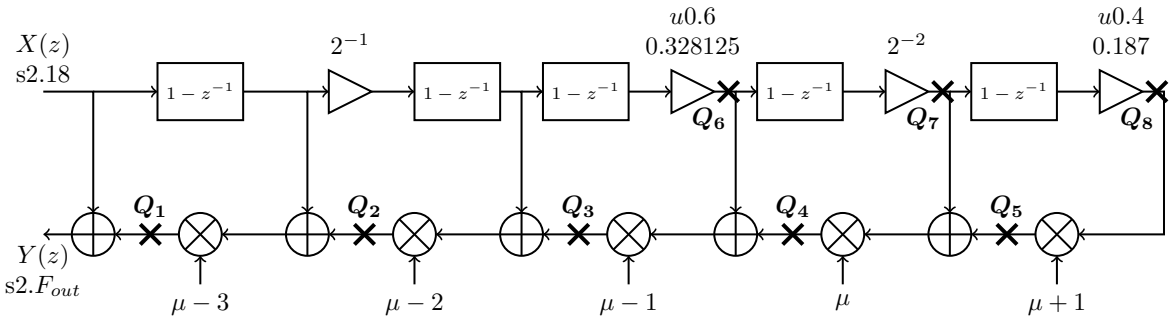


Figure 5.5: Fractional Part Quantization of the Newton Module

5.3.2 Newton Module Quantization

Integer Part Quantization:

Fractional Part Quantization:

[...] We suppose as a practical example in this case that the data interface between the Newton module and software processing is limited to 22 bits per sample. Then the maximum accepted quantization of the output fractional part is on 20 bits. This condition can be introduced at steps 3 to 5 in the algorithm to limit F_1 corresponding to Q_1 to 20 bits maximum, and therefore limiting F_{out} to 20 bits quantization as well. The optimal F_i combination found by the algorithm is shown in Table 5.2. The number of added bits at each quantization location is expressed by $F_i - F_{in}$. At the output, 2 bits were added to the fractional part. Therefore, for a maximum tolerated effective bits loss of $\Delta b_{max} = 0.1$ bit through the Newton module, the output is quantized on 22 bits (s2.22), while the input is on 20 bits (s2.18).

Table 5.2: Fractional Part Quantization Parameters for the Newton Module

i	1	2	3	4	5	6	7	8
F_i	20	23	23	23	21	24	22	21
$F_i - F_{in}$	2	5	5	5	3	6	4	3

If we consider brute quantization approaches, when the signals in this Newton structure are all quantized using the same fractional part as that of the input (18 bits), the effective bits loss is equal to $\Delta b_{max} = 3.1$ bits. While in the brute case when zero added quantization error is tolerated ($\Delta b_{max} = 0$ bits), the fractional part of the signals will get extremely large, and at the output it will reach 61 bits. It may look surprising that with only 2 added bits at the output, it is possible to achieve such a low effective bits degradation of $\Delta b_{max} = 0.1$ bit, however this demonstrates the efficiency of the proposed method that optimizes each quantization parameter independently, where more bits are introduced inside the structure than at its output.

5.4 Conclusion

This chapter started by explaining the basics of quantization, and then presented the different quantization methods available in the literature. These methods were found to be either too complicated, or to require long simulation time to find an appropriate quantization scheme. The work developed in this thesis proposes a simple yet efficient analytical quantization method. Through this method the quantization scheme can be found directly without requiring long simulation time, while offering a guaranteed precision performance level. The only complexity is in developing the analytical expressions of the quantization error, which was explained through this chapter. A general definition was provided on an arbitrary filter structure, and then two concrete examples were provided using two SRC filter structures. These examples demonstrated how it is possible to limit the added quantization error to a minimum by adding a few extra bits for each signal in the filter. With the quantization method defined, it is now possible to implement the newly developed structures in hardware to study their practical complexity and power consumption. The next and final chapter of this manuscript develops and compares the hardware implementation of different SRC solutions using multiple implementation strategies.

Chapter 6

Hardware Implementation

Contents

6.1 Digital Front-End Implementation 114

6.1.1 Deployment Environment 114

6.1.2 Hardware Implementation 116

6.2 SRC Implementation Architecture 118

6.2.1 Interconnection Protocol 118

6.2.2 SRC System Architecture 120

6.2.3 SRC Controller Architecture 122

6.2.4 SRC Filter Architecture 124

6.3 Implementation Results 128

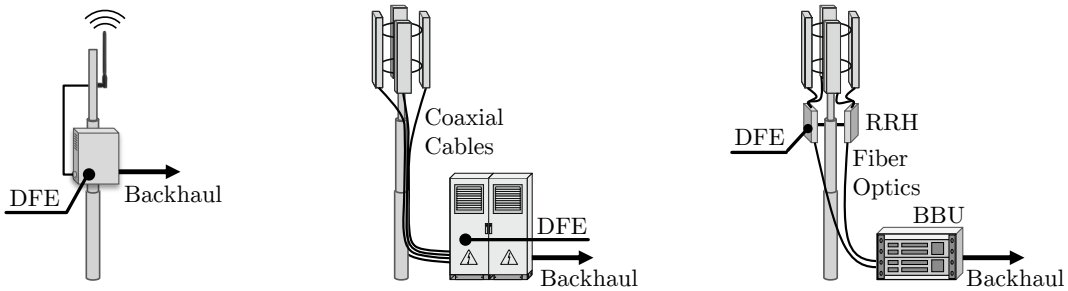
6.3.1 Implementation Context 128

6.3.2 ASIC Implementation 130

6.3.3 FPGA Implementation 133

6.4 Conclusion 137

The objective of this thesis is to design and optimize a DFE that is implemented in hardware, with the main focus on the SRC function. This chapter addresses the hardware implementation aspects of the DFE on both ASIC and FPGA targets. The DFE may be implemented in different configurations depending on the application and the deployment context. Each deployment environment may have different hardware requirements, where one environment may require reduced complexity and lower consumption, other environments may favor higher operating speeds. Depending on the deployment requirements, certain implementation strategies may be more suited for the given hardware constraints. Up to this chapter, the SRC filters developed were only presented from a structural point of view. This chapter investigates the application of the different implementation strategies to these developed SRC filters, and to some classical SRC solutions. This chapter is organized as follow. Section 6.1 presents the different deployment environments, the corresponding implementation constraints, and the implementation strategies used to respect the different constraints. Section 6.2 then develops the hardware implementation of multiple SRC filters using different strategies, and provides the corresponding register transfer level (RTL) architectures to be implemented. Section 6.3 obtains the implementation results on both ASIC and FPGA targets. These results are then studied and analyzed to demonstrate the usefulness of each implementation strategy. Finally, Section 6.4 concludes this chapter.



(a) Dedicated Gateway (b) Base Transceiver Station (c) Remote Radio Head

Figure 6.1: Three deployment strategies that place the DFE in different parts of the system

6.1 Digital Front-End Implementation

Beyond designing and optimizing a generic DFE, this thesis also aims to implement the optimized modules in hardware. Implementation of these modules provides practical measures to quantify the reduction of complexity provided by the newly developed modules. These measures are presented later in Section 6.3. As it was discussed in Chapter 1, any signal processing module can be implemented in either hardware or software. In this thesis, both implementations were developed. The software implementation was mainly used for verification purposes, while the hardware implementation represents the final product that is deployed in the multi-standard gateway. This section develops the hardware implementation context of the DFE. Section 6.1.1 discusses the environment of the DFE hardware implementation. Section 6.1.2 then develops the constraints used to qualify the implementation, the different strategies that can be used to respect these constraints, and finally the hardware verification process.

6.1.1 Deployment Environment

This section discusses the deployment environment of the IoT DFE. First, the deployment strategies are discussed, and then the hardware implementation targets are presented.

6.1.1.1 Deployment Strategies

The hardware implementation of the DFE may be deployed in various configurations. Depending on the supported IoT standards, the expected number of users, the type of application, and other factors, certain deployment strategies will be more adapted than others. In this section we present the three most common deployment configurations of gateways: Dedicated gateway, Base transceiver station (BTS), and the Remote Radio Unit (RRU).

Dedicated Gateway : This solution is most common with IoT standards and technologies using non-licensed frequency spectrum for transmission (e.g. LoRa and Sigfox standards). In this case the dedicated gateway will usually include many processing functions: the RF front-end, the digital/analog conversion, the digital front-end, the baseband processing, and wired or wireless back-haul connectivity with the application server. This architecture is presented in Figure 6.1-a.

Base Transceiver Station (BTS) : or the Node B are the classical solutions to connect mobile devices to the cellular network. This architecture is of interest when integrating the IoT DFE in a GSM or WCDMA network [FaL11, Chapter 5], or in the case of deployment of the EC-GSM standard, that is a modification of the GSM standard for IoT applications [Gra16]. In this configuration, the antenna output signal is connected to the BTS using coaxial

cables, as shown in Figure 6.1-b. The BTS then runs the signal through the RF front-end, the analog/digital conversion, and the digital signal processing. The DFE can be introduced in the BTS to enable processing of different cellular or IoT standards [Pri09], with a separate back-haul connection for the IoT data.

Remote Radio Unit (RRU) : In many cases the BTS is located inside a cabinet on the ground, and is connected to the antenna through coaxial cables, with the antennas being installed on a mast. In this case, the signal carried by the coaxial cables will be subjected to attenuation and to interfering signals. In modern cellular network implementations, where larger bandwidth capacities and a greater number of antennas are required, a more efficient base station is needed. This was made possible by using remote radio units (RRU) or remote radio heads (RRH), that are installed on the mast next to the antennas [Lit12]. This is illustrated in Figure 6.1-c. A RRU usually includes the RF front-end, the analog/digital conversion, and the DFE [Alt09]. The RRU baseband output is then connected through fiber optics cables to a local or remote baseband unit (BBU), that handles further processing and the data transportation on the back-haul connection. This is the deployment strategy that will be most adapted for the 5G network that will include both cellular and IoT standards. Using this strategy provides a modular deployment, with higher bandwidth and lower consumption and latency relatively to the classical BTS solution.

6.1.1.2 Implementation Targets

In each deployment strategy, various implementation targets are possible. The software approach is the most reconfigurable, that consists of implementing the signal processing function on a general purpose processor (GPP) or a digital signal processor (DSP). However this approach is not the most efficient, and can be often impractical when implementing very high speed signal processing DFEs, that require direct hardware implementation. The latter is also required for large scale production and deployment, where low consumption and low manufacturing costs are primordial. Field programmable gate arrays (FPGAs) and application specific integrated circuits (ASIC) are the two widely used targets for direct hardware implementation.

FPGA : An FPGA is an integrated circuits containing arrays of programmable logic elements or cells, that is capable of implementing signal processing applications of various complexity. Since FPGAs can be reprogrammed, they offer a great prototyping and testing solution for hardware implementation. They may be also used in the final product if they manage to satisfy the required implementation constraints. In this case they offer a great advantage over ASIC by giving the possibility of reprogramming and updating the implementation when needed. For hardware implementation, we often require very high speed and efficient hardware with the lowest costs for large deployment, in which case FPGAs are not practical and ASICs are the better solution.

ASIC : ASIC consists of an integrated circuit that is destined for a specific application as its name applies, where the final hardware implementation is fixed. ASIC presents many advantages over FPGA, most notably when it comes to efficiency. ASIC allows higher operation speeds while consuming much less power. And in the case of mass production, the costs of ASICs are a fraction of the same production with FPGAs. Another advantage of ASIC is the mixed signal technology, which allows implementing both analog and digital signal processing on the same chip, thereby increasing the implementation efficiency even further. The hardware implementations developed and presented in this chapter consider both ASIC and FPGA implementation targets.

6.1.2 Hardware Implementation

The hardware implementation of the same signal processing function can be done in many different ways. On one hand, it is possible to maximize the operation speed by using more hardware resources, however this leads to higher complexity and more power consumption. On the other hand, the complexity can be minimized by reusing hardware resources, however this reduces the operation speed. Therefore, the choice of implementation strategy depends on a compromise between different constraints. In this section, the hardware implementation constraints are presented, and then the different strategies of implementation and the advantages of each are discussed.

6.1.2.1 Implementation Constraints

Hardware implementations are usually evaluated using four main measures: complexity, precision, speed, and consumption. Depending on the application and deployment environment, specific constraints on these measures may be required (e.g. operation speed of 300 MHz), or general constraints may be applied (e.g. use less than 20% of the total hardware resources). To respect the required constraints, the designer should consider different hardware aspects, with each affecting the implementation in a specific way.

Complexity : The first measure reflects how much hardware resources are required for the implementation. This depends mainly on the signal processing module itself, where a more complex function will evidently requires more resources. However the hardware complexity depends also on two other aspects: the implementation strategy (explained below), and the hardware quantization (studied in Chapter 5). Measuring the complexity is done differently on ASIC and FPGA. In the first case, the transistor count or the number of logic gates is used to quantify the complexity. While in the second case, the logic cells count and the number of DSP blocks is used. Comparing the complexity between ASIC implementations is not the same as comparing the same implementations on FPGA. This is studied in more details later in Section 6.3.

Precision: The second measure concerns the calculation precision of the implementation. This is defined mainly by the hardware quantization, which is the number of bit elements used to represent the signal. This measure is quantified with the signal to quantization noise ratio (SQNR). By using more bit elements, the noise level will be lower and therefore the SQNR will be higher. Many quantization schemes exist, however in this work we focus on the fixed point quantization that is very efficient for low complexity implementations. Chapter 5 develops the fixed point quantization methods used in this work.

Speed : This measures the maximum operating frequency of the implemented circuit, that is defined by the critical path of signal transfer. Operation speed is mainly influenced by the technology used. ASIC implementations are usually faster that FPGA ones for the same module with the same technology node. And in general, smaller and more recent technology nodes are faster than larger and older ones (e.g. 14nm technology is faster than 45nm). The operation speed is also influenced by the hardware architecture, and the implementation strategy as explained below.

Consumption : Measuring the exact consumption is tricky, and often an approximation is given. The difficulty comes from the fact that the consumption does not only depends on the circuit voltage and current. The consumption of a hardware implementation also depends on the digital clock used, the signal toggle rate, circuit temperature, capacitance, etc. Hardware design software usually offer tools to find an approximate of the implementation consumption.

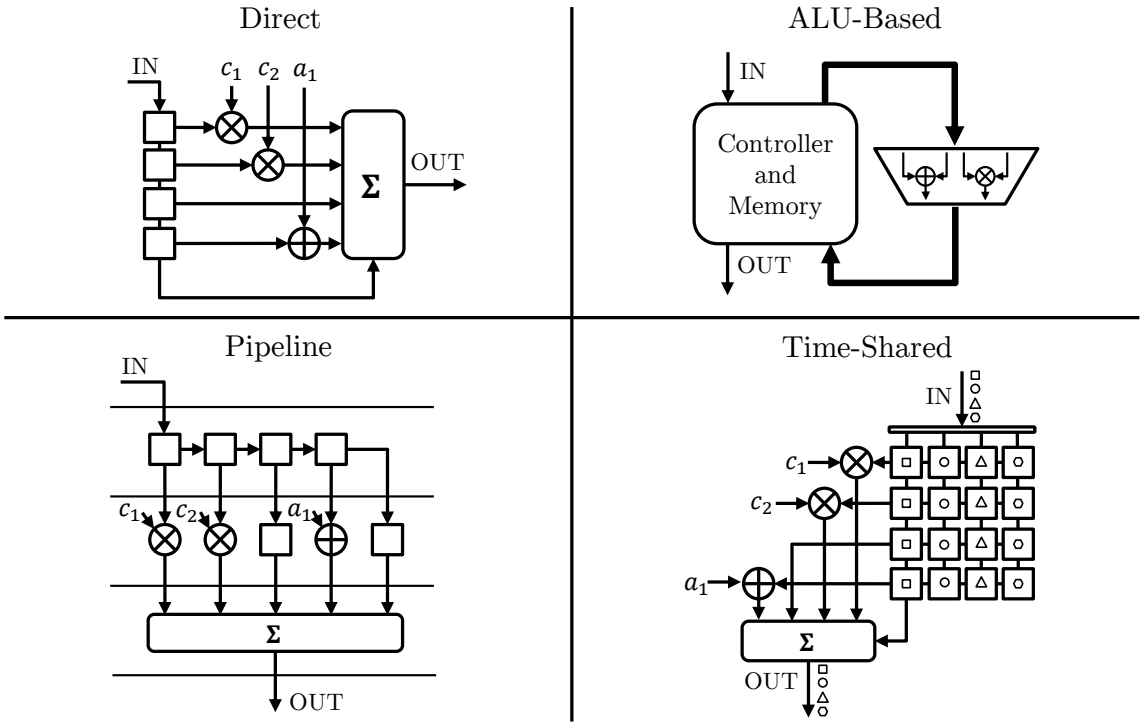


Figure 6.2: The different implementation strategies implementing the same module

6.1.2.2 Implementation Strategy

When implementing a hardware module, it is important to use the implementation strategy that best correspond with the specified constraints. The possible implementation strategies of a hardware module are discussed below, and illustrated in Figure 6.2.

Direct : In this case the hardware circuit is a direct implementation of the module structure, and a complete calculation is performed every clock period. The problem with this strategy is that the signal propagation time between input and output can be very large. This will greatly limit the maximum operation frequency of the implementation.

Pipeline : The pipeline strategy aims at maximizing the operating frequency by reducing to a minimum the critical path length. The module structure is split into sequential operations, and each operation is executed separately between two hold and sample buffers (presented as horizontal lines on Figure 6.2). Any signals that need to be used by later operations need to be propagated through the pipeline using registers. Therefore this strategy offer faster operation speed, at the cost of many added registers and increased complexity.

ALU-based : It is possible to reuse only one adder and multiplier to implement all the addition and multiplication operations in a module. We call this strategy the ALU-based, since we have an arithmetic logic unit (ALU) that contains basic operations that are used to implement the total calculation. A controller is needed to manage the sequence of calculation, and to control writing/reading the inputs/outputs of the ALU respectively. This strategy offers important reductions in complexity, however calculating one output requires many clock cycles, which is only possible when the data rate is much lower than the digital clock rate.

Time-Shared : This is another implementation approach that consists of reusing hardware. However it is not an independent strategy by itself, as it needs to be applied to one of the strategies presented above. Time-sharing consists of processing multiple data streams using

one hardware implementation. The added complexity for implementing this feature consists mainly of memory registers and switching control circuitry. There is no need for a dedicated time-sharing controller, where usually start of word/end of word (sow/eow) signals are used to indicate the first and last input streams. This strategy does not offer the highest operation speeds, however it is interesting for multi-channel processing implementations.

6.1.2.3 Hardware Verification

Hardware verification is a critical step of every design to ensure the correct functioning. This is a major step that increasingly requires more time as the size and complexity of the implementation grows. In the work of this thesis, all the developed hardware implementation were verified following the co-verification process developed by the technological research institute b-com. This process consists of developing first the bit-true software implementation of the module, and its correct functioning is then verified by simulations. Afterwards, the developed hardware description is simulated in parallel to the software implementation. The outputs of the two implementations are compared and they are supposed to match exactly. Once the functional verification of the hardware is done, the implementation is programmed on the FPGA, and real-time execution outputs are used to verify again the correct operation. For ASIC implementations, similar verification methods are used. However verification after implementation on chip is usually done by the ASIC manufacturer using specific tools for the used technology.

6.2 SRC Implementation Architecture

This section develops the hardware architecture of the implemented DFE modules. Specific examples for the SRC systems are developed. Section 6.2.1 starts with presenting the interconnection scheme used for implementation in this work. Section 6.2.2 then develops the SRC system architecture using the interconnection scheme presented. Two modules may be used to build the SRC system: the SRC controller, and the FIR filter. The architectures of these modules are developed in Section 6.2.3 and Section 6.2.4 respectively.

6.2.1 Interconnection Protocol

Since the DFE consists of sequential processing through a cascade of systems, there is no need for a bus connecting the different modules. The handshake protocol offers a very efficient solution for such implementation. In addition, for the time-shared implementations were multiple data streams are processed by a single module, control signals are required to track the different samples.

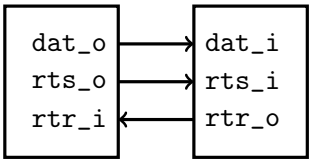
Handshake Protocol: The handshake protocol defines the interface between the different hardware modules, and controls the progress of the processing chain. The main concept of this protocol is to establish a communication link between the hardware modules to signal when a data sample can be sent from one module to another. This protocol is implemented using two types of signals:

1. Ready To Send (rts): The previous module signals to the next module a new output is ready to be sent
2. Ready To Receive (rtr): The next module signals to the previous module it is ready to receive an offered input

Then the protocol functioning is defined as shown in Table 6.1. When both signals are zeros, this means that both modules are not ready to communicate. If only one of both modules

is ready for communication, then this module has to hold and wait until the other module becomes ready. This should be taken into consideration when designing the DFE, since having one module holding for another may lead to the whole processing chain being blocked waiting for the slowest module. It only in the case when both modules are ready to communicate that a data sample is transferred from one module to another, and the processing of new samples may resume.

Table 6.1: Handshake protocol behavior



rtr	rts	Behavior
0	0	Both modules are not ready to communicate
0	1	The previous module holds its output sample `dat_o` waiting for the next module to use it
1	0	The next module waits for the previous module to deliver a new valid output
1	1	The next module consumes as input `dat_i` the output `dat_o` of the previous module

Time-Shared Control: A time-shared implementation consists of system processing multiple data streams that are time interlaced. To track the processed sample and to identify the corresponding data streams, control signals are required alongside the data sample. In this case there are two identification signals:

1. Start of Word (sow): The current data sample corresponds to the first data stream
2. End of Word (eow): The current data sample corresponds to the last data stream

These signals are forwarded through the time-shared modules in the DFE processing chain. They are used by the time-shared modules to operate correctly. They are also used when the time interlaced data streams need to be separated again.

Example Communication: To clarify the interconnection protocol, a communication example between two time-shared modules is presented in Figure 6.3. The time-shared operation is supposed to have N time interlaced data streams. It is seen how the first and last data samples are associated with a `sow` and `eow` signals respectively. The handshake protocol behavior is illustrated with the `rts` and `rtr` signals, demonstrating the handshake procedure explained in Table 6.1.

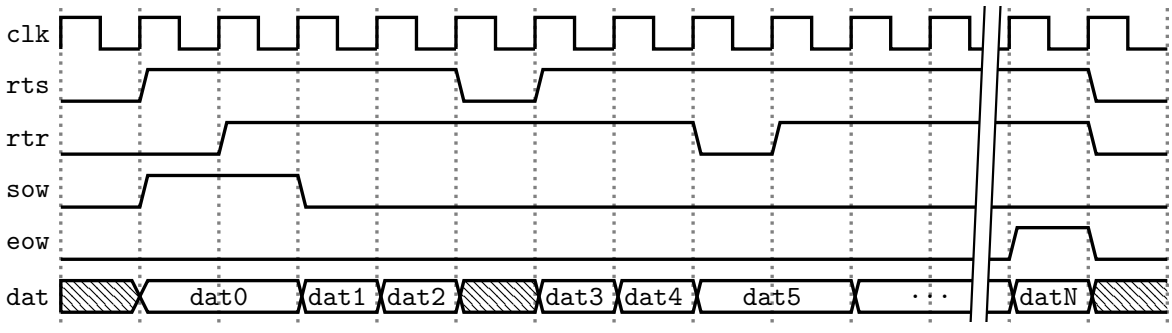


Figure 6.3: Three deployment strategies that place the DFE in different parts of the system

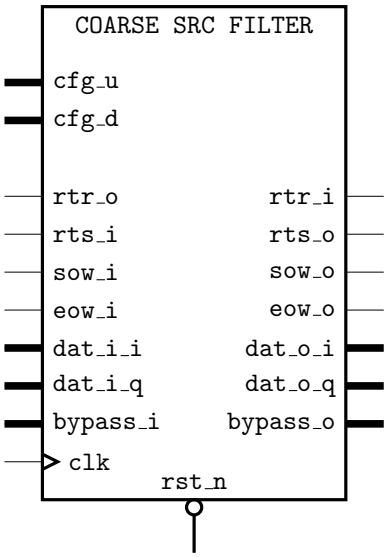


Figure 6.4: Coarse SRC hardware module

6.2.2 SRC System Architecture

Two cases are possible when implementing an SRC operation in hardware. On one hand, it is possible to have only one digital clock that drives the SRC system. On the other, it is also possible to have two or more digital clocks that drive different parts of the SRC system of different sampling rates. Having multiple clocks helps building efficient systems by reducing the clock speed when possible, this reduces thereby the energy consumption. In this thesis, the objective is to evaluate the hardware complexity of the developed SRC modules, therefore the case of only one digital clock is considered. In Chapter 2, two types of SRC systems were presented. The first performs up and down sampling directly on the signal, which is used primarily to implement coarse SRC operations. This is the case of the U-F-D and CIC filter structures. The second type of system finds the output samples by applying interpolation on the input ones. This is used primarily to implement fine SRC operations, such in the case of the Farrow and Newton structures. This section develops the SRC system hardware modules for both cases, and shows the configuration of the input and output ports of each module.

Coarse SRC Filter: The implementation of coarse SRC consists of only the FIR filter and the up and/or down-sampling operations. In this case no SRC controller is needed, and only one hardware module implements the SRC operation. Figure 6.4 illustrates the input and output ports configuration of a coarse SRC filter. On this figure, four kinds of signals can be identified: clock, control, configuration and data. The clock signal driving the module is associated with an asynchronous active low reset. The control ports (`rtr,rts,sow,eow`) are used to manage the SRC module connection to the other DFE modules. The configuration ports (`cfg_u,cfg_d`) are used to define the SRC factor $R = U/D$. These signals may be made constant, or may be made variable if the SRC module is required to implement different SRC factors. In the latter case, the configuration parameters are provided externally from the DFE, by a micro-controller for example. The bitwidth of these ports is defined based on the values that U and D might take. It is supposed that a complex modulation is used, therefore two data sample streams are required for the in-phase (I) and quadrature (Q) components. The bitwidth of these ports is defined based on the quantization precision requirements of the system (see Chapter 5). The `bypass` port carries any information associated to the data samples, such as timing or debugging information. The hardware module is responsible for keeping the bypass information and the data samples synchronized. The bitwidth of this

Table 6.2: Coarse SRC hardware module ports

Port	Bitwidth	Description
IN		
cfg_u	DYN_CFG_U	Up-sampling parameter U
cfg_d	DYN_CFG_D	Down-sampling parameter D
rtr_o	1 bit	Ready to receive input
rts_i	1 bit	Previous module ready to send its output
sow_i	1 bit	Start of word (for time-shared implementations only)
eow_i	1 bit	End of word (for time-shared implementations only)
dat_i_i	DYN_DAT_I	In-phase input data
dat_i_q	DYN_DAT_I	Quadrature input data
bypass_i	DYN_BYPASS	Bypassed information associated to the data sample
clk	1 bit	System clock
rst_n	1 bit	Asynchronous active low reset
OUT		
rtr_i	1 bit	Next module ready to receive input
rts_o	1 bit	Ready to send output
sow_o	1 bit	Start of word (for time-shared implementations only)
eow_o	1 bit	End of word (for time-shared implementations only)
dat_o_i	DYN_DAT_O	In-phase output data
dat_o_q	DYN_DAT_O	Quadrature output data
bypass_o	DYN_BYPASS	Bypassed information associated to the data sample

port depends on the amount of information to be transferred. Table 6.2 resumes the ports configuration of the coarse SRC filter module. The internal architecture of the system is detailed in Section 6.2.4.

Fine SRC Filter: Considering the case of polynomial based filters, such as the Farrow and Newton structures, the fine SRC filter consists of two modules: the SRC controller, and the V-FDF filter. The hardware module with both sub-modules is illustrated in Figure 6.5. The SRC controller as explained in Chapter 2 is responsible for managing the operation of the V-FDF filter, by handling the input and output samples indexing, and by calculating the fractional delay. The internal architecture of this module is detailed in Section 6.2.3. The V-FDF filter is then responsible for interpolating the output value. The internal architecture of this module is discussed in Section 6.2.4. Most input and output ports of this module are similar to the ones in the previous case shown in Table 6.2, with only few added ports to this case. The most important of these added ports are the ones connecting the SRC controller to the V-FDF filter. The `uk` port carries the fractional delay value μ_k calculated by the SRC controller. The bitwidth of this port is found as explained in Chapter 2 (see Section 2.3.2). The control signals `rts_ack` and `rts_cmp` in parallel to `uk` are responsible for signaling to the V-FDF filter if a new input is available and/or if a new output is required respectively. The added port `cfg_inv_u` provides the value of $1/U$ required by the SRC controller for calculation. This is provided externally in order to eliminate the hardware complexity required to perform this division. Finally, the port `cfg_csh` is used in the case the V-FDF is reconfigurable, as in the case the reconfigurable Newton structures developed in Chapter 4. The bitwidth of this port depends on the reconfiguration granularity required. The added ports in the case of fine SRC filter are resumed in Table 6.3.

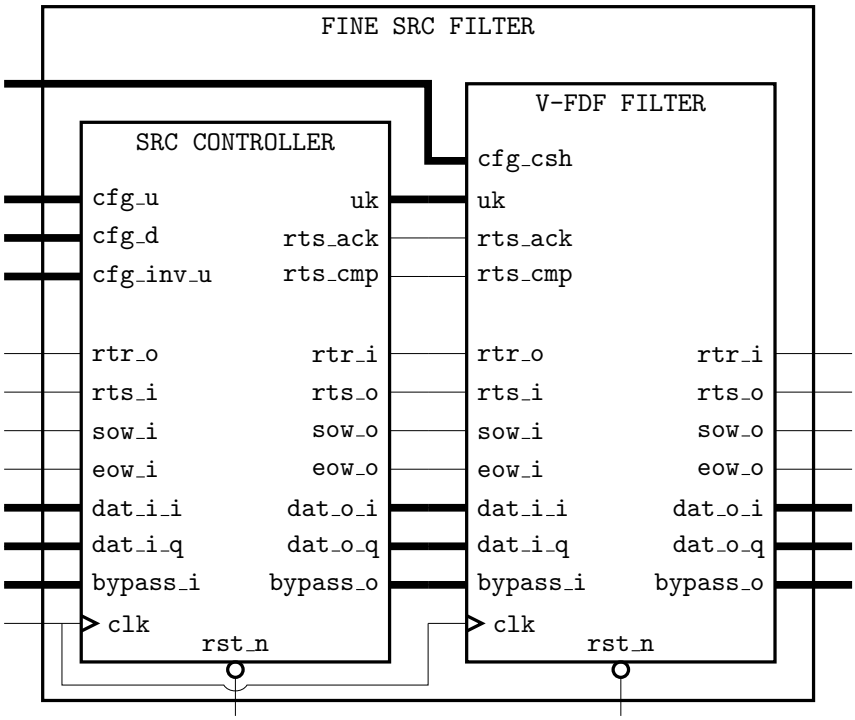


Figure 6.5: Fine SRC hardware module

Table 6.3: Fine SRC hardware module specific ports

Port	Bitwidth	Description
IN		
cfg_csh	DYN_CFG_CSH	Reconfiguration variable (reconfigurable structures only)
cfg_inv_u	DYN_CFG_INV_U	Precalculated value of $1/U$
SRC CONTROLLER		
uk	DYN_UK	Fractional delay value for output k
rts_ack	1 bit	New input available signal
rst_cmp	1 bit	New output need to be computed

6.2.3 SRC Controller Architecture

An essential part of the fine SRC hardware implementation is the SRC controller module. When increasing or decreasing the sampling rate, the number of input and output samples of the SRC filter is different. Therefore many outputs may be required before an input is available, or vice versa. The V-FDF filter is not capable of controlling the flow of input and output samples on its own. Therefore the SRC controller is required. This SRC controller also has the responsibility of updating the value of the fractional delay μ_k associated to each output sample. The algorithm that defines how to control the samples flow and how to find the value of the fractional delay is explained in Chapter 2 (See Section 2.2.2 and Section 2.3.2). The hardware implementation of this algorithm is done using a finite state machine (FSM). The state transition diagram of this FSM is shown in Figure 6.6. When reset, the controller starts at the Idle state. Then based on the aforementioned SRC controller algorithm, the states are updated as follow:

1. Neutral: This is the neutral interpolation state, where a new input is taken (`rts_ack`), the value of `uk` is recalculated, and a new output is found (`rts_cmp`). The SRC controller stays in this state until multiple outputs are required from the same input, or until no

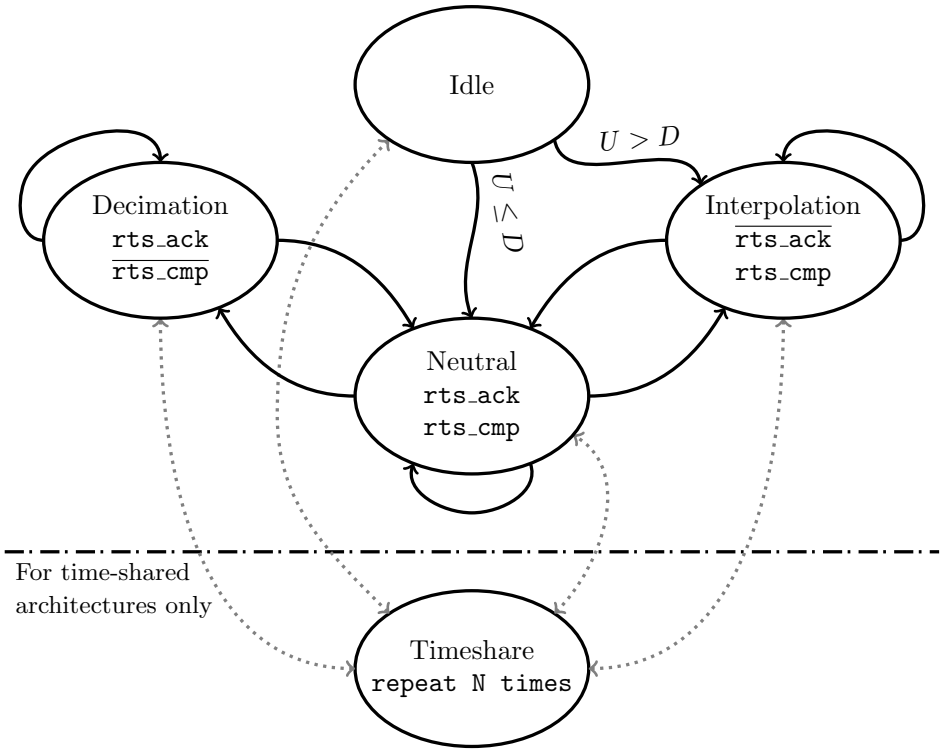


Figure 6.6: SRC controller state transition diagram

output is required from a new input. The controller then goes to the Interpolation or Decimation states respectively. For finely-tuned SRC operations with a factor $R \approx 1$, the SRC controller stays primarily in this state the majority of the time.

2. Interpolation: When the sampling rate is increased, certain inputs might be reused to find at least two outputs. This is because the output sampling period is smaller than the input one. The interpolation state finds the new value of `uk`, and signals to the V-FDF filter that an output is required (`rts_cmp`), while no new input is available ($\overline{\texttt{rts\_ack}}$). The SRC controller stays in the Interpolation state until a new input is available.

3. Decimation: When the sampling rate is reduced, the output sampling period is larger than the input one. This means that not every time a new input is available there will be a required output. In this state, the SRC controller signals to the V-FDF filter than an input is available (`rts_ack`) but that no new output is required ($\overline{\texttt{rts\_cmp}}$). The value of `uk` is ignored in this state.

4. Timeshare: This state is only used by the SRC controller of time-shared implementations. In this case, the SRC controller goes to the Timeshare state every time the `sow` signal is high. In this state, the same control signals (`rts_ack`,`rts_cmp`) and the value of (`uk`) are held until the `eow` signal is high, meaning that all time-shared channels has been processed. Then the SRC controller progresses to the next state defined before going to the Timeshare state.

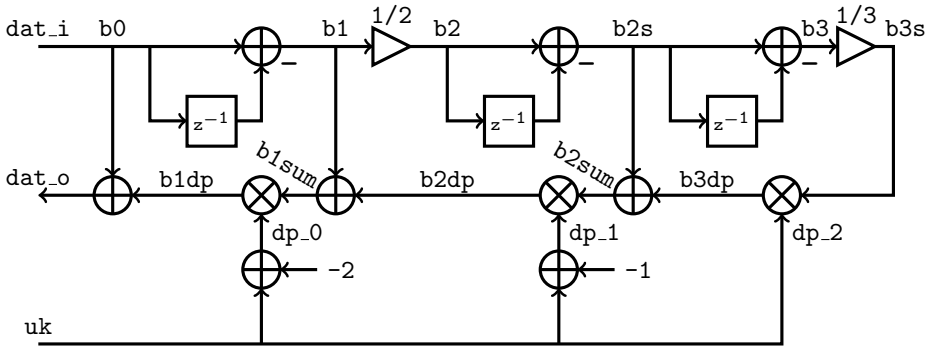


Figure 6.7: Newton structure of order 3

6.2.4 SRC Filter Architecture

Different implementation strategies were presented in Section 6.1.2, and it was shown how each strategy favors the improvement of certain hardware constraints. For the implementation of the coarse SRC module or the V-FDF filter, the different implementation strategies might be interesting depending on the application. This section studies these different implementation strategies through concrete examples on certain SRC filter structures. Implementing the rest of the SRC filter structures using these implementation strategies is based on the same approach presented in the these examples. The pipeline strategy is first used, followed by the ALU-based strategy, which is finally combined with time-shared capacities.

Pipeline architecture: The pipeline architecture aims to maximally reduce the critical path in order to maximize the operation speed. This is done by breaking the filter structure into stages containing only one arithmetic operation each. First each signal on the filter structure is assigned an identifier as shown in Figure 6.7 for the Newton structure of order 3. Then starting from the input port, each arithmetic operation is assigned to a state, while the signals that might be later used are buffered using registers. For the Newton structure of order 3, the pipeline implementation architecture is shown in Figure 6.8. The registers in the first three stages are enabled by the `stage_ack` signal that is driven by the `rts_ack` port. The `rts_cmp` on the other side controls if a new output is calculated or not. It should be also noted that stage 4 consists of only delay elements, and this is due to hardware limitations. For FPGA implementations, the multiplication operations are performed on DSP slices next to the FPGA fabric. Having two consecutive multiplication operations will results limit the circuit's operation frequency, due to the long routed signals required to connect two multipliers resources. Hence a delay stage is introduced between two consecutive multiplication operations. The quantization parameters shown in parentheses describe the number of bits that should be added to the integer and fractional quantization parts of an input quantized on s2.16 bits, in order to have a degradation of SSNR lower than 0.1 dB (refer to Chapter 5 for more details on quantization). The truncated outputs of the multipliers are rounded to the nearest value to avoid adding a DC offset.

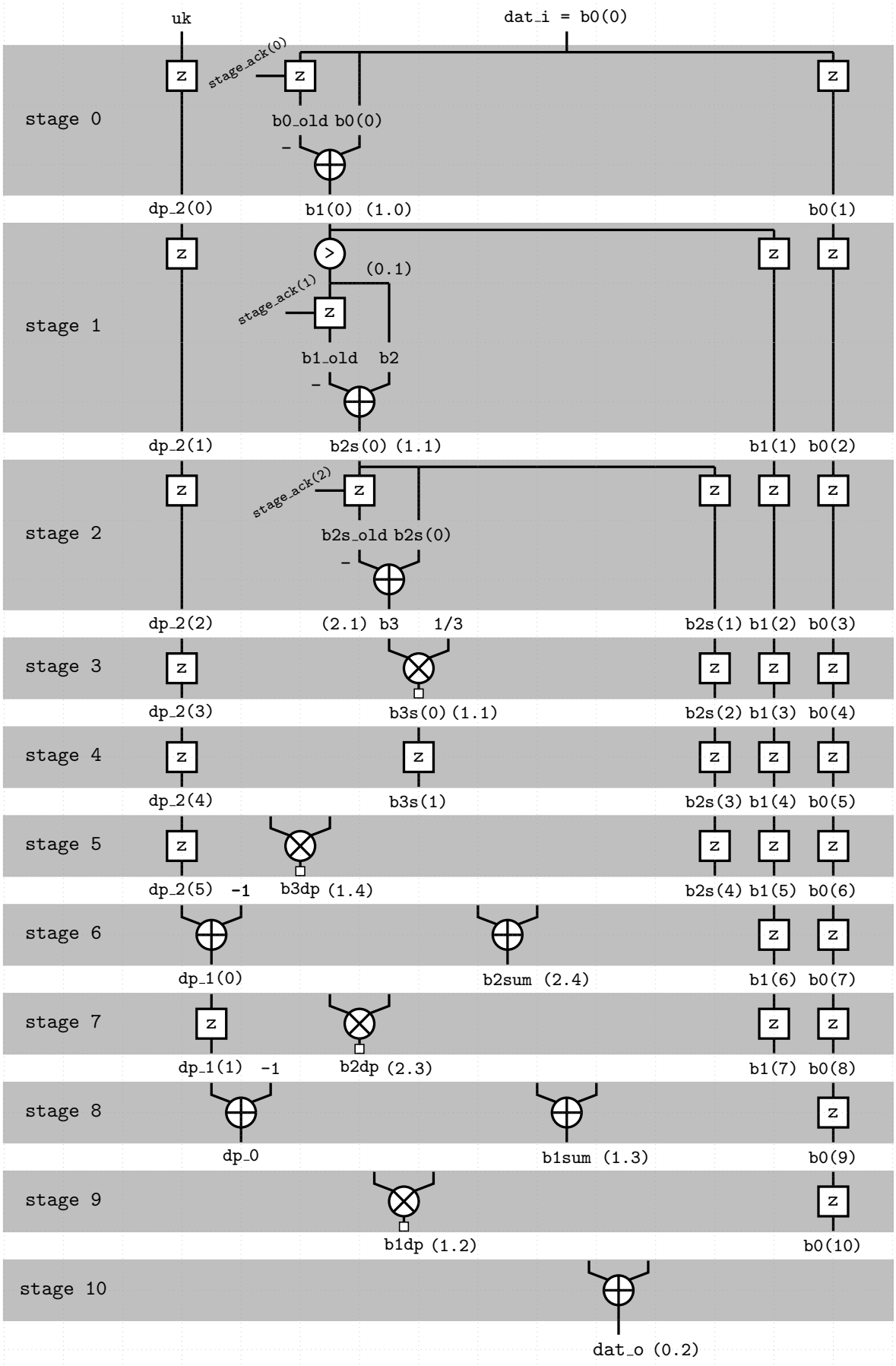


Figure 6.8: Pipeline architecture of the Newton structure of order 3

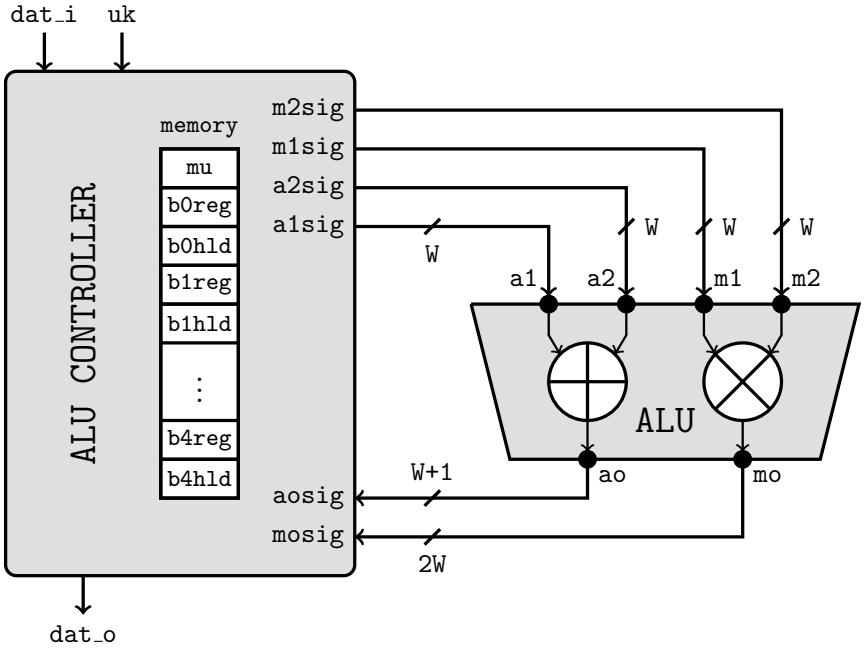


Figure 6.9: ALU-based implementation architecture

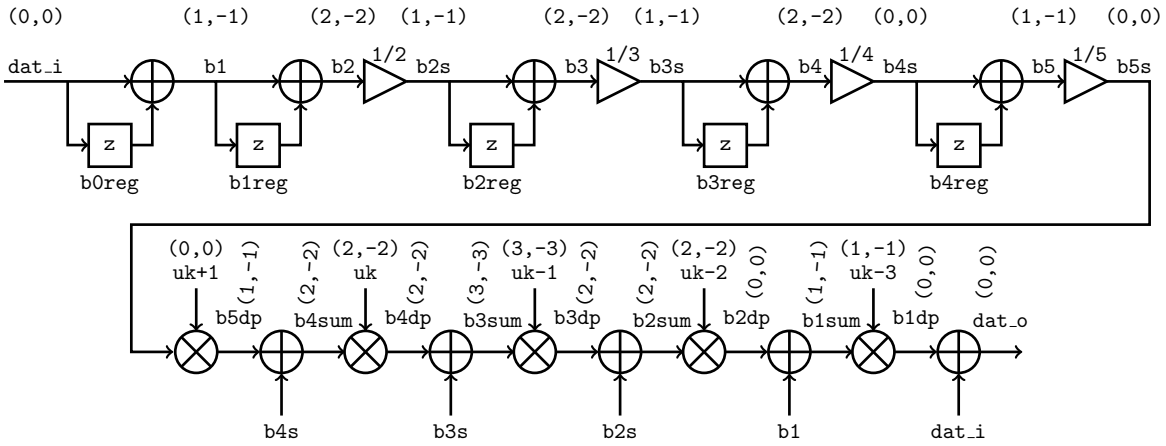


Figure 6.10: Newton structure of order 5 implemented in ALU-based architecture

ALU-based architecture: The objective of this architecture is to implement the whole structure using only one adder and one multiplier. This two arithmetic operators are what form the arithmetic logic unit (ALU) as shown in Figure 6.9. The ALU controller has the role of using the ALU to find the output `dat_o`. The calculation is done over a certain number of cycles, where for each cycle, the controller updates the ALU inputs and reads the ALU outputs. In parallel, any value that will be used lated is stored in a memory array. To develop the ALU-based architecture, the Newton structure implementing Lagrange interpolation of order 5 is considered. The flat structure and the signal identifier are illustrated in Figure 6.10. In this case, 18 cycles and 11 memory registers are needed to find the output. The scheduling of these cycles is developed in Table 6.4, that shows how the inputs and output of the ALU are managed, and how the memory is updated. Since the Newton structure is modular, it is possible to bypass certain stages to implement the Lagrange interpolation of a lower order. For example, the interpolation of order 3 can be implemented by jumping to cycle 11 after the cycle 03.

Supposing that the ALU inputs are quantized one W bits, the adder and multiplier out-

Table 6.4: Scheduling of the ALU-based implementation

Cycle	a1	a2	m1	m2	memory	Notes
C00	dat_i	-b0reg			mu = uk b0hld = input	Start when rts_i = 1 and rtr_o = 1
C01	ao	-b1reg			b1hld = ao b0reg = b0hld	Update bXreg only when rts_ack = 1
C02	ao/2	-b2reg			b2hld = ao/2 b1reg = b1hld	
C03			ao	1/3	b2reg = b2hld	Bypass to cycle 11 for order 3
C04	mo	-b3reg			b3hld = mo	
C05	ao/4	-b4reg			b4hld = ao/4 b3reg = b3hld	
C06	mu	+1	ao	1/5	b4reg = b4hld	
C07			mo	ao		
C08	mo	b4hld				
C09	mu	-1	ao	mu		
C10	mo	b3hld			mu = ao	
C11	mu	-1	ao	mu		
C12	mo	b2hld			mu = ao	
C13	mu	-1	ao	mu		
C14	mo	b1hld			mu = ao	
C15			ao	mu		
C16	mo	b0hld				
C17					dat_o = ao	Wait while rts_o = 1 and rtr_i = 0

puts are then quantized on W+1 and 2W bits respectively. The precision performance of the ALU-based structure is then defined by the input quantization, since all the signals share the same bitwidth. In this case, the fixed point position is defined by the integer part quantization, where any added bits to the integer part are removed from the fractional part. This is illustrated in Figure 6.10 for each signal, with the numbers between parentheses (x,-x) expressing the number of bits added to the integer part and removed from the fractional part. Then for a given precision constraint the largest bitwidth is used for selecting the bitwidth W. In the case of the Newton structure of order 5 quantization developed in Chapter 5, it is the signal `b3sum` that represents the worst case, with 3 and 6 bits to be added for the integer and fractional parts respectively. Then W is 9 bits larger than the input `dat_i` quantization.

Time-Shared Architecture: In gateway implementations, multiple channels are processed simultaneously, and the same SRC operation needs to be applied to all these channels. In this case, time-shared architectures are an efficient implementation when the digital clock speed of the system allows it. It is possible to time-share both the pipeline and ALU-based architectures. The pipeline architecture requires a significant amount of memory resources to keep the context for the different channels, since the filter structure is broken into many stages. Time-sharing the ALU-based architecture is more interesting for low-cost implementations. However this is only practical when the sampling rate is much lower than the digital clock rate. This is due to the ALU-based requiring many cycles to calculate the output, and then

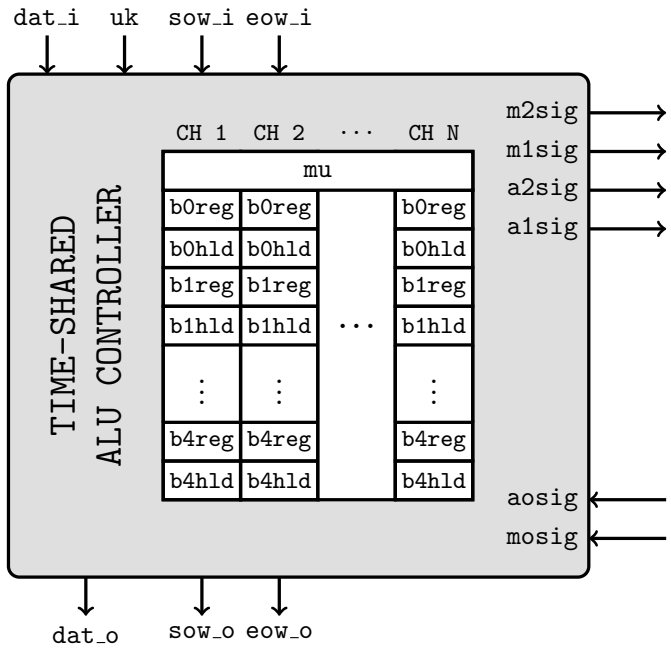


Figure 6.11: Time-shared ALU-based implementation architecture

time-sharing this architecture will apply each ALU controller cycle to the different channels. Nevertheless, this is still practical to implement the SRC operations at the end of many receiver DFE processing chains, after the channels have been separated and their sampling rate reduced. An example of a time-shared CIC filter configuration is presented in [Alt07b].

To integrate the time-shared processing into the ALU-based structure shown in Figure 6.9, only the ALU controller needs to be modified. First, the time-sharing control signals (`sow,eow`) are introduced, and then they are used by the controller to identify the channel of each data sample. For each channel, a dedicated memory array is used to save the context of the registers. Only the memory elements storing the fractional delay value is shared between the channels, since the same SRC operation is applied to all the channels, and therefore the same fractional delay is used. The time-shared adaptation of the ALU controller is illustrated in Figure 6.11.

6.3 Implementation Results

Having developed the implementation architectures in Section 6.2, this section investigates the practical complexity by implementing the architecture in hardware. This section considers both ASIC and FPGA targets. Section 6.3.1 presents the implemented architectures and the used hardware technologies. Then the implementation on ASIC is developed and the results are shown in Section 6.3.2. Section 6.3.3 develops the implementation and results for FPGA target. Finally, Section 6.4 concludes the chapter and reviews the obtained implementation results.

6.3.1 Implementation Context

This thesis developed new structures to implement finely-tuned SRC. To compare their hardware complexity, the implementation of these structure needs to developed on both ASIC and FPGA. This section presents the different implemented hardware architectures, and the implementation targets and technologies used to obtain the results in the following sections. The quantization of these modules is done according to the methodology developed in Chapter 5.

Implemented SRC Modules: In order to have meaningful hardware complexity measures of the developed structures in Chapter 4, the classical Farrow, Newton, and CIC filter structures are implemented and their complexity measures are used for referencing. The Farrow structure is implemented to compare the possible savings achieved by using the Newton structure as a replacement arbitrary SRC solution. While the CIC filter structures of different orders are implemented as they are supposed to be the SRC solution of the lowest complexity requiring no multipliers. The implemented hardware modules and their designations are listed below:

1. `CIC2`: Pipelined classical CIC filter structure order 2 (linear interpolation)
2. `CIC4`: Pipelined classical CIC filter structure order 4 (Spline interpolation order 3)
3. `CIC6`: Pipelined classical CIC filter structure order 6 (Spline interpolation order 5)
4. `FAR5`: Pipelined classical Farrow structure for Lagrange interpolation of order 5
5. `NPL3`: Pipelined classical Newton structure for Lagrange interpolation of order 3
6. `NPL5`: Pipelined classical Newton structure for Lagrange interpolation of order 5
7. `NPH3`: Pipelined modified Newton structure for Hermite interpolation of order 3
8. `NPH5`: Pipelined modified Newton structure for Hermite interpolation of order 5
9. `NPRH`: Pipelined reconfigurable Newton structure capable of implementing both Lagrange and Hermite interpolation of order 3, and any frequency response in between the two
10. `NPRS`: Pipelined reconfigurable Newton structure capable of implementing both Lagrange and Spline interpolation of order 3, and any frequency response in between the two
11. `NAL3`: ALU-based classical Newton structure for Lagrange interpolation of order 3
12. `NAL5`: ALU-based classical Newton tructure for Lagrange interpolation of order 5
13. `NTL3`: Time-shared ALU-based classical Newton structure of order 3
14. `NTL5`: Time-shared ALU-based classical Newton structure of order 5

All modules are implemented with an input and output fixed-point quantization on 18 bits. In the case of the Farrow and Newton modules, the SRC factor precision is quantized on 18 bits as well, while the fractional delay is quantized on 6 bits. The internal signals of the ALU-based structures of order 3 and 5 are quantized on 22 and 24 bits respectively. The variable parameter of the reconfigurable structures is quantized on 4 bits. Finally, the time-shared architectures implemented support a maximum of 16 channels.

Implementation targets: FPGAs are often used during the first steps of hardware design for prototyping, development, and testing. Once the final product is ready, and when mass production is needed, the implementation is then usually done on ASIC targets. In this work, the hardware implementation of the above mentioned SRC modules is done on both ASIC and FPGA targets. The ASIC implementation uses the X-FAB XH018 process [XFA17], which is a mixed signal high voltage ASIC technology. Mixed signal means that the implementation contains both analog and digital circuitry. This might be interesting when implementing both the analog and digital front-ends on the same electronic chip. The implementation of digital circuits using the XH018 process is done using a 180 nm technology, using a core voltage of 1.8 volts. The FPGA implementation on the other side is done using Xilinx Virtex-6 XC6VLX365T FPGA [Xil15]. The Virtex series is the flagship family of FPGAs developed by Xilinx. The used FPGA is fabricated using a 60 nm technology, with a core voltage of 1.0 volts.

6.3.2 ASIC Implementation

This section develops the hardware implementation on an ASIC target using the X-FAB XH018 process. The design flow for developing ASIC implementations is first presented and then the internal ASIC architecture is quickly discussed. Finally the implementation results are presented and studied, then the complexity of the newly proposed modified Newton structures are compared with the classical SRC solutions.

ASIC Design Flow: Similarly to software development, the design flow of a hardware module consists of two main tasks: the design itself and the verification process. The design flow of an ASIC implementation is separated into two major parts. First, the frontend design consists of developing the architecture of the hardware module at the logic gate-level, with the possible introduction of some requirements related to the ASIC die layout. Then the backend design takes the gate-level architecture and implements it as an integrated circuit that is fabricated on a silicon wafer. The different steps of each design part are illustrated in Figure 6.12, and are briefly presented below.

The design flow has 6 tasks alongside multiple verification steps. In this thesis, the Cadence ASIC design tools were used to develop the ASIC implementation:

1. Functional Specification: The first step in any design is to define the function of the hardware module, which in this case is sample rate conversion, either coarse or fine, using the CIC, Farrow, or the Newton structures.
2. HDL Description: The second step is to develop the register transfer level (RTL) description of the hardware module using a hardware description language (HDL). In this step, the decision needs to be made as to which one of the implementation strategies presented in the last section should be used. Also during this task, design for testability (DFT) is integrated into the design, which consists of introducing features to verify the hardware during and after manufacturing. Once this task is done, behavioral simulations are run to validate that the developed HDL description corresponds to the functional specification. The Cadence NCLaunch tool is used to synthesize and verify the HDL description.
3. Logic Synthesis: This task consists of developing the gate-level architecture of the hardware module using the provided HDL description. The Cadence RTL Compiler superseded by Genus Synthesis are the tools used to perform this task. These tools take an input the HDL description, and the ASIC process library (in this

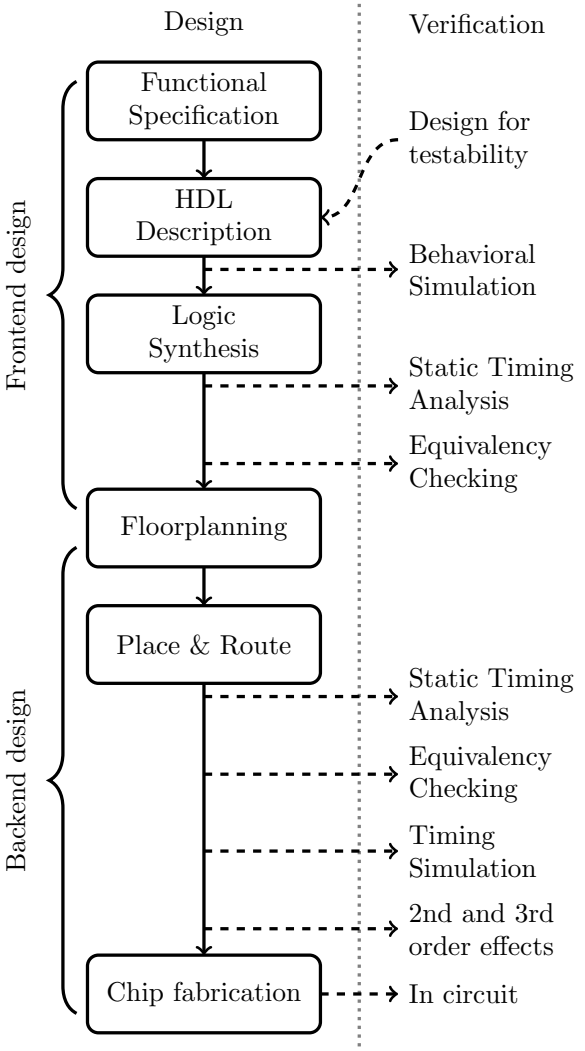


Figure 6.12: ASIC design flow

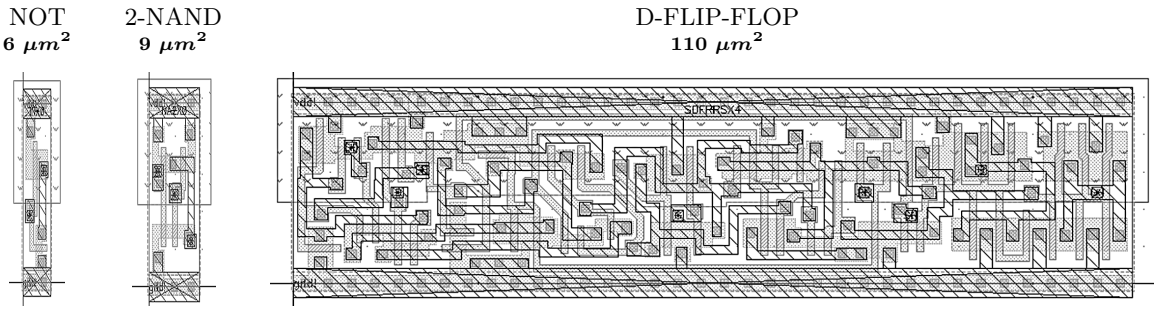


Figure 6.13: Die surface of different X-FAB XH018 logic gates

case the X-FAB XH018) that contains the available logic gates. Following this task, two verification processes are usually done. First, a static timing analysis is performed to ensure that it is still possible to respect the timing constraints using the gate-level architecture found. And second, an equivalency check is needed to verify that the hardware module functioning was not altered through the synthesis process.

4. Floorplanning: This task is usually the border between the frontend and backend design sections, and the rest of the tasks are usually performed at the foundry that will fabricate the integrated circuits. Floorplanning consists of defining the main layout elements the die, such as the positioning of the input and output ports, the placement of the power distribution network, the partitioning of the die as multiple sub-modules, and any other layout requirements.
5. Place & Route: This task is responsible for the main tasks of defining the layout of the implemented ASIC circuit. This step consists of placing the logic gates and connecting them together and to the power source. The result of this step is the different masks that will be used in fabrication. Many verification steps are required after this task before launching the fabrication process. Static timing analysis and equivalency checking are re-run to verify that the timing constraints and the functional specification are always respected respectively. Timing simulations are also needed that consider in more details the properties of the electronic components and their delays, to verify the correct functioning of the implemented hardware. Finally, the nonlinear distortions generated by the electronic components known as the 2nd and 3rd order effects are studied to ensure they do not break the hardware operation. Many other verification steps may also be done to ensure the hardware is ready to be sent to manufacturing.
6. Chip fabrication: Once all verifications are passed, the ASIC circuits are fabricated, and then validated using the DFT features introduced during the HDL description task.

ASIC Complexity: To compare the complexity of the different implementations, a measurement unit is needed. When logic synthesis is run, the implementation is defined using a combination of different gates. These gates are provided by the library of the ASIC process used, and they vary widely in complexity, starting from basic logic gates, up to complicated functions and registers. Three different gates from the X-FAB XH018 process are illustrated in Figure 6.13. A simple one input NOT gate consisting of two transistors is implemented on a surface of 6 μm^2, while a two-input gate NAND gate consisting of four transistors is implemented on a 9 μm^2 surface. However, other gates in the library may be much more complex. An example is a D flip-flop with set, reset, and scan signals, which form one gate that is implemented using numerous transistors on a surface of 110 μm^2. Then it is clear that the number of gates in general is not an accurate complexity measure of an ASIC implementation. In literature, a common practice is to measure the complexity using the surface of the

Table 6.5: ASIC implementation results using the X-FAB XH018 technology

Module	Gates Count	2-NAND Equivalent	Complexity %	Surface (um^2)	Power (mW)	180 MHz Timing
NPL3	10702	30 k	37%	590x590	130	✓
NPL5	19824	58 k	72%	800x800	251	✓
NPH3	10903	28 k	35%	540x540	134	✓
NPH5	14803	41 k	50%	660x660	187	✓
NPRH	14744	40 k	49%	650x650	186	✓
NPRS	12858	34 k	42%	600x600	167	✓
NAL3	09158	20 k	25%	460x460	186	- 184ps
NAL5	11617	26 k	32%	520x520	255	- 491ps
NTL3	08750	18 k	23%	440x440	171	- 190ps
NTL5	10960	23 k	28%	490x490	235	- 351ps
CIC2	04639	12 k	15%	360x360	053	✓
CIC4	06493	19 k	23%	450x450	082	✓
CIC6	08299	25 k	30%	510x510	113	✓
FAR5	27328	81 k	100%	950x950	362	✓

2 input NAND gate as a unit of measurement, and thereby the complexity is expressed as the equivalent number of NAND gates. The implementation results presented next compare the complexity of the different architectures using this measure.

Implementation Results: The implementation follows the ASIC design flow presented in Figure 6.12 up to the place and route task. The main objective is to obtain complexity and power consumption measures of the implementations. The different obtained ASIC implementation results are resumed in Table 6.5. First is the total gate count reported after the logic synthesis task, which includes the different type of gates available in the X-FAB XH018 process library used to implement the hardware module. This count is then rounded to the one thousand nearest equivalent 2 input NAND gates count, which gives a meaningful measure to compare the complexity of the different implementations. The surface measure shows how much surface in required on the ASIC die to implement the hardware module. This measure is obtained after the place and route task, and takes into consideration the extra surface required for routing the internal and external signals. The power consumption estimation is obtained after the logic synthesis step, for an operating frequency of 180 MHz, a signal toggle rate of 20 MHz, and a core voltage of 1.8 V. More accurate power consumption figures can be obtained by simulating the implemented electronic circuit after the place and route task, however this is beyond the interest of this work, and the estimation obtained are sufficient. Finally, Table 6.5 shows which implementations managed to respect the timing constraints corresponding to a 180 MHz operating frequency. The slack of the implementations that did not manage to respect the constraints is shown in picoseconds.

The hardware complexity and power consumption results are illustrated in Figure 6.14 for a visual comparison, with the x-axis sorted by complexity order. The following conclusions and remarks can be drawn from this comparison:

1. The Newton structure offers a better option to implement Lagrange interpolation for fine-tuned SRC. It is seen that a pipeline Newton implementation of order 5 is equivalent to the Farrow implementation while requiring only 72% of the resources. In addition, when it is possible to use the ALU-based architecture, the same operation can be implemented with 35% of the resources.

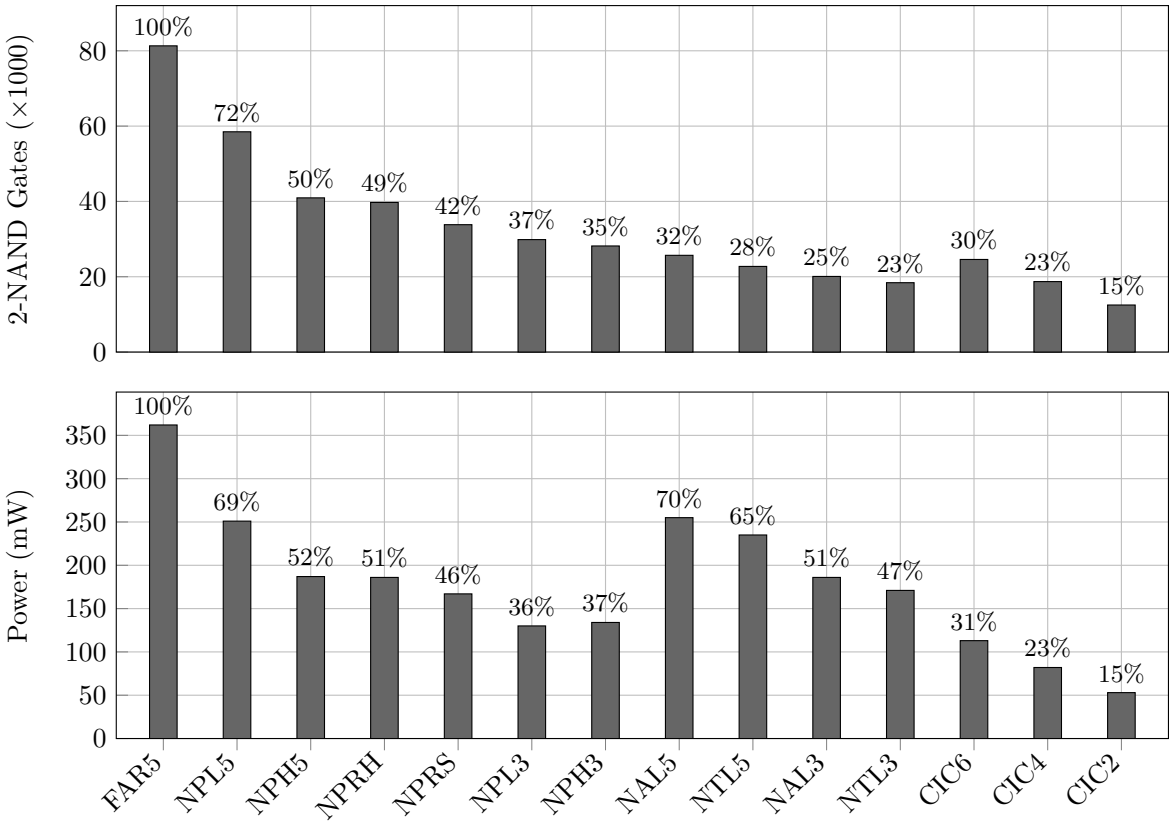


Figure 6.14: ASIC implementation results using the X-FAB XH018 technology

2. It should be noted that a counter intuitive result is the lower complexity and consumption of the time-shared structures compared to their simple ALU-based counterparts, even though added memory resources are required in the former case. This is most probably due to a better optimization by the logic synthesizer that exploits the modification introduced to the time-shared version of the ALU-based architecture.
3. The different implementation options based on the Newton structure offer a wide variety of implementation options for fine-tuned SRC, with a very low complexity comparable to that of coarse SRC implemented using the CIC filters.
4. The relation between hardware complexity and power consumption depends primarily on the implementation strategy used. It can be seen that all the architectures implemented using the pipeline approach have proportional complexity and power consumption levels. However when looking at the ALU-based implementation, even though their hardware complexity is much lower than the pipeline implementations, their power consumption is much higher due to the included memory resources.
5. The maximum operation frequency obtained with the different implementations validate the discussion about the implementation strategies in Section 6.1.2. The pipeline architectures manage to respect the 180 MHz timing constraint, contrary to the ALU-based structure that have longer critical paths limiting their maximum operating frequency.

6.3.3 FPGA Implementation

This section develops the implementation on the Xilinx Virtex-6 FPGA. First, the FPGA design flow used to develop the implementations is presented, and its differences compared to the ASIC design flow are discussed. Second, the FPGA architecture is studied in order

to develop a complexity measure for the implementations. Finally, using these measures, the implementation results obtained are presented and analyzed.

FPGA Design Flow: Compared to the ASIC design flow, designing an FPGA implementation is far less complicated. An FPGA is basically an ASIC implementation of a group of reusable and re-programmable logic resources. Then the design of an FPGA consists of programming these logic resources accordingly to implement a desired function. The design flow is illustrated in Figure 6.15, where it is seen that it consists of the same tasks performed in the ASIC design flow, however the underlying role of the tasks is different, except for the first two tasks (function specification and HDL description) which are the same in both cases. The simplicity of the FPGA design flow explains why FPGAs are usually used for prototyping hardware implementations, due to their reduced complexity resulting in a faster design time. The FPGA design flow tasks differ from the previously presented tasks in Section 6.3.2 as follow:

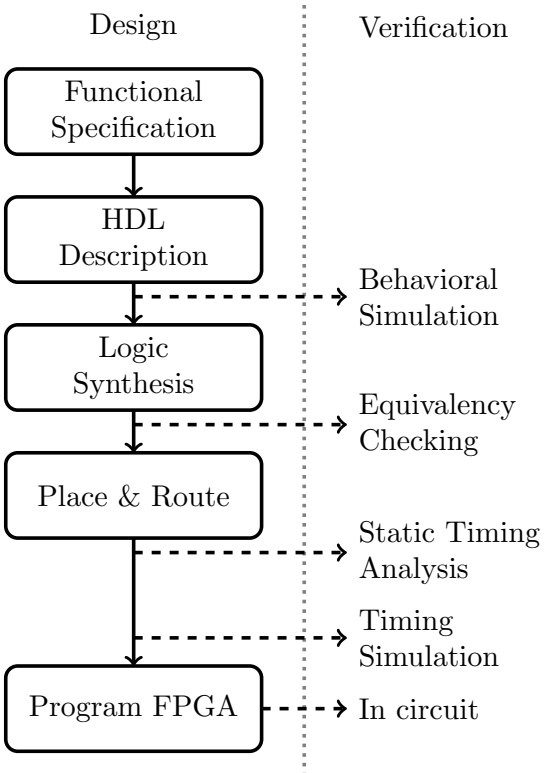


Figure 6.15: FPGA design flow

1. Logic Synthesis: This task has the role of translating the HDL description into a set of logic and arithmetic operations to be implemented using the FPGA resources. In this case it is the library provided by the FPGA manufacturer that is used to develop the gate-level implementation of the hardware module. However the gates in FPGA consist mainly of look-up tables (LUT) that are used to implement different logic functions as explained later below. At the end of this task, equivalency checking may be run to verify the correct synthesis.

2. Place & Route: As its name implies, this task has the role of placing the logic functions using the FPGA resources and routing the signals between them. However an important difference compared to the ASIC design flow is the lack of a floorplanning task. This is since the FPGA is already implemented and the logic resources are placed. However in modern FPGA design softwares there is what is called a floorplanning tool which is used to partition the FPGA resources. However this remains different from an actual floorplanning task responsible for defining the foundation of an electronic circuit. The feature provided by the floorplanning tool is better considered as part of the place & route task. Another difference with the ASIC design flow is that fewer verification tasks are needed. This is because the FPGA logic resources have already been verified and tested by the manufacturer, and the appropriate design tools and features are already provided.

3. Program FPGA: The FPGA design is concluded by programming the designed configuration into the FPGA, and then in-circuit testing may be performed.

FPGA Complexity: Measuring the complexity of an FPGA implementation is much more complicated than measuring the complexity on ASIC. This is due to the way an FPGA imple-

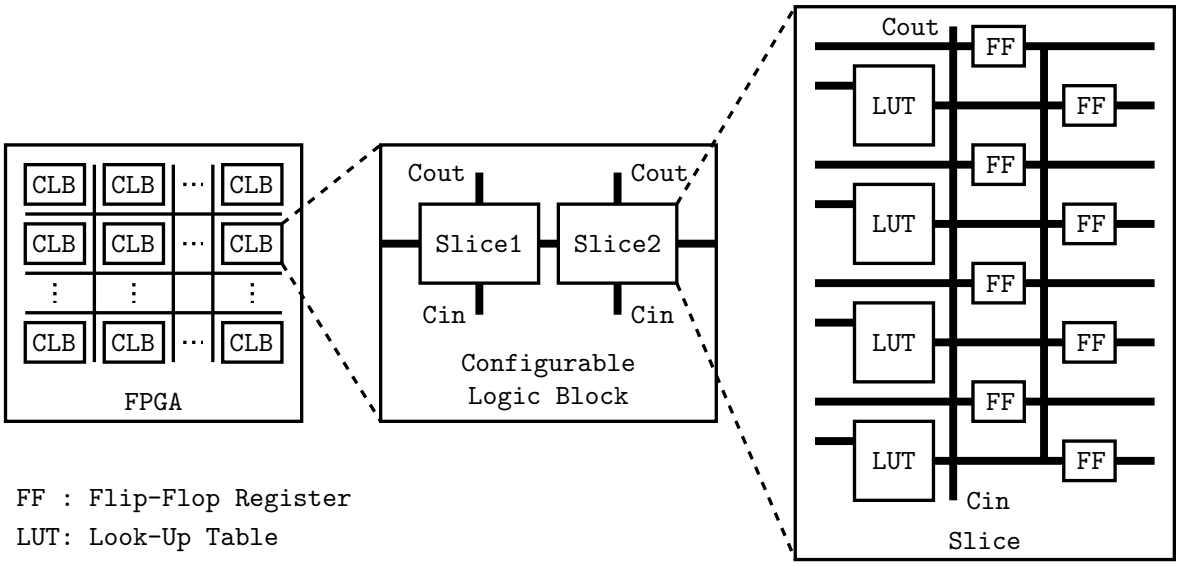


Figure 6.16: Simplified presentation of the Virtex-6 FPGA architecture

mentation is developed. Contrary to ASIC, where a number of different logic gates are available, an FPGA consists of a regular matrix of the same logic units as shown in Figure 6.16. These units are called configurable logic blocks (CLB) in Xilinx FPGA architectures [Xil12], and are inter-connected using the FPGA switching matrix. The CLB has a fixed architecture that contains two slices. Each slice has a carry in (`Cin`) and a carry out (`Cout`) pins, that are directly connected to other slices in adjacent CLB units without passing through the switching network. Each one of the slices has four 6-input look-up tables and 8 flip-flop (FF) registers, that are connected though a network of multiplexers. The illustrated architecture in Figure 6.16 is simplified and does not show other FPGA resources such as integrated multipliers and embedded block random access memory (BRAM). In this development, the implementation is limited to use only CLB units in order to get a simple complexity comparison measure. The implementation results obtained report the number of LUT gates and FF registers used, from which the equivalent count of CLB is deduced to quantify the complexity as done below.

Implementation Results: Using the FPGA design flow shown in Figure 6.15, the different architectures are implemented and the obtained results are resumed in Table 6.6. For each architecture, the number of used LUT gates and FF registers is shown, alongside the equivalent number of CLB units required. The latter provides a measure to compare the different implementations complexity. The power consumption only considers the dynamic power draw, and is found using the Xilinx Xpower Analyzer tool. For this estimation, the operating frequency is supposed to be 160 MHz, the signal toggle rate is 20 MHz, and the core voltage is 1.0 V. The design of the different implementations was performed with a timing constraint corresponding to an operating frequency of 160 MHz. The last column shows which architectures respected this constraints, and the violation for the ones that did not is shown in picoseconds. The complexity and power consumptions results are illustrated in Figure 6.17 for a visual comparison. The x-axis is sorted by thh decreasing order of the ASIC implementations complexity. From these obtained results, the following conclusions can be drawn:

1. The implementation complexity on FPGA is not of the same order as on ASIC. This is clearly seen in the case of ALU-based structures and the CIC filters, that have a much higher complexity relatively to the Farrow structure when compared with the results

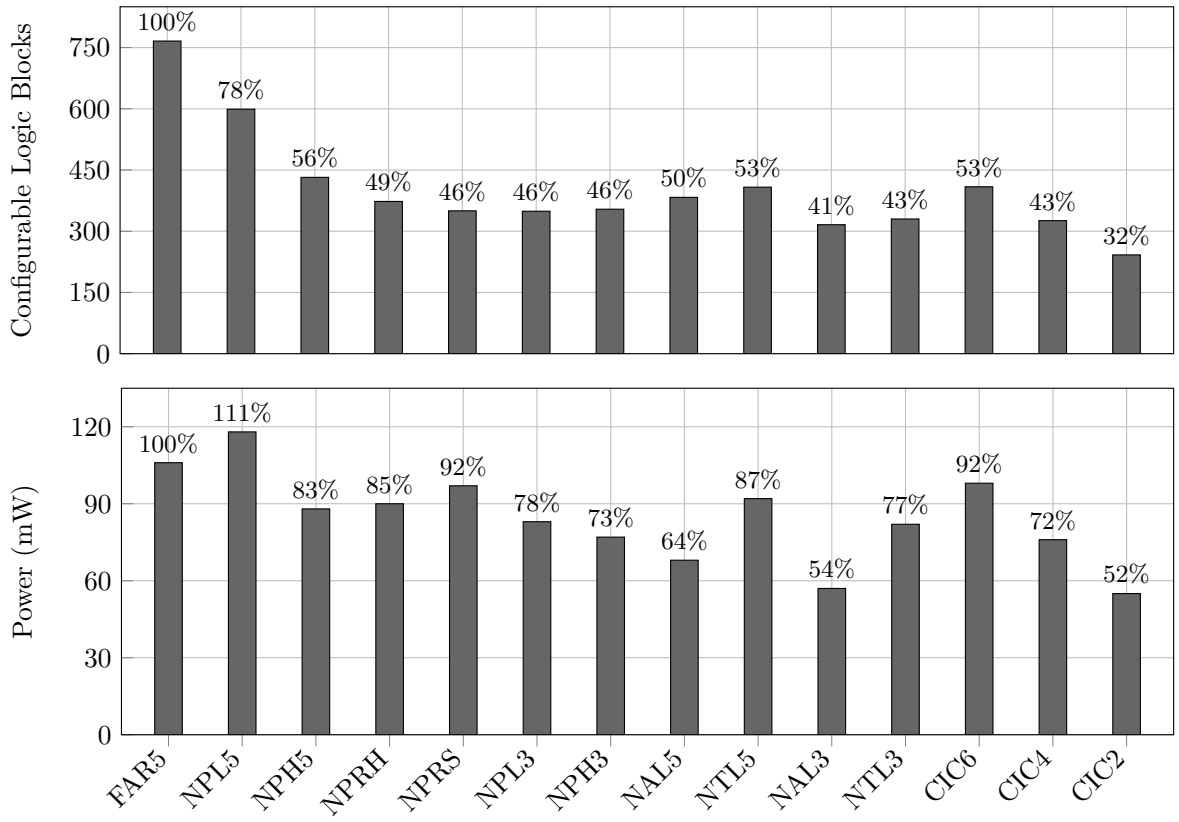


Figure 6.17: FPGA implementation results using the Xilinx Virtex-6

Table 6.6: FPGA implementation results using the Xilinx Virtex-6

Module	REG Count	LUT Count	Equivalent CLB Count	CLB %	Power (mW)	160 MHz Timing
NPL3	1852	2791	349	46%	083	✓
NPL5	3744	4789	599	78%	118	✓
NPH3	1579	2829	354	46%	077	✓
NPH5	2623	3453	432	56%	088	✓
NPRH	2103	2983	373	49%	090	✓
NPRS	1954	2800	350	46%	097	✓
NAL3	0681	2527	316	41%	057	-605ps
NAL5	0965	3066	383	50%	068	-987ps
NTL3	0656	2640	330	43%	082	-401ps
NTL5	0790	3263	408	53%	092	-725ps
CIC2	1989	1939	242	32%	055	✓
CIC4	3081	2607	326	43%	076	✓
CIC6	4173	3275	409	53%	098	✓
FAR5	5545	6128	766	100%	106	✓

obtained on ASIC. This is mainly due to the fact that the FPGA implementations are based on LUT gates, which greatly limit the efficiency and the optimization possibilities of the implementation.

2. The power consumption on FPGA is not correlated with the number of resources used. This is caused by the different routing of each implementation, which may be more complex for implementations using less resources. Another source of increased power consumption are memory elements implemented in the CLB, where a clear increase in consumption is seen in the case of the time-shared architectures.
3. Albeit the limited optimization of FPGA implementations, the fine-tuned SRC solutions based on the Newton structure sill offer an advantageous solution with a complexity level close to that of the CIC filters.

6.4 Conclusion

This chapter developed the hardware implementation aspects of the DFE, with the main focus on the newly developed SRC solutions developed in Chapter 4. The quantization method developed in Chapter 5 was used to quantize the signals in the implemented structures. Different deployment contexts impose a variety of constraints on the hardware implementation. To respect these constraints, a number of implementation strategies were presented, such as the pipeline strategy for high operating speed constraints, the ALU-based strategy for low complexity constraints, or the time-shared strategy for multi-channel low-cost implementations. These strategies were then applied to build the implementation of a number of SRC filters on both ASIC and FPGA. The detailed RTL architectures were developed and a complete definition of the hardware configuration was provided. For both hardware targets, ASIC & FPGA, the results validated the advantages of the different implementation strategies. Most importantly, the results demonstrated how the developed SRC filters in this thesis provide a range of fine-tuned SRC solutions with a low complexity level, approaching that of the simple CIC filters.

Conclusion and Perspectives

General Conclusion

This thesis proposes a generic digital front-end (DFE) architecture for multi-standard internet of things (IoT) modems. Optimizations are proposed for the sample rate conversion (SRC) function, both at the DFE core level, and at the quantization level for an efficient hardware implementation. The developments done in this work offer more flexible SRC solutions with an improved efficiency, improving thereby the multi-standard IoT modem's flexibility and efficiency.

The DFE role in the multi-standard modem is presented in Chapter 1, showing the DFE importance in the efficient deployment of the IoT network. Next, the different core and enhancement DFE functions is presented, and then used to propose a generic DFE architecture for both transmission and reception. This generic architecture shows how the SRC functions, which are used at multiple positions, are the heart of the DFE, and that their optimization is crucial for an efficient DFE implementation.

The unified vision presentation developed in Chapter 2 derives the main five finite impulse response (FIR) based SRC filters from a common starting point, showing the relations between them. The different structures offer both coarse and fine SRC solutions, either favoring high performance filtering on the one hand, or low-cost implementation on the other hand. The developed presentation also provides a guide that helps building any SRC operation of any factor, taking into account practical implementation aspects, and completing the theoretical derivation to help in efficiently implementing the filters in hardware.

Before addressing the generalization of the Newton structure, Chapter 3 develops the direct relation between Lagrange interpolation and the Newton backward difference formula (NBDF), that is then used to demonstrate the convergence of the Newton structure transfer function.

Developing the closed form expressions of the Newton structure generalization in Chapter 4, allows the design of multiple new SRC structures. First, the modified Newton structure implementing Hermite interpolation offers improved filtering performance with a reduced implementation complexity. Second, the proposed reconfigurable Newton structures offer a flexible filter response that can be adapted to the signal processed in order to optimize the filtering performance. Finally, filter optimization techniques enable achieving similar filtering performance relatively to the optimization of the classical Farrow structure, while using a modified Newton structure of substantially lower complexity.

The quantization method of Chapter 5 provides a simple solution for the quantization of digital circuits, able of guaranteeing a given precision constraint, while providing the optimal quantization parameters. The provided example shows the advantage of this method for the quantization of different types of DFE modules.

To efficiently implement the DFE in hardware, Chapter 6 studies the implementation constraints and the strategies used to develop hardware architectures. This study shows the importance of using the appropriate implementation strategy depending on the application requirements. Different SRC filters are implemented using different strategies, and the

comparison of their complexity and power consumption explains the advantages of each architecture. An important result of this work is the implementation of fine SRC at a similar cost of the very simple cascaded-integrator-comb (CIC) filters for coarse SRC, which was inconceivable before this work started.

Perspectives

The studies and developments done in this thesis inspire multiple future research subjects. The most promising topics are presented below.

Sample Rate Conversion Performance Measures: Comparing the performance of SRC filters is a complex task due to the nature of SRC. This is due to the aliasing and image creation effects of manipulating the sampling frequency, which is different for each SRC factor. This results in aliases of the signal falling back in the baseband when rational SRC factors are used. The first point to study is the effect of the aliasing component position on the signal quality, depending if it is close or far from its original position. Then an objective measure needs to be developed that would be universal to all SRC filters for the comparison of the performance of the different solutions. This measure should be adjustable to any SRC factor or filter parameters that might be used. Finally, if the modulation scheme is known, it would also be interesting to go further by developing the relation between this measure and the error vector magnitude (EVM) resulting from the signal degradation caused by the SRC operation. To develop these objective measures, the impulse response of the filter in question should be well defined. In the case of the Newton structure, the extension of the convergence proof from the frequency to the time domain would help in easier manipulation of the analytical expressions of the filter response, which is required to develop these performance measures.

Reconfigurable and Flexible Filter Banks: The filter bank is the next DFE function to study after the SRC, which is responsible for extracting the useful band from a wideband signal. In the multi-standard modem, the captured wideband contains many signals of different bands and carrier frequencies, that will be time variant in most cases. This presents many challenges for the filter bank function that is required to be very flexible. Future research should investigate efficient structure for filter banks that allow having flexible and reconfigurable bands, in terms of both bandwidth and carrier frequency. Finally, SRC is usually required at the filter bank output, and the developments in this thesis will be helpful to efficiently integrate the SRC operation in the filter bank structure.

Further Development of the Quantization Method: The developed optimal quantization methods require the manual derivation of the error expression that is then used by the optimization algorithm. However, this step can be automatized by developing the appropriate software program that takes a description of the filter structure as input, and then finds automatically the quantization error expression. Attention should be given to maintain this method's simplicity by providing simple tools to describe the filter structure. This automation can then be extended to a cascade of modules, and eventually the totality of the DFE or any other signal processing system. Another aspect that needs to be developed in more details is the application of the optimal quantization method to recursive filter structures with an infinite impulse response. The objective would be to identify the resulting challenges and problems, and to propose subsequently the corresponding solutions and modifications of the method.

Optimization of Other DFE Functions: Many optimization possibilities still exist for the DFE in the multi-standard modem, with the main focus being the development of generic

structures that can be used with the largest number, and ideally all, communication technologies. Two DFE functions have a particular priority over the rest of the functions. The first is the automatic gain control, that is often implemented on a case by case basis. The appropriate generic theoretical definition of this function would be very useful to design a generic implementation that can be easily adapted to different signal types and wave forms. The second function is detection that is responsible of recognizing an active transmission in the signal stream. Optimizations of this function are possible regarding signals of constant envelope, that are becoming increasingly more common in the IoT field. Finally, when looking at the DFE from a higher level, each function should also be studied from an implementation mutuality point of view, which concerns the possibility to process multiple signals of different technologies using the same implemented modules.

Appendices

Appendix A

Duality and Transposition

Chapter 2 presented the anti-imaging filters which are destined to removing images caused by up-sampling, and are preferred for interpolation SRC applications of a factor $R > 1$. The counterparts of these filters are the anti-aliasing filters, that are designed to protect the baseband from aliasing caused by down-sampling, and are practically preferred for SRC applications of a factor $R < 1$. Anti-imaging filtering is achieved by positioning the filter zeros on multiples of F_{in}. Analogously the anti-aliasing filtering is achieved by having the zeros on multiples of F_{out}, which are the components that will alias back to $f = 0$ Hz after down-sampling.

A.1 Definition

Many functions in digital signal processing possess the duality property. A dual system is defined as one that performs an operation complementary to that of an original system [Cro83]. Some examples of dual function are: Modulation and Demodulation, Digital Up Conversion (DUC) and Digital Down Conversion (DDC), Up-sampling and Down-sampling. This is also the case for the anti-imaging filtering that is the dual of anti-aliasing filtering. The transposition theorems were developed in [Cla78], and are also presented in section 5 of chapter 3 in [Cro83]. Simply stated, a structure can be obtained from its dual through transposition, consisting of three steps:

1. Reversing the direction of all signal flow branches
2. Replacing the nodes with summations and vice versa
3. Interchanging the input and output ports

The transposition theorem also specifies that a linear time-invariant system response is not modified by network transposition. However, in the case of time variant systems, such in the case of up-sampling and down-sampling functions, the result of network transposition is the dual of the original system. The duality property of SRC functions is shown in Figure A.1, where the effects of network transposition are illustrated. First, the input and the output are exchanged which is equivalent to exchanging the sampling frequencies F_{in} and F_{out}. The

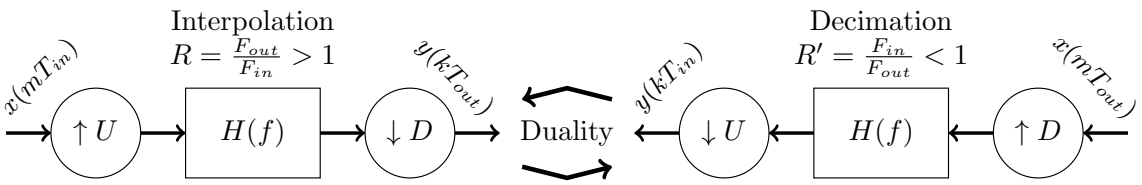


Figure A.1: Duality Property Between Interpolation and Decimation

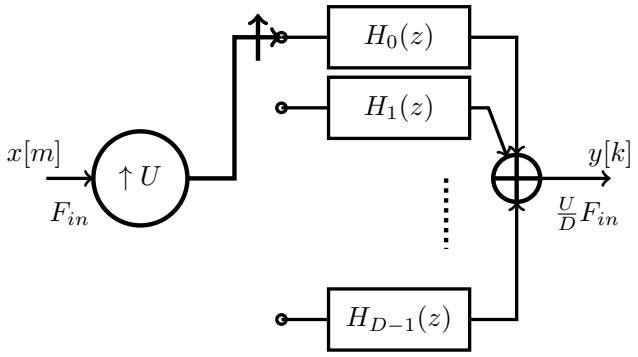


Figure A.2: Polyphase Decimation Structure with $D > U$

FIR filter keeps the same response since it is an LTI system. Therefore the FIR filter after transposition plays the role of an anti-aliasing filter in place of anti-imaging, where instead of having the role of maximally rejecting the images found at multiples of F_{in}, the new role is to maximally reject the frequency components at multiples of F_{out} that risk to alias on the passband components after down-sampling. Finally, the most notable effect of transposition is the SRC factor inversion. Therefore, it is possible to design an anti-aliasing filter for an SRC operation of factor R by using the anti-imaging solutions developed in Chapter 2 through two steps:

1. Develop the anti-aliasing filter for SRC of factor $R' = 1/R$
2. Perform transposition on the AI filter structure

A.2 Transposed SRC Filters

This section presents the transposed versions of the anti-imaging SRC filters presented in Chapter 2, which are the U-F-D, polyphase, Farrow, CIC, and Newton structures. The SRC controller is part of the SRC filter, and its transposition is also developed.

U-F-D and Polyphase Decimation Structures: The U-F-D transposed structure can be simply found through direct application of the transposition theorem, resulting in the decimation U-F-D structure presented in Figure A.1. In the case of the anti-imaging polyphase structure, the multiplexer presented in Figure 2.6-b has the role of a commutator that scrolls through the filters $H_i(z)$ with a step equal to the down-sampling factor D. Applying the transposition theorem on this structure will result in the decimation polyphase structure shown in Figure A.2. The multiplexer that selects one in D outputs from U FIR filters is transposed into a commutator that feeds an up-sampled signal by U to D FIR filters. The filters are not modified by the transposition being linear time-invariant systems. And finally the distribution node to the different filters transposes to a summation node.

Transposed Farrow Structure: The Farrow structure shown in Figure 2.8 has a function not explicitly illustrated, that is the interface between the sampling frequency domains F_{in} and F_{out}. This interface is found at the output of the different FIR filters. This function is known as the hold and sample $H\&S$, and it needed in the case of interpolation, where multiple outputs may be required using the same inputs [Leh09]. Therefore the $H\&S$ block can be seen as the block holding the same value that is used for multiple calculations. The $H\&S$ function possesses a transpose which is the integrate and dump $I\&D$ [Hun08]. In decimation there are more inputs than outputs, however all of the inputs can be used to improve the filtering performance. This intuitively justifies the role of the $I\&D$ blocks, that is to accumulate the

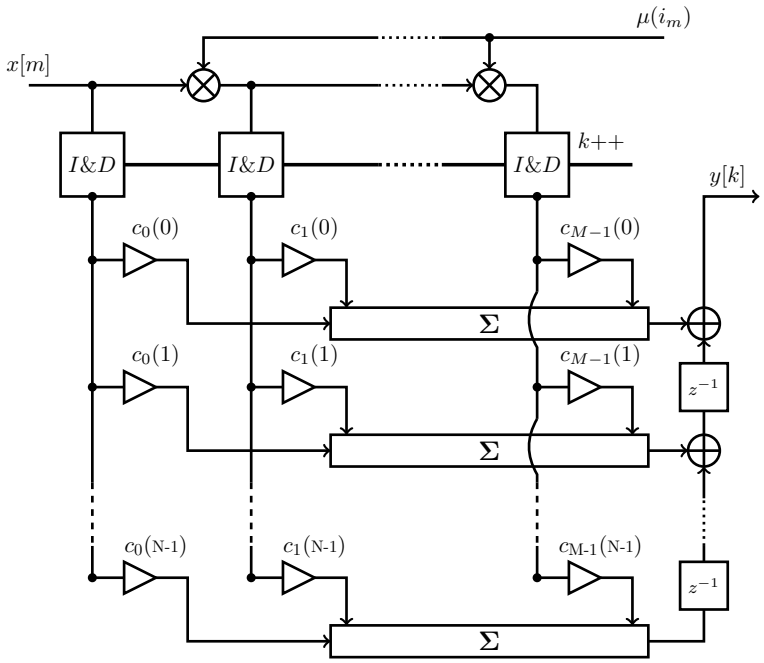


Figure A.3: Transposed Farrow Structure

inputs until a new output is required. Applying the transposition theorem on the Farrow structure results in the transposed structure illustrated in Figure A.3. The input and output are exchanged, as well as the nodes and summations. The $H\&S$ functions are replaced by their transposed $I\&D$ blocks. The fractional delay $\mu(i_m)$ is now referenced with respect to the input index m, while the $I\&D$ blocks are controlled following the output index k. The evolution of these parameters is now managed by the decimation SRC controller, which is the transposed version of the controller presented in Chapter 2.

SRC Controller: The transposed SRC filters evaluate the input and output indexes differently that their dual counterparts. This is because the transposed filters are normalized with respect to the output sampling frequency instead of the input one. Algorithm 2 provided below describes the implementation of the decimation SRC filter controller that is used with the transposed SRC filters. Triggering the $I\&D$ means that the value accumulated is forwarded to the output.

Algorithm 2 Algorithm of the decimation SRC controller

Input: U, D
Output: k, i_m, m
1: $k = 0$, $i_m = 0$, $m = 0$
2: **while** True **do**
3: $i_m = i_m - U$
4: **while** $i_m < U$ **do**
5: $k++$ {Calculate output k and insert in $FIFO_y$}
6: $i_m = i_m + D$
7: Trigger and Reset $I\&D$
8: **end while**
9: $m++$ {Get next input from $FIFO_x$}
10: Accumulate new input in $I\&D$
11: **end while**

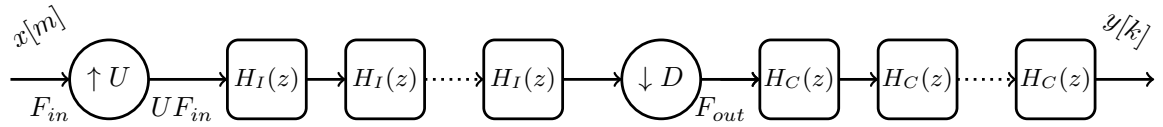


Figure A.4: CIC decimation filter with $D > U$

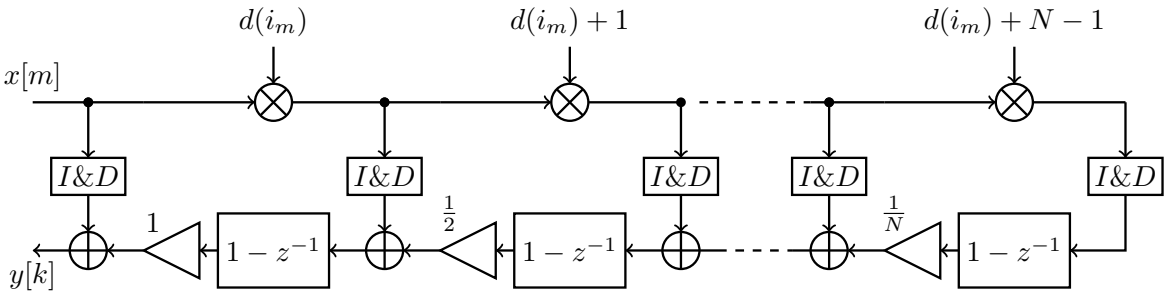


Figure A.5: The Decimation Newton Structure of order N

CIC Decimation Filter: The case of the CIC filters is very straight-forward, where the transposition will only affect the down-sampling and the up-sampling blocks, resulting in their dual function. The structure for a decimation SRC operation of a factor $R = U/D$ with $D > U$ is shown in Figure A.4. In this case, the comb blocks are placed after the decimation operation, resulting in the zeros being positioned on multiples of F_{out}.

Newton Decimation Structure: Using the same principals of duality between the $H\&S$ and the $I\&D$ blocks presented for the Farrow structure, the Newton decimation structure is obtained through direct application of the transposition theorem, resulting in the structure shown in Figure A.5.

List of Publications

International Conferences

C1 Zeineddine, A.; Nafkha, A.; Moy, C.; Paquelet, S. & Jezequel, P.-Y. Variable fractional delay filter: A novel architecture based on hermite interpolation 2018 25th International Conference on Telecommunications (ICT), 2018, 93-97

C2 Zeineddine, A.; Paquelet, S.; Nafkha, A.; Moy, C. & Jezequel, P.-Y. Generalization and Coefficients Optimization of the Newton Structure 2018 25th International Conference on Telecommunications (ICT), 2018, 98-103

C3 Zeineddine, A.; Paquelet, S.; Kanj, M.; Moy, C.; Nafkha, A. & Jezequel, P. Reconfigurable Newton structure for sample rate conversion 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2018, 271-275

C4 Zeineddine, A.; Paquelet, S.; Nafkha, A.; Jezequel, P.-Y. & Moy, C. Efficient Arbitrary Sample Rate Conversion for Multi-Standard Digital Front-Ends 2019 17th International IEEE NEW Circuits and Systems Conference (NEWCAS), 2019

International Journals

J1 Paquelet, S.; Zeineddine, A.; Nafkha, A.; Jezequel, P. & Moy, C., Convergence of the Newton Structure Transfer Function to the Ideal Fractional Delay Filter, IEEE Signal Processing Letters.

Patent

P1 Zeineddine, A.; Paquelet, S.; Filtre interpolateur numérique, dispositif de changement de rythme et équipement de réception correspondants, INPI n°FR17 62632

Bibliography

[3GP15] 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC). *Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT).* V13.1.0 [Online, accessed 14/02/2019]. 2015. URL: http://www.3gpp.org/dynareport/45820.htm.

[3GP18] 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC). *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception.* V16.0.0 [Online, accessed 14/02/2019]. 2018. URL: https://www.3gpp.org/DynaReport/36101.htm.

[Aam01] T. Aamodt. "Floating-point to fixed-point compilation and embedded architectural support". MA thesis. University of Toronto, 2001.

[Ada14] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver. "IEEE 802.11 AH: the WiFi approach for M2M communications". In: *IEEE Wireless Communications* 21.6 (2014), pp. 144–152. ISSN: 1536-1284. DOI: 10.1109/MWC.2014.7000982.

[Alt07a] Altera Corporation. *Accelerating DUC & DDC System Designs for WiMAX.* Application Note 421 [Online, accessed 29/10/2018]. May 2007. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an421.pdf.

[Alt07b] Altera Corporation. *Tool Flow for Design of Digital IF for Wireless Systems.* Application Note 442 [Online, accessed 29/10/2018]. 2007. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an442.pdf.

[Alt07c] Altera Corporation. *Understanding CIC Compensation Filters.* Application Note 455 [Online, accessed 29/10/2018]. Apr. 2007. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an455.pdf.

[Alt09] Altera Corporation. *Simplifying Simultaneous Multimode RRH Design.* White paper WP-01097-1.0 [Online, accessed 29/10/2018]. Mar. 2009. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01097-arria-ii-gx-multimode-rrh.pdf.

[Anz13] D. Anzaldo. *Navigate the AFE and data-converter maze in mobile wireless terminals.* Application Note. Maxim Integrated Application Note 5519 [Online, accessed 29/10/2018]. May 2013. URL: https://www.maximintegrated.com/en/app-notes/index.mvp/id/5519.

[Bab04] D. Babic and M. Renfors. "Reconstruction of non-uniformly sampled signal using transposed Farrow structure". In: *Proceedings of the 2004 International Symposium on Circuits and Systems. ISCAS'04.* Vol. 3. 2004, pp. III–221. DOI: 10.1109/ISCAS.2004.1328723.

[Bab05] D. Babic and M. Renfors. "Power efficient structure for conversion between arbitrary sampling rates". In: *IEEE Signal Processing Letters* 12.1 (2005), pp. 1–4. ISSN: 1070-9908. DOI: 10.1109/LSP.2004.838193.

[Bab02] D. Babic, J. Vesma, T. Saramaki, and M. Renfors. "Implementation of the transposed Farrow structure". In: *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002.* Vol. 4. IEEE. 2002, pp. IV–5. DOI: 10.1109/ISCAS.2002.1010374.

[Bel76] M. Bellanger, G. Bonnerot, and M. Coudreuse. "Digital filtering by polyphase network: Application to sample-rate alteration and filter banks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.2 (1976), pp. 109–114. ISSN: 0096-3518. DOI: 10.1109/TASSP.1976.1162788.

[Bis01] C. Bisdikian. "An overview of the Bluetooth wireless technology". In: *IEEE Communications magazine* 39.12 (2001), pp. 86–94. ISSN: 0163-6804. DOI: 10.1109/35.968817.

[Bre11] R. Bregovic, Y. J. Yu, T. Saramaki, and Y. C. Lim. "Implementation of linear-phase FIR filters for a rational sampling-rate conversion utilizing the coefficient symmetry". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.3 (2011), pp. 548–561. ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2072350.

[Cac02] D. Cachera and T. Risset. "Advances in bit width selection methodology". In: *Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on*. IEEE. 2002, pp. 381–390. DOI: 10.1109/ASAP.2002.1030737.

[Can06] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie. "A metric for automatic word-length determination of hardware datapaths". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.10 (2006), pp. 2228–2231. DOI: 10.1109/TCAD.2005.862733.

[Car84] C. Caraiscos and B. Liu. "A roundoff error analysis of the LMS adaptive algorithm". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.1 (1984), pp. 34–41. DOI: 10.1109/TASSP.1984.1164286.

[Cat88] F. Catthoor, H. De Man, and J. Vandewalle. "Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters". In: *International journal of circuit theory and applications* 16.4 (1988), pp. 371–390. DOI: 10.1002/cta.4490160404.

[Che95] A. Chen, R. McDanell, M. Boytim, and R. Pogue. "Modified CORDIC demodulator implementation for digital IF-sampled receiver". In: *Proceedings of GLOBECOM '95*. Vol. 2. Nov. 1995, 1450–1454 vol.2. DOI: 10.1109/GLOCOM.1995.502642.

[Che12] D. Chen, T. Yan, J. Jin, C. Mao, Y. Lu, W. Pan, and J. Zhou. "A tri-mode Compass/GPS/Galileo RF receiver with all-digital automatic gain control loop". In: *Analog Integrated Circuits and Signal Processing* 70.1 (2012), pp. 69–77. DOI: 10.1007/s10470-011-9656-z.

[Cho96] J.-I. Choi, H.-S. Jun, and S.-Y. Hwang. "Efficient hardware optimisation algorithm for fixed point digital signal processing ASIC design". In: *Electronics Letters* 32.11 (1996), pp. 992–994. DOI: 10.1049/el:19960703.

[Cla78] T. Claasen and W. Mecklenbräuker. "On the transposition of linear time-varying discrete-time networks and its application to multirate digital systems". In: *Philips journal of research* 33.1 (1978), pp. 78–102. URL: http://www.extra.research.philips.com/hera/people/aarts/_Philips%20Bound%20Archive/PJR/PJR-33_34-1978_79_A-078.pdf.

[Cma99] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens. "A methodology and design environment for DSP ASIC fixed point refinement". In: *Proceedings of the conference on Design, automation and test in Europe*. ACM. 1999, p. 56. DOI: 10.1109/DATE.1999.761133.

[Cre08] S. Creaney and I. Kostarnov. *Designing Efficient Digital Up and Down Converters for Narrowband Systems*. Application Note XAPP1113 [Online, accessed 30/10/2018]. Xilinx Inc. Nov. 2008. URL: https://www.xilinx.com/support/documentation/application_notes/xapp1113.pdf.

[Cro75] R. Crochiere. "A new statistical approach to the coefficient word length problem for digital filters". In: *IEEE Transactions on Circuits and Systems* 22.3 (1975), pp. 190–196. DOI: 10.1109/TCS.1975.1084022.

[Cro75] R. E. Crochiere and L. Rabiner. "Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 23.5 (1975), pp. 444–456. ISSN: 0096-3518. DOI: 10.1109/TASSP.1975.1162719.

[Cro83] R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing*. ISBN-10: 9780136051626. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[Da 14] L. Da Xu, W. He, and S. Li. "Internet of things in industries: A survey". In: *IEEE Transactions on industrial informatics* 10.4 (2014), pp. 2233–2243. DOI: 10.1109/TII.2014.2300753.

[De 98] L. De Cosier, M. Ade, R. Lauwereins, and J. Peperstrate. "Code generation for compiled bit-true simulation of DSP application". In: *Proceedings. 11th International Symposium on System Synthesis (Cat. No.98EX210)*. 1998. DOI: 10.1109/ISSS.1998.730590.

[Den07] T. B. Deng. "Coefficient-Symmetries for Implementing Arbitrary-Order Lagrange-Type Variable Fractional-Delay Digital Filters". In: *IEEE Transactions on Signal Processing* 55.8 (2007), pp. 4078–4090. ISSN: 1053-587X. DOI: 10.1109/TSP.2007.893967.

[Din08] L. Ding, Z. Ma, D. R. Morgan, M. Zierdt, and G. T. Zhou. "Compensation of frequency-dependent gain/phase imbalance in predistortion linearization systems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55.1 (2008), pp. 390–397. ISSN: 1549-8328. DOI: 10.1109/TCSI.2007.910545.

[Dlo15] N. Dlodlo and J. Kalezhi. "The internet of things in agriculture for sustainable rural development". In: *2015 international conference on emerging trends in networks and computer communications (ETNCC)*. IEEE. 2015, pp. 13–18. DOI: 10.1109/ETNCC.2015.7184801.

[Dol09] G. J. Dolecek. "Simple wideband CIC compensator". In: *Electronics Letters* 45.24 (2009), pp. 1270–1272. ISSN: 0013-5194. DOI: 10.1049/el.2009.1860.

[Dol08] G. J. Dolecek and S. K. Mitra. "Simple method for compensation of CIC decimation filter". In: *Electronics Letters* 44.19 (2008), pp. 1162–1163. ISSN: 0013-5194. DOI: 10.1049/el:20081603.

[Doo99] S. R. Dooley, R. W. Stewart, and T. S. Durrani. "Fast on-line B-spline interpolation". In: *Electronics Letters* 35.14 (1999), pp. 1130–1131. ISSN: 0013-5194. DOI: 10.1049/el:19990825.

[Du03] Q. Du, M. Jiang, G. Lin, and N. Sun. "ALL-digital AGC in CDMA base station receiver". In: *International Conference on Communication Technology Proceedings, 2003. ICCT 2003.* Vol. 2. IEEE. 2003, pp. 1037–1041. DOI: 10.1109/ICCT.2003.1209707.

[Eru93] L. Erup, F. M. Gardner, and R. A. Harris. "Interpolation in digital modems. II. Implementation and performance". In: *IEEE Transactions on Communications* 41.6 (1993), pp. 998–1008. ISSN: 0090-6778. DOI: 10.1109/26.231921.

[Ett] Ettus Research. *USRP X300 and X310 X Series Datasheet*. [Online, accessed 22/03/2019]. URL: http://www.ettus.com/wp-content/uploads/2019/01/X300_X310_Spec_Sheet.pdf.

[Eul83] L. Euler. "De eximio usu methodi interpolationum in serierum doctrina". In: *Opuscula Analytica* (1783), pp. 157–210.

[Eva11] D. Evans. *The Internet of Things: How the Next Evolution of the Internet is Changing Everything*. Cisco White Paper. [Online; accessed 29/10/2018]. Apr. 2011. URL: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[Eve90] C. Everhart, D. Nicholas, and E. Caplan. *Email exchange on interesting uses of networking*. [Online; accessed 29/10/2018]. 1990. URL: https://www.cs.cmu.edu/~coke/coke.history.txt.

[Far88] C. W. Farrow. "A continuously variable digital delay element". In: *1988 IEEE International Symposium on Circuits and Systems*. June 1988, 2641–2645 vol.3. DOI: 10.1109/ISCAS.1988.15483.

[Fer12] A. Fernandez-Vazquez and G. J. Dolecek. "Maximally flat CIC compensation filter: Design and multiplierless implementation". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 59.2 (2012), pp. 113–117. ISSN: 1549-7747. DOI: 10.1109/TCSII.2011.2180093.

[Fet02] G. Fettweis and T. Hentschel. "The Digital Front End: Bridge Between RF and Baseband Processing". In: *Software Defined Radio*. Wiley-Blackwell, 2002. Chap. 6, pp. 151–198. ISBN: 9780470846001. DOI: 10.1002/0470846003.ch6.

[Fra11] A. Franck. "Efficient algorithms for arbitrary sample rate conversion with application to wave field synthesis". PhD thesis. Universitätsbibliothek Ilmenau, 2011.

[Fra09] A. Franck and K. Brandenburg. "A closed-form description for the continuous frequency response of Lagrange interpolators". In: *IEEE Signal Processing Letters* 16.7 (2009), pp. 612–615. ISSN: 1070-9908. DOI: 10.1109/LSP.2009.2020475.

[Fu17] Y. Fu, J. Lei, T. Yunshuai, H. Zhang, Z. Wang, and Z. Xu. "Automatic frequency control". US9680485B2. June 2017.

[Gaj95] D. D. Gajski and F. Vahid. "Specification and design of embedded hardware-software systems". In: *IEEE Design & Test of Computers* 12.1 (1995), pp. 53–67. ISSN: 0740-7475. DOI: 10.1109/54.350695.

[Gao00] Y. Gao, L. Jia, J. Isoaho, and H. Tenhunen. "A comparison design of comb decimators for sigma-delta analog-to-digital converters". In: *Analog Integrated Circuits and Signal Processing* 22.1 (2000), pp. 51–60. DOI: 10.1023/A:1008372010560.

[Gar93] F. M. Gardner. "Interpolation in digital modems. I. Fundamentals". In: *IEEE Transactions on Communications* 41.3 (1993), pp. 501–507. ISSN: 0090-6778. DOI: 10.1109/26.221081.

[Gau86] C. Gauss. "Theoria interpolationis methodo nova tractata Werke band 3, 265–327". In: *Göttingen: Königliche Gesellschaft der Wissenschaften* (1886).

[Gni15] Z. D. R. Gnimpieba, A. Nait-Sidi-Moh, D. Durand, and J. Fortin. "Using Internet of Things technologies for a collaborative supply chain: Application to tracking of pallets and containers". In: *Procedia Computer Science* 56 (2015), pp. 550–557. DOI: 10.1016/j.procs.2015.07.251.

[Gom12] C. Gomez, J. Oller, and J. Paradells. "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology". In: *Sensors* 12.9 (2012), pp. 11734–11753. DOI: 10.3390/s120911734.

[Gor17] A. D. Gore, J. P. Nair, C. S. Park, K. Bynam, S. J. Yun, and Y. J. Hong. "Method and apparatus for automatic gain control in wireless receiver". US9800279B2. Oct. 2017.

[Gor16] A. D. Gore, J. Nair, K. Bynam, Y.-J. Hong, and C. Park. "AGC and DCOC algorithms for sliding IF non-coherent ULP wireless receiver". In: *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*. IEEE. Jan. 2016, pp. 780–784. DOI: 10.1109/CCNC.2016.7444879.

[Gou15] C. Goursaud and J.-M. Gorce. "Dedicated networks for IoT: PHY/MAC state of the art and challenges". In: *EAI endorsed transactions on Internet of Things* 1 (2015), pp. 1–11. DOI: 10.4108/eai.26-10-2015.150597.

[Gra16] S. Grant. *3GPP Low Power Wide Area Technologies-GSMA White Paper*. [Online, accessed 20/02/2019]. GSMA Mobile IoT Industry Alignment group. 2016. URL: https://www.gsma.com/iot/3gpp-low-power-wide-area-technologies-white-paper/.

[Han05] S. H. Han and J. H. Lee. "An overview of peak-to-average power ratio reduction techniques for multicarrier transmission". In: *IEEE Wireless Communications* 12.2 (Apr. 2005), pp. 56–65. ISSN: 1536-1284. DOI: 10.1109/MWC.2005.1421929.

[Har97] F. Harris. "Performance and design considerations of the Farrow filter when used for arbitrary resampling of sampled time series". In: *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems amp; Computers*. Vol. 2. 1997, 1745–1749 vol.2. DOI: 10.1109/ACSSC.1997.679201.

[Hei02] G. Heinzel, A. Rudiger, and R. Schilling. *Spectrum and spectral density estimation by the Discrete Fourier transform (DFT)*. [Online, accessed 29/10/2018]. Feb. 2002. URL: https://holometer.fnal.gov/GH_FFT.pdf.

[Hen00] M. Henker and G. Fettweis. "Combined Filter for Sample Rate Conversion, Matched Filtering and Symbol Synchronization in Software Radio Terminals". In: *Proceedings of the European Wireless*. 2000, pp. 61–66. DOI: N/A.

[Hen01] T. Hentschel and G. Fettweis. "Continuous-time digital filters for sample-rate conversion in reconfigurable radio terminals". In: *Frequenz* 55.5-6 (2001), pp. 185–188. DOI: 10.1515/FREQ.2001.55.5-6.185.

[Hen99] T. Hentschel, M. Henker, and G. Fettweis. "The digital front-end of software radio terminals". In: *IEEE Personal communications* 6.4 (Aug. 1999), pp. 40–46. ISSN: 1070-9916. DOI: 10.1109/98.788214.

[Her92] E. Hermanowicz. "Explicit formulas for weighting coefficients of maximally flat tunable FIR delayers". In: *Electronics Letters* 28.20 (1992), pp. 1936–1937. ISSN: 0013-5194. DOI: 10.1049/el:19921239.

[Hie10] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke. "The IEEE 802.11 universe". In: *IEEE Communications Magazine* 48.1 (2010), pp. 62–70. ISSN: 0163-6804. DOI: 10.1109/MCOM.2010.5394032.

[Hog81] E. Hogenauer. "An economical class of digital filters for decimation and interpolation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.2 (Apr. 1981), pp. 155–162. ISSN: 0096-3518. DOI: 10.1109/TASSP.1981.1163535.

[Hol14] L. Hollevoet, F. Naessens, P. Raghavan, S. Pollin, and E. L. Estraviz. "Digital front-end circuit and method for using the same". US8861656B2. Oct. 2014.

[Hor08] F. Horlin and A. Bourdoux. *Digital compensation for analog front-ends: a new approach to wireless transceiver design*. ISBN: 978-0-470-51708-6. John Wiley & Sons, 2008.

[Hsi87] C.-C. Hsiao. "Polyphase filter matrix for rational sampling rate conversions". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87*. Vol. 12. IEEE. 1987, pp. 2173–2176. DOI: 10.1109/ICASSP.1987.1169404.

[Hsu16] C.-L. Hsu and J. Lin. "An empirical examination of consumer adoption of Internet of Things services: Network externalities and concern for information privacy perspectives". In: *Computers in Human Behavior* 62 (Sept. 2016), pp. 516–527. DOI: 10.1016/j.chb.2016.04.023.

[Hua12] X. Huang, Y. J. Guo, and J. A. Zhang. "Sample rate conversion using B-spline interpolation for OFDM based software defined radios". In: *IEEE Transactions on Communications* 60.8 (Aug. 2012), pp. 2113–2122. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2012.062511.110318.

[Hun08] M. T. Hunter. "Design of Polynomial-based Filters for Continuously Variable Sample Rate Conversion with Applications in Synthetic Instrumentation and Software Defined Radio". PhD thesis. University of Central Florida Orlando, Florida, 2008.

[Int05] International Telecommunications Union. *ITU Internet Reports 2005: The Internet of Things*. 2005. URL: http://handle.itu.int/11.1002/pub/800eae6f-en.

[Jac70] L. B. Jackson. "On the interaction of roundoff noise and dynamic range in digital filters". In: *Bell System Technical Journal* 49.2 (1970), pp. 159–184. DOI: 10.1002/j.1538-7305.1970.tb01763.x.

[Jer77] A. J. Jerri. "The Shannon sampling theorem - Its various extensions and applications: A tutorial review". In: *Proceedings of the IEEE* 65.11 (1977), p. 1565. DOI: 10.1109/PROC.1977.10771.

[Jia02] W. Jiancheng and Y. Sanmin. "A novel and efficient finitely precise design of linear-phase FIR filters". In: *6th International Conference on Signal Processing, 2002*. Vol. 1. IEEE. 2002, pp. 158–161. DOI: 10.1109/ICOSP.2002.1181013.

[Joh03] H. Johansson and P. Lowenborg. "On the design of adjustable fractional delay FIR filters". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 50.4 (2003), pp. 164–169. ISSN: 1057-7130. DOI: 10.1109/TCSII.2003.809712.

[Joh98] R. Johnson, P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, and R. A. Casas. "Blind equalization using the constant modulus criterion: a review". In: *Proceedings of the IEEE* 86.10 (Oct. 1998), pp. 1927–1950. ISSN: 0018-9219. DOI: 10.1109/5.720246.

[Jor02] R. Jordan and C. T. Abdallah. "Wireless communications and networking: an overview". In: *IEEE Antennas and Propagation Magazine* 44.1 (2002), pp. 185–193. ISSN: 1045-9243. DOI: 10.1109/74.997963.

[Ked98] H. Keding, M. Willems, M. Coors, and H. Meyr. "FRIDGE: a fixed-point design and simulation environment". In: *Proceedings of the conference on Design, automation and test in Europe*. IEEE Computer Society. 1998, pp. 429–435. DOI: 10.1109/DATE.1998.655893.

[Kho15] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin. "A survey on IEEE 802.11 ah: An enabling networking technology for smart cities". In: *Computer Communications* 58 (2015), pp. 53–69. DOI: 10.1016/j.comcom.2014.08.008.

[Kim95] S. Kim, K.-I. Kum, and W. Sung. "Fixed-point optimization utility for C and C++ based digital signal processing programs". In: *VLSI Signal Processing, VIII, 1995. IEEE Signal Processing Society [Workshop on]*. IEEE. 1995, pp. 197–206. DOI: 10.1109/VLSISP.1995.527491.

[Kim98a] S. Kim, K.-I. Kum, and W. Sung. "Fixed-point optimization utility for C and C++ based digital signal processing programs". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 45.11 (1998), pp. 1455–1464. DOI: 10.1109/82.735357.

[Kim94] S. Kim and W. Sung. "Fixed-point simulation utility for C and C++ based digital signal processing programs". In: *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*. Vol. 1. 1994, 162–166 vol.1. DOI: 10.1109/ACSSC.1994.471437.

[Kim98b] S. Kim and W. Sung. "Fixed-point error analysis and word length optimization of 8/spl times/8 IDCT architectures". In: *IEEE Transactions on Circuits and Systems for Video Technology* 8.8 (1998), pp. 935–940. DOI: 10.1109/76.736720.

[Koo96] P. J. Kootsookos and R. C. Williamson. "FIR approximation of fractional sample delay systems". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 43.3 (1996), pp. 269–271. ISSN: 1057-7130. DOI: 10.1109/82.486473.

[Kra00] T. P. Krauss, M. D. Zoltowski, and G. Leus. "Simple MMSE equalizers for CDMA downlink to restore chip sequence: comparison to zero-forcing and RAKE". In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. Vol. 5. June 2000, 2865–2868 vol.5. DOI: 10.1109/ICASSP.2000.861120.

[Kum01] K.-I. Kum and W. Sung. "Combined word-length optimization and high-level synthesis of digital signal processing systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20.8 (2001), pp. 921–930. DOI: 10.1109/43.936374.

[Kwe97] A. Y. Kwentus, Z. Jiang, and A. N. Willson. "Application of filter sharpening to cascaded integrator-comb decimation filters". In: *IEEE Transactions on Signal Processing* 45.2 (1997), pp. 457–467. ISSN: 1053-587X. DOI: 10.1109/78.554309.

[Laa96] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine. "Splitting the unit delay [FIR/all pass filters design]". In: *IEEE Signal Processing Magazine* 13.1 (Jan. 1996), pp. 30–60. ISSN: 1053-5888. DOI: 10.1109/79.482137.

[Lam16] D. Lamb, L. Chamon, and V. Nascimento. "Efficient filtering structure for spline interpolation and decimation". In: *Electronics Letters* 52.1 (2016), pp. 39–41. ISSN: 0013-5194. DOI: 10.1049/el.2015.1957.

[Leh09] V. Lehtinen and M. Renfors. "Structures for interpolation, decimation, and nonuniform sampling based on Newton's interpolation formula". In: *8th international conference on Sampling Theory and Applications. SAMPTA'09*. HAL ID : hal-00451769. 2009, Special–session. DOI: N/A.

[Li04] W. Li and M. Tomisawa. "Transposed-Farrow-structure-based multirate filters for symbol timing synchronization in software defined radio (SDR)". In: *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th.* Vol. 3. IEEE. 2004, pp. 1668–1672. DOI: 10.1109/VETECF.2004.1400318.

[Li11] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, and X. Lin. "Smart community: an internet of things application". In: *IEEE Communications Magazine* 49.11 (2011), pp. 68–75. DOI: 10.1109/MCOM.2011.6069711.

[Lia08] C.-F. Liao and S.-I. Liu. "40 Gb/s transimpedance-AGC amplifier and CDR circuit for broadband data receivers in 90 nm CMOS". In: *IEEE Journal of Solid-State Circuits* 43.3 (2008), pp. 642–655. ISSN: 0018-9200. DOI: 10.1109/JSSC.2007.916626.

[Lip92] S. P. Lipshitz, R. A. Wannamaker, and J. Vanderkooy. "Quantization and dither: A theoretical survey". In: *Journal of the audio engineering society* 40.5 (1992). [Online, accessed 29/10/2018], pp. 355–375. URL: http://www.aes.org/e-lib/browse.cfm?elib=7047.

[Lit12] Littelfuse, Inc. *Distributed Base Stations*. Application Note [Online, accessed 20/02/2019]. 2012. URL: https://www.littelfuse.com/~/media/electronics_technical/application_notes/dbs/littelfuse_distributed_base_station_application_note.pdf.

[FaL11] L. Fa-Long, ed. *Digital Front-End in Wireless Communications and Broadcasting: Circuits and Signal Processing*. United Kingdom: Cambridge University Press, 2011. DOI: 10.1017/CBO9780511744839.

[Lóp02] F. López, J. Vesma, and M. Renfors. "Defining the wordlength of the fractional interval in interpolation filters". In: *2002 11th European Signal Processing Conference*. IEEE Xplore ID: 7071873. IEEE. 2002, pp. 1–4. DOI: N/A.

[Lu99] W.-S. Lu and T.-B. Deng. "An improved weighted least-squares design for variable fractional delay FIR filters". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 46.8 (1999), pp. 1035–1040. ISSN: 1057-7130. DOI: 10.1109/82.782046.

[Luc16] S. Lucero et al. *IoT platforms: enabling the Internet of Things*. IHS Technology Whitepaper [Online, accessed 29/10/2018]. Mar. 2016. URL: https://cdn.ihs.com/www/pdf/enabling-IOT.pdf.

[Lue90] J. Luecke and M. Jordan. "Programmable digital communications receiver architecture for high data rate avionics and ground applications". In: *9th IEEE / AIAA / NASA Conference on Digital Avionics Systems*. Oct. 1990, pp. 552–556. DOI: 10.1109/DASC.1990.111348.

[Lun14] D. Lund, C. MacGillivray, V. Turner, and M. Morales. *Worldwide and regional internet of things (iot) 2014–2020 forecast: A virtuous circle of proven value and demand*. Tech. rep. IDC 248451. International Data Corporation (IDC), 2014. URL: https://www.business.att.com/content/article/IoT-worldwide_regional_2014-2020-forecast.pdf.

[Lyo05] R. Lyons. *Understanding cascaded integrator-comb filters*. Embedded.com Article. [Online, accessed 30/10/2018]. Mar. 2005. URL: http://www.embedded.com/design/configurable-systems/4006446/Understanding-cascaded-integrator-comb-filters.

[Mah11] R. Mahesh, A. P. Vinod, E. M. Lai, and A. Omondi. "Filter bank channelizers for multi-standard software defined radio receivers". In: *Journal of signal processing systems* 62.2 (2011), pp. 157–171. DOI: 10.1007/s11265-008-0327-y.

[Mar01] E. Martin, J. Toureilles, and C. Nouet. "Conception optimisée d'architectures en précision finie pour les applications de traitement du signal". In: *Traitement du Signal* 18.1 (2001), pp. 47–56. URL: http://hdl.handle.net/2042/2167.

[Mei02] E. Meijering. "A chronology of interpolation: from ancient astronomy to modern signal and image processing". In: *Proceedings of the IEEE* 90.3 (2002), pp. 319–342. ISSN: 0018-9219. DOI: 10.1109/5.993400.

[Mén02] D. Ménard. "Méthodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe sous contrainte de précision". HAL ID : tel-00609159. PhD thesis. Université Rennes 1, 2002. DOI: N/A.

[Men06] D. Menard, D. Chillet, and O. Sentieys. "Floating-to-fixed-point conversion for digital signal processors". In: *EURASIP journal on applied signal processing* 2006 (2006), pp. 1–19. DOI: 10.1155/ASP/2006/96421.

[Men08] D. Menard, R. Rocher, and O. Sentieys. "Analytical fixed-point accuracy evaluation in linear time-invariant systems." In: *IEEE Trans. on Circuits and Systems* 55.10 (2008), pp. 3197–3208. ISSN: 1549-8328. DOI: 10.1109/TCSI.2008.923279.

[Men97] U. Mengali and A. N. D'Andrea. *Synchronization techniques for digital receivers*. ISBN 978-1-4899-1807-9. Springer Science & Business Media, 1997. URL: https://www.springer.com/us/book/9780306457258.

[Mey01] H. Meyr, M. Moeneclaey, and S. Fechtel. *Digital communication receivers: synchronization, channel estimation, and signal processing*. John Wiley & Sons, Inc., 2001. DOI: 10.1002/0471200573.

[Mil09] L. Milic. *Multirate Filtering for Digital Signal Processing: MATLAB Applications: MATLAB Applications*. IGI Global, 2009.

[Mil10] G. V. Milovanović and Z. Udovičić. "Calculation of coefficients of a cardinal B-spline". In: *Applied Mathematics Letters* 23.11 (2010), pp. 1346–1350. DOI: 10.1016/j.aml.2010.06.029.

[Mit95] J. Mitola. "The software radio architecture". In: *IEEE Communications Magazine* 33.5 (May 1995), pp. 26–38. ISSN: 0163-6804. DOI: 10.1109/35.393001.

[Mol11] G. Molnar and M. Vucic. "Closed-form design of CIC compensators based on maximally flat error criterion". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 58.12 (2011), pp. 926–930. DOI: 10.1109/TCSII.2011.2172522.

[Mor83] P. Moroney, A. Willsky, and P. Houpt. "Roundoff noise and scaling in the digital implementation of control compensators". In: *IEEE transactions on acoustics, speech, and signal processing* 31.6 (1983), pp. 1464–1477. DOI: 10.1109/TASSP.1983.1164230.

[Moy10] C. Moy and M. Raulet. "High-level design for ultra-fast software defined radio prototyping on multi-processors heterogeneous platforms". In: *Advances in Electronics and Telecommunications* 1.1 (2010). HAL ID : hal-00488485, Pages–67. DOI: N/A.

[Neu15] T. Neu. *Direct RF conversion: From vision to reality*. Texas Instruments Whitepaper SLYY068 [Online, accessed 30/10/2018]. May 2015. DOI: http://www.ti.com/lit/wp/slyy068/slyy068.pdf.

[Nic17] C. P. Niculescu and M. M. Stănescu. "A note on Abel's partial summation formula". In: *Aequationes mathematicae* 91.6 (2017), pp. 1009–1024. DOI: 10.1007/s00010-017-0504-9.

[Nie89] J. J. Nielsen. "Design of linear-phase direct-form FIR digital filters with quantized coefficients using error spectrum shaping". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.7 (1989), pp. 1020–1026. DOI: 10.1109/29.32280.

[Osb90] V. N. Osborne. *Quantization noise characteristics resulting from Gaussian, negative-exponential, and sinusoidal random input signals*. Tech. rep. [Online, accessed 29/10/2018]. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, 1990. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a230664.pdf.

[Osc95] C. F. A. Oscarsson. "Digital HF receiver". In: *1995 Sixth International Conference on Radio Receivers and Associated Systems*. Sept. 1995, pp. 47–51. DOI: 10.1049/cp:19951115.

[Pai01] D. Painchaud and L. J. Wachter. "Automatic gain control for input to analog to digital converter". US6292120B1. Sept. 2001.

[Pal13] J. Palicot, ed. *Radio engineering: From software radio to cognitive radio*. HAL ID : hal-00657728. John Wiley & Sons, 2013.

[Pan16] N. Pandeya and N. Temple. *About USRP Bandwidths and Sampling Rates*. Ettus Research Application Note AN-177 [Online, accessed 30/10/2018]. Ettus Research. May 2016. URL: https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates.

[Paq18] S. Paquelet and V. Savaux. "On the symmetry of FIR filter with linear phase". In: *Digital Signal Processing* 81 (2018), pp. 57–60. DOI: 10.1016/j.dsp.2018.07.011.

[Par10] K. Parashar, D. Menard, R. Rocher, O. Sentieys, D. Novo, and F. Catthoor. "Fast performance evaluation of fixed-point systems with un-smooth operators." In: *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2010, pp. 9–16. DOI: 10.1109/ICCAD.2010.5654064.

[Pei01] S.-C. Pei and P.-H. Wang. "Closed-form design of maximally flat FIR Hilbert transformers, differentiators, and fractional delayers by power series expansion". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 48.4 (2001), pp. 389–398. DOI: 10.1109/81.917976.

[Pér11] J. P. A. Pérez, S. C. Pueyo, and B. C. López. *Automatic Gain Control*. Springer, 2011. DOI: 10.1007/978-1-4614-0167-4.

[Pri09] V. Prithiviraj, K. Manikandan, C. Prasanna, S. Saranesh, and R. Subramanian. "Front end design of Software Defined BTS for interoperability between GSM and CDMA". In: *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*. IEEE. 2009, pp. 655–659. DOI: 10.1109/WIRELESSVITAE.2009.5172524.

[Qia06] J. Qiao, P. Fu, and S. Meng. "A combined optimization method of finite wordlength fir filters". In: *First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06)*. Vol. 3. IEEE. 2006, pp. 103–106. DOI: 10.1109/ICICIC.2006.378.

[Rah11] M. Rahman and C. L. Sobchak. "Radio frequency receiver having dynamic bandwidth control and method of operation". US7912437B2. Mar. 2011.

[Raj00] K. Rajamani, Y.-S. Lai, and C. Furrow. "An efficient algorithm for sample rate conversion from CD to DAT". In: *IEEE Signal Processing Letters* 7.10 (2000), pp. 288–290. DOI: 10.1109/97.870683.

[Raj94] R. S. Raji. "Smart networks for control". In: *IEEE Spectrum* 31.6 (June 1994), pp. 49–55. ISSN: 0018-9235. DOI: 10.1109/6.284793.

[Ram90] V. Ramachandran, C. S. Gargour, and M. Ahmadi. "Generation of digital transfer functions of the FIR type approximating z/sup-p/q". In: *IEEE International Symposium on Circuits and Systems*. IEEE. 1990, pp. 3260–3262. DOI: 10.1109/ISCAS.1990.112707.

[Ram84] T. Ramstad. "Digital methods for conversion between arbitrary sampling frequencies". In: *IEEE transactions on acoustics, speech, and signal processing* 32.3 (1984), pp. 577–591. DOI: 10.1109/TASSP.1984.1164362.

[Raz17] U. Raza, P. Kulkarni, and M. Sooriyabandara. "Low power wide area networks: An overview". In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 855–873. DOI: 10.1109/COMST.2017.2652320.

[Roc06] R. Rocher. "Évaluation analytique de la précision des systèmes en virgule fixe". HAL ID : tel-00609822. PhD thesis. Université Rennes 1, 2006. DOI: N/A.

[Roy04] S. Roy and P. Banerjee. "An algorithm for converting floating-point computations to fixed-point in MATLAB based FPGA design". In: *Proceedings of the 41st annual Design Automation Conference*. ACM. 2004, pp. 484–487. DOI: 10.1145/996566.996701.

[Sam04] S. Samadi, M. O. Ahmad, and M. Swamy. "Results on maximally flat fractional-delay systems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 51.11 (2004), pp. 2271–2286. DOI: 10.1109/TCSI.2004.836848.

[Sch01] A. Schuchert, R. Hasholzner, and P. Antoine. "A novel IQ imbalance compensation scheme for the reception of OFDM signals". In: *IEEE Transactions on Consumer Electronics* 47.3 (2001), pp. 313–318. DOI: 10.1109/30.964115.

[Sha07] T. Shahana, R. K. James, B. R. Jose, K. P. Jacob, and S. Sasi. "Polyphase implementation of non-recursive comb decimators for sigma-delta A/D converters". In: *IEEE Conference on Electron Devices and Solid-State Circuits, 2007. EDSSC 2007.* IEEE. 2007, pp. 825–828. DOI: 10.1109/EDSSC.2007.4450253.

[Sha49] C. E. Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (Jan. 1949), pp. 10–21. DOI: 10.1109/JRPROC.1949.232969.

[Sin04] A. Sinha, M. Bhardwaj, and K. Chadha. "Unified digital front end for IEEE 802.11 g WLAN system". US20040152418A1. Aug. 2004.

[Soo11] P. Soontornwong, S. Chivapreecha, and C. Pradabpet. "A Cubic Hermite variable fractional delay filter". In: *International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), 2011*. IEEE. Dec. 2011, pp. 1–4. DOI: 10.1109/ISPACS.2011.6146195.

[Sri77] A. Sripad and D. Snyder. "A necessary and sufficient condition for quantization errors to be uniform and white". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.5 (1977), pp. 442–448. DOI: 10.1109/TASSP.1977.1162977.

[Sta10] Z. Stamenković, K. Tittelbach-Helmrich, M. Krstić, J. Ibanez, V. Elvira, and I. Santamaria. "MAC and baseband hardware platforms for RF-MIMO WLAN". In: *Proceedings of Papers 5th European Conference on Circuits and Systems for Communications (ECCSC'10)*. IEEE Xplore ID: 5733849. Nov. 2010, pp. 26–33. DOI: N/A.

[Sto17] B. L. R. Stojkoska and K. V. Trivodaliev. "A review of Internet of Things for smart home: Challenges and solutions". In: *Journal of Cleaner Production* 140 (2017), pp. 1454–1464. DOI: 10.1016/j.jclepro.2016.10.006.

[Stu06] R. Stuhlberger, L. Maurer, G. Hueber, and A. Springer. "The Impact of RF-Impairments and Automatic Gain Control on UMTS-HSDPA-Throughput Performance". In: *IEEE Vehicular Technology Conference*. Sept. 2006, pp. 1–5. DOI: 10.1109/VTCF.2006.389.

[Sun95] W. Sung and K.-I. Kum. "Simulation-based word-length optimization method for fixed-point digital signal processing systems". In: *IEEE transactions on Signal Processing* 43.12 (1995), pp. 3087–3090. DOI: 10.1109/78.476465.

[Tan10] D. Tandur. "Digital compensation of front-end non-idealities in broadband communication systems". PhD thesis. Ph. D. dissertation - Dept. Elect. Eng. - Katholieke Univ. - Leuven - Belgium, Mar. 2010. URL: ftp://ftp.esat.kuleuven.be/sista/dtandur/report/10-68.pdf.

[Tar07] H. Tarn, K. Neilson, R. Uribe, and D. Hawke. *Designing Efficient Wireless Digital Up and Down Converters Leveraging CORE Generator and System Generator*. Application Note XAPP1018 [Online, accessed 30/10/2018]. Xilinx Inc. Oct. 2007. URL: https://www.xilinx.com/support/documentation/application_notes/xapp1018.pdf.

[Tas96] S. Tassart and P. Depalle. "Fractional delay lines using Lagrange interpolation". In: *Proceedings of the 1996 International Computer Music Conference*. HAL ID : hal-01105472. The International Computer Music Association. 1996, pp. 341–343. DOI: N/A.

[Tas97] S. Tassart and P. Depalle. "Analytical approximations of fractional delays: Lagrange interpolators and allpass filters". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP97*. Vol. 1. IEEE. 1997, pp. 455–458. DOI: 10.1109/ICASSP.1997.599673.

[Tex09a] Texas Instruments. *GC4016 MULTI-STANDARD QUAD DDC CHIP*. Data Manual SLWS133B [Online, accessed 30/10/2018]. July 2009. URL: https://datasheet.ciiva.com/2796/gc4016-2796594.pdf.

[Tex09b] Texas Instruments. *GC5325 Wideband Digital Predistortion Transmit Processor*. Data Manual SLWS215 [Online, accessed 30/10/2018]. Jan. 2009. URL: https://media.digikey.com/pdf/Data%20Sheets/Texas%20Instruments%20PDFs/GC5325.pdf.

[Tex15a] Texas Instruments. *66AK2L06 Multicore DSP+ARM KeyStone II System-on-Chip (SoC)*. Data Manual SPRS930 [Online, accessed 30/10/2018]. Apr. 2015. URL: http://www.ti.com/lit/ds/symlink/66ak2l06.pdf.

[Tex15b] Texas Instruments. *Digital Front End (DFE) User Guide for Keystone II Devices*. User's Guide SPRUHX8A [Online, accessed 30/10/2018]. Apr. 2015. URL: http://www.ti.com/lit/ug/spruhx8a/spruhx8a.pdf.

[Tho93] D. E. Thomas, J. K. Adams, and H. Schmit. "A model and methodology for hardware-software codesign". In: *IEEE Design Test of Computers* 10.3 (Sept. 1993), pp. 6–15. ISSN: 0740-7475. DOI: 10.1109/54.232468.

[Tou99] J.-M. Tourreilles. "Conception d'architectures pour traitement du signal en precision finie". PhD thesis. Université Rennes 1, 1999.

[Tse12] C.-C. Tseng and S.-L. Lee. "Design of fractional delay filter using Hermite interpolation method". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 59.7 (2012), pp. 1458–1471. DOI: 10.1109/TCSI.2011.2177136.

[Tsu05] K. M. Tsui, S.-C. Chan, and K. Tse. "Design of complex-valued variable FIR digital filters and its application to the realization of arbitrary sampling rate conversion for complex signals". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 52.7 (2005), pp. 424–428. DOI: 10.1109/TCSII.2005.850405.

[Tuc11] M. Tuchler and A. C. Singer. "Turbo Equalization: An Overview". In: *IEEE Transactions on Information Theory* 57.2 (Feb. 2011), pp. 920–952. ISSN: 0018-9448. DOI: 10.1109/TIT.2010.2096033.

[Uns99] M. Unser. "Splines: A perfect fit for signal and image processing". In: *IEEE Signal processing magazine* 16.6 (1999), pp. 22–38. DOI: 10.1109/79.799930.

[Val95] V. Valimaki. "A new filter implementation strategy for Lagrange interpolation". In: *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*. Vol. 1. IEEE. 1995, pp. 361–364. DOI: 10.1109/ISCAS.1995.521525.

[Ves00] J. Vesma, F. Lopez, T. Saramäki, and M. Renfors. "The effects of quantizing the fractional interval in interpolation filters". In: *Proceedings of the Nordic Signal Processing Symposium NORSIG2000*. 2000, pp. 215–218. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.4249&rep=rep1&type=pdf.

[Ves96] J. Vesma and T. Saramaki. "Interpolation filters with arbitrary frequency response for all-digital receivers". In: *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*. Vol. 2. IEEE. 1996, pp. 568–571. DOI: 10.1109/ISCAS.1996.541788.

[Ves97] J. Vesma and T. Saramaki. "Optimization and efficient implementation of FIR filters with adjustable fractional delay". In: *Proceedings of 1997 IEEE International Symposium on Circuits and Systems. Circuits and Systems in the Information Age ISCAS '97*. Vol. 4. IEEE. 1997, pp. 2256–2259. DOI: 10.1109/ISCAS.1997.612771.

[Vuc09] M. Vucic and M. Butorac. "All-digital high-dynamic automatic gain control". In: *2009 IEEE International Symposium on Circuits and Systems. ISCAS09*. IEEE. 2009, pp. 1032–1035. DOI: 10.1109/ISCAS.2009.5117935.

[Wei14] J. Wei. "How Wearables Intersect with the Cloud and the Internet of Things: Considerations for the developers of wearables." In: *IEEE Consumer Electronics Magazine* 3.3 (2014), pp. 53–56. DOI: 10.1109/MCE.2014.2317895.

[Wei91] M. Weiser. "The Computer for the 21st Century". In: *Scientific American* 265 (Sept. 1991), pp. 94–104. DOI: 10.1038/scientificamerican0991-94.

[Whi35] J. M. Whittaker. *Interpolatory function theory*. Vol. 33. The University Press, 1935.

[Wid56] B. Widrow. "A study of rough amplitude quantization by means of Nyquist sampling theory". In: *IRE Transactions on Circuit Theory* 3.4 (1956), pp. 266–276. DOI: 10.1109/TCT.1956.1086334.

[Wid61] B. Widrow. "Statistical analysis of amplitude-quantized sampled-data systems". In: *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry* 79.6 (1961), pp. 555–568. DOI: 10.1109/TAI.1961.6371702.

[Wid96] B. Widrow, I. Kollar, and M.-C. Liu. "Statistical theory of quantization". In: *IEEE Transactions on instrumentation and measurement* 45.2 (1996), pp. 353–361. DOI: 10.1109/19.492748.

[Wil97] M. Willems, V. Bürsgens, H. Keding, T. Grötker, and H. Meyr. "System level fixed-point design based on an interpolative approach". In: *Proceedings of the 34th annual Design Automation Conference*. ACM. 1997, pp. 293–298. DOI: 10.1109/DAC.1997.597160.

[Won91] P. W. Wong. "Quantization and roundoff noises in fixed-point FIR digital filters". In: *IEEE Transactions on Signal Processing* 39.7 (1991), pp. 1552–1563. DOI: 10.1109/78.134394.

[XFA17] X-FAB Semiconductor Foundries AG. *XH018 0.18 Micron Modular Analog Mixed HV Technology*. Process Family Datasheet [Online, accessed 03/07/2019]. 2017. URL: https://www.xfab.com/fileadmin/X-FAB/Download_Center/Technology/Datasheet/XH018_Datasheet.pdf.

[Xil02] Xilinx, Inc. *LogiCORE Digital Down Converter V1.0*. Product Specification [Online, accessed 30/10/2018]. Mar. 2002. URL: https://forums.xilinx.com/xlnx/attachments/xlnx/Virtex/24694/1/ddc_coregen.pdf.

[Xil12] Xilinx, Inc. *Virtex-6 FPGA Configurable Logic Block*. User Guide UG364 [Online, accessed 03/07/2019]. 2012. URL: https://www.xilinx.com/support/documentation/user_guides/ug364.pdf.

[Xil15] Xilinx, Inc. *Virtex-6 Family Overview*. Product Specification DS150 [Online, accessed 03/07/2019]. 2015. URL: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf.

[Ye15] F. Ye, Y. Qian, and R. Q. Hu. "Energy efficient self-sustaining wireless neighborhood area network design for smart grid". In: *IEEE Transactions on Smart Grid* 6.1 (Aug. 2015), pp. 220–229. DOI: 10.1109/TSG.2014.2344659.

[Zan14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. "Internet of things for smart cities". In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32. DOI: 10.1109/JIOT.2014.2306328.

[Zha09] L. Zhang, Y. Zhang, and W. Zhou. "A fast and flexible accuracy-guaranteed fractional bit-widths optimization approach". In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1517–1520. DOI: 10.1109/ISCAS.2009.5118056.

[Zöl08] U. Zölzer. *Digital audio signal processing*. ISBN: 978-0-470-68002-5. John Wiley & Sons, 2008.

List of Acronyms

3GPP 3rd Generation Partnership Project

4G 4th Generation Mobile Network

5G 5th Generation Mobile Network

AA Anti-Aliasing

ADC Analog to Digital Converter

AFC Automatic Frequency Control

AFE Analog Front-End

AGC Autmoatic Gain Control

AI Anti-Imaging

ALU Arithmetic Logic Unit

ARPANET Advanced Research Projects Agency Network

ASIC Application Specific Integrated Circuit

BB Baseband

BBU Baseband Unit

BD Backward Difference

BPSK Binary Phase-Shift Keying

BRAM Block Random Access Memory

BTS Base Transceiver Station

BW Bandwidth

CAD Computer Aided Design

CD Central Difference

CFR Crest Factor Reduction

CIC Cascaded Integrator Comb

CLB Configurable Logic Block

CSS Chirp Spread Spectrum

dB Decibel

DBPSK Differential Binary Phase-Shift Keying

DCOC Direct Current Offset Correction

DCR Direct Conversion Receiver

DDC Digital Down Conversion

DFE Digital Front-End

DFT Discrete Fourier Transform

DPD Digital Pre-Distortion

DR Dynamic Range

DSP Digital Signal Processor

DTFT Discrete Time Fourier Trasform

DUC Digital Up Conversion

ENOB Effective Number of Bits

EVM Error Vector Magnitude

FDMA Frequency Division Multiple Access

FEC Forward Error Correction

FF Flip-Flop

FHSS Frequency-Hopping Spread Spectrum

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

FSK Frequency-Shift Keying

FSM Finite State Machine

FT Fourier Transform

GFSK Gaussian Frequency-Shift Keying

GMSK Gaussian Minimum-Shift keying

GPP General Purpose Processor

GSM Global System for Mobile Communications

HDL Hardware Description Language

IDC International Data Corporation

IDTFT Inverse Discrete Time Fourier Transform

IF Intermediate Frequency

IFT Inverse Fourier Transform

IIR Infinite Impulse Response

IoT Internet of Things

LLN Law of Large Numbers

LNA Low Noise Amplifier

LO Local Oscillator

LO-PLL Local Oscillator Phase-Locked Loop

LP-WAN Low Power Wide Area Network

LPF Low Pass Filter

LSB Least Significant Bit

LTE Long Term Evolution

LTI Linear Time Invariant

LUT Look-Up Table

MMSE Minimum Mean Squared Error

MSB Most Significant Bit

NBDF Newton Backward Difference Formula

NCO Numerically Controlled Oscillator

OFDMA Orthogonal Frequency Division Multiple Access

OSI Open Systems Interconnection

PA Power Amplifier

PAPR Peak-to-Average Power Ratio

PBF Polynomial Based Filter

PG Processing Gain

ppm Parts per million

PSD Power Spectral Density

QAM Quadrature Amplitude Modulation

QoS Quality of Service

QPSK Quadrature Phase-Shift Keying

RAN Radio Access Network

RFID Radio Frequency Identification

RPMA Random Phase Multiple Access

RRH Remote Radio Head

RRU Remote Radio Unit

RTL Registor Transfer Level

Rx Receiver

SC-FDMA Single-Carrier Frequency Division Multiple Access

SDR Software Defined Radio

SNR Signal to Noise Ratio

SQNR Signal to Quantization Noise Ratio

SRC Sample Rate Conversion

SSNR Spectral Signal to Noise Ratio

TDMA Time Division Multiple Access

TFS Transposed Farrow Structure

Tx Transmitter

UFD Up-sampling Filtering Down-Samlping

UNB Ultra Narrow Band

USRP Universal Software Radio Peripheral

V-FDF Variable Fractional Delay Filter

VCO Voltage Controlled Oscillator

VDF Variable Digital Filter

VGA Variable Gain Amplifier

W-LAN Wireless Local Area Network

W-NAN Wireless Neighborhood Area Network

W-PAN Wireless Personal Area Network

W-WAN Wireless Wide Area Network

ZF Zero Forcing

ZOH Zero Order Hold

Titre : Conception d'un Front-End Numérique Générique pour l'Internet des Objets

Mots clés : Front-end numérique, changement de rythme d'échantillonnage, quantification, hardware

Résumé : Le nombre de technologies et de normes de communications sans fil est en augmentation constante afin de fournir des solutions de communication à distance pour les différents besoins technologiques actuels et à venir. Ceci est particulièrement le cas de l'Internet des objets (IoT), où déjà de nombreuses solutions sont disponibles, et de nombreuses autres sont attendues. Pour un déploiement efficace du réseau IoT, l'interopérabilité entre les différentes solutions est essentielle pour éviter la fragmentation de ce réseau, ce qui augmente ses coûts d'installation et d'exploitation et complique sa gestion. L'interopérabilité sur le plan physique est fournie par des modems multistandards prenant en charge le plus grand nombre de normes avec un coût de mise en œuvre minimal. Ces modems sont possibles grâce au front end numérique (DFE), qui offre une interface radio flexible capable de traiter une large gamme de signaux.

Cette thèse développe tout d'abord deux architectures génériques de DFE pour la transmission et la réception, pouvant être facilement adaptées aux différentes normes IoT. Ces architectures mettent en évidence le rôle principal du changement de rythme (SRC) dans le DFE et l'importance de l'optimisation de la mise en œuvre de cette fonction. Cette optimisation est ensuite réalisée grâce à une étude approfondie des fonctions SRC, et au développement de nouvelles structures plus efficaces en termes de complexité de mise en œuvre et de consommation, pour des performances égales ou supérieures. La dernière partie de la thèse concerne l'optimisation de la mise en œuvre matérielle du DFE, réalisée par le développement d'une méthode de quantification optimale qui minimise l'utilisation de ressources matérielles tout en garantissant un certain niveau de performance. Les résultats obtenus sont enfin mis en valeurs en comparant différentes stratégies de mise en œuvre sur des cibles FPGA et ASIC.

Title: Design of a Generic Digital Front-End for the Internet of Things

Keywords: Digital front-end, sample rate conversion, quantization, hardware

Abstract: The number of wireless communication technologies and standards is constantly increasing to provide communication solutions for today's technological needs. This is particularly relevant in the domain of the Internet of Things (IoT), where many standards are available, and many others are expected. To efficiently deploy the IoT network, the interoperability between the different solutions is critical in order to avoid the fragmentation of this network, which increases its installation and operation costs, and complicates its management. Interoperability on the physical level is achieved through multi-standard modems that support the largest number of standards with minimal added implementation costs. These modems are made possible through the digital front-end (DFE), that offers a flexible radio front-end able of processing a wide range of signal types.

This thesis first develops a generic architecture of both transmission and reception DFEs, which can be easily adapted to support different IoT standards. These architectures highlight the main role of sample rate conversion (SRC) in the DFE, and the importance of optimizing the SRC implementation. This optimization is then achieved through an in-depth study of the SRC functions, and the development of new structures of improved efficiency in terms of implementation complexity and power consumption, while offering equivalent or improved performance. The final part of the thesis addresses the optimization of the DFE hardware implementation, which is achieved through developing an optimal quantization method that minimizes the use of hardware resources while guaranteeing a given performance constraint. The obtained results are finally highlighted through implementing and comparing different implementation strategies on both FPGA and ASIC targets.