Université du Québec
Institut national de la recherche scientifique
Centre Énergie Matériaux Télécommunications

PROTOTYPAGE RAPIDE ET VALIDATION SUR UNE PLATEFORME SDR D'UN ESTIMATEUR HYBRIDE À MAXIMUM DE VRAISEMBLANCE DU RAPPORT SIGNAL SUR BRUIT

Par

Zied Ben Gamra

Mémoire présenté pour l'obtention du grade de
Maître es Sciences, M.Sc.
en télécommunications

Jury d'évaluation

Examinateur interne	Prof. Nahi Kandil LRTCS
Examinateur externe	Prof. Jean-François Frigon École Polytechnique de Montréal
Directeur de recherche	Prof. Sofiène Affes INRS-ÉMT
Co-directeur de recherche	Dr. Faouzi Bellili Associé de recherche INRS-ÉMT
Co-directeur de recherche	Dr. Abdelaziz Samet Associé de recherche INRS-ÉMT

Remerciements

Ce travail a été élaboré au sein du Wireless Lab du centre Énergie, Matériaux et Télécommunications de l'Institut National de la Recherche Scientifique comme une application de la méthode du prototypage rapide aux algorithmes de télécommunications sans fil.

Tout d'abord je tiens à exprimer ma profonde gratitude à mon directeur de recherche le Professeur Sofiène Affes pour ses prodigieux conseils et directives qui m'ont guidé tout au long de l'accomplissement de ces travaux de recherche. Je le remercie pour son aide précieuse et ses critiques constructives qui m'ont permis d'avancer à chaque difficulté rencontrée.

Ensuite, je tiens à remercier M. Abdelaziz Samet qui m'a soutenu et encouragé pendant toute la durée de mes travaux. Je lui suis reconnaissant pour son écoute, ses conseils et ses suggestions.

Je tiens également à remercier toute l'équipe du Wireless Lab pour leur aide et spécialement M. Faouzi Bellili pour sa disponibilité, ses explications pertinentes et sa volonté d'aider.

Mes remerciements vont également aux membres du Jury pour l'attention qu'ils ont portée à ce mémoire.

Finalement, je dédie ce mémoire à mes chers parents pour leur patience, encouragement et soutien, à mes sœurs pour avoir toujours cru en moi ainsi qu'à mes amis et ceux que j'aime.

Résumé

Les standards de télécommunications sans fil ne cessent de changer et d'évoluer rapidement. Ceci présente un défi pour les chercheurs et les industriels qui doivent s'y adapter au fur et à mesure. La nécessité d'avoir une méthode et un matériel qui permettent de faire face à ce changement constant a popularisé l'utilisation et du concept du prototypage rapide et des plateformes de radio logicielle (SDR) à base de FPGA. Évidemment, plusieurs approches et outils basés sur ce concept ont vu le jour en ce qui concerne la programmation des radios logicielles. Toutes ces approches se caractérisent par le gain en temps de développement qu'elles offrent ainsi que leur facilité d'application. Les outils proposés ne nécessitent plus une connaissance approfondie ni des langages de programmation du matériel ni du matériel ciblé.

Le travail présenté dans ce mémoire est une application de la méthode de prototypage sur une plateforme SDR. Un estimateur du rapport signal sur bruit (SNR), développé au sein de notre équipe du Wireless Lab dirigé par le professeur Sofiène Affes, a été implémenté, testé et validé sur une plateforme radio logicielle.

L'approche de conception choisie pour cela est celle basée sur un modèle (Model-Based Design ou MBD). Cette approche ne nécessite aucune ligne de code vu qu'elle se base sur les outils Matlab/Simulink et Xilinx System Generator (XSG) qui présentent une interface graphique et une méthode de «Drag-and-drop». Ces outils logiciels permettent de créer une architecture de l'estimateur qui est, par la suite, traduite en langage matériel (HDL) et implémentée sur la plateforme SDR ciblée. La plateforme radio logicielle PicoSDR2X2 de Nutaq et l'émulateur de canaux EB Propsim d'Anite constituent notre environnement expérimental. Les résultats obtenus ont été comparés à ceux obtenus avec MATLAB.

Mots-clés Prototypage rapide; télécommunications sans fil; radio logicielle; FPGA; technique de conception basée sur un modèle; estimateur du rapport signal sur bruit; MATLAB; Simulink; Xilinx System Generator; PicoSDR2X2 de Nutaq; émulateur de canal EB Propsim d'Anite.

Abstract

Telecommunication standards are changing and evolving rapidly. This is a challenge for researchers and industrials who must adapt constantly. The need for methods and hardware that allow to face these changes made the use of rapid prototyping concept and FPGA-based software defined radios (SDR) popular. Predictably, many approaches and tools based on this concept and dealing with SDR programming have emerged. All these approaches have in common the fact that they are development-time-saving and easy-to-use. The proposed tools do not require any deep knowledge neither of hardware-programming languages nor of the targeted hardware itself.

The work presented in this thesis is an application of rapid prototyping method on an SDR platform. A signal-to-noise ratio (SNR) estimator, developed by our Wireless Lab research group led by Professor Sofiène Affes, was implemented, tested and validated on an SDR platform.

The approach selected to achieve this work is model-based. It requires no coding since it is based on Matlab/Simulink and Xilinx Sytsem Generator (XSG) tools that present a graphical user interface and a drag-and-drop method. Those software tools allow the creation of a design architecture for the estimator which will be translated into hardware description language (HDL) and implemented on the targeted SDR. Nutaq's PicoSDR2X2 platform and Anite's EB Propsim channel emulator define our experimental environment. The obtained results were compared to those of MATLAB.

Keywords Rapid prototyping; wireless telecommunications; software defined radio; FPGA; model-based design technique; Signal-to-Noise Ratio estimator; MATLAB; Simulink; Xilinx System Generator; Nutaq's PicoSDR2X2; Anite's EB Propsim channel emulator.

Table des matières

Remerciements iii

Résumé v

Abstract vii

Table des matières ix

Liste des figures xi

Liste des tableaux xiii

Liste des abréviations xv

1 Introduction générale 1

2 Prototypage rapide sur plateforme de radio logicielle: Approche de conception basée sur un modèle 3
- 2.1 Introduction . . . 3
- 2.2 La radio logicielle . . . 4
- 2.3 Outils de programmation de la radio logicielle . . . 4
  - 2.3.1 Outils de haut niveau de synthèse . . . 6
  - 2.3.2 Outils à interface graphique . . . 6
- 2.4 Méthodologie de conception et d'implémentation . . . 7
- 2.5 Conclusion . . . 9

3 Présentation de l'estimateur du rapport signal à bruit 11
- 3.1 Introduction . . . 11
- 3.2 Choix de l'estimateur . . . 11
- 3.3 Description de l'estimateur . . . 13
  - 3.3.1 Modèle du système . . . 13
  - 3.3.2 Estimateur DA . . . 14
  - 3.3.3 Estimateur NDA . . . 16
  - 3.3.4 Estimateur hybride: Initialisation de l'algorithme EM . . . 18
- 3.4 Conclusion . . . 19

4 Conception et implémentation de l'estimateur du SNR 21
- 4.1 Introduction . . . 21
- 4.2 Paramétrage de l'estimateur . . . 22

4.3 Conception et implémentation 22
4.3.1 Chemin des données 23
4.3.2 Chemin de contrôle 27
4.3.3 Blocs de modélisation personnalisés 28
4.4 Génération du bitstream 32
4.5 Conclusion 33

5 Description de l'environnement de travail et validation des performances 35
5.1 Introduction 35
5.2 Environnement de travail 35
5.2.1 PicoSDR2X2 de Nutaq 36
5.2.2 Émulateur de canaux EB Propsim d'Anite 39
5.3 Validation des performances du PSM sur la plateforme PicoSDR2X2 39
5.3.1 Validation de l'estimation du SNR 40
5.3.2 Validation de la démodulation des symboles QPSK 43
5.3.3 Identification du canal de transmission 46
5.4 Conclusion 54

6 Conclusion générale 55

Références 57

Liste des figures

2.1 Les différentes approches de programmation de FPGA. 5
2.2 Les étapes d'implémentation. 8

3.1 Comparaison de l'estimateur choisi avec WGM-SIMO pour différents nombres d'antennes: (a) $N_r = 2$, (b) $N_r = 4$, et (c) $N_r = 8$, avec $F_D T_s = 7 \times 10^{-3}$, $N = 112$, $\bar{N}_{DA} = 112$, $\bar{N}_{NDA} = 56$ et $L = 4$, QPSK. 13

4.1 Diagramme en blocs du design global. 23
4.2 Implémentation dans l'environnement MATLAB/Simulink et Xilinx du design global. 23
4.3 Diagramme en blocs de l'estimateur DA. 24
4.4 Diagramme en blocs de l'estimateur NDA. 26
4.5 Chemin de contrôle du design global. 27
4.6 Bloc de multiplication de matrices complexes. 29
4.7 Bloc de multiplication complexe et ses composants. 30
4.8 Bloc de calcul du module au carré et ses composants. 30
4.9 Bloc de sommation et ses composants. 31
4.10 Génération du bitstream. 32

5.1 Plateforme PicoSDR2X2 de Nutaq. 36
5.2 Architecture interne de la plateforme PicoSDR2X2 de Nutaq. 36
5.3 Carte FPGA Virtex-6. 37
5.4 Carte AMC Perseus601x. 37
5.5 Carte radio FMC 420m. 37
5.6 Agencement des outils de Xilinx et des outils de Nutaq. 38
5.7 Émulateur de canaux EB Propsim d'Anite. 39
5.8 PCM de la validation de l'estimation du SNR. 40
5.9 NMSE de l'algorithme sur MATLAB de l'estimateur NDA comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK. 41
5.10 NMSE de l'algorithme sur MATLAB de l'estimateur NDA comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK. 42
5.11 NMSE de l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK. 42
5.12 NMSE de l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK. 43

5.13 PCM de la validation de la démodulation des symboles QPSK. 44
5.14 BER de l'algorithme sur MATLAB de l'estimateur hybride comparé à celui du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK. 45
5.15 BER de l'algorithme sur MATLAB de l'estimateur hybride comparé à celui du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK. 45
5.16 PCM de la validation de du canal de transmission créé avec MATLAB. 46
5.17 Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 47
5.18 Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 47
5.19 Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 48
5.20 Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 48
5.21 Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 49
5.22 Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 49
5.23 PCM de l'identification du canal de transmission émulé avec EB Propsim. 50
5.24 Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 51
5.25 Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 51
5.26 Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 52
5.27 Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 52
5.28 Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 53
5.29 Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK. 53

Liste des tableaux

4.1 Configuration des paramètres de l'estimateur 22

Liste des abréviations

AMC: Advanced Mezzanine Card
BER: Bit Error Rate
BPSK: Binary Phase-Shift Keying
BSDK: Board Software Design Kit
CAN: Convertisseur Analogique Numérique
CFO: Carrier Frequency Offset
CNA: Convertisseur Numérique Analogique
DA: Data-Aided
DSP: Digital Signal Processing
EM: Expectation-Maximization
FDD: Frequency Division Duplex
FMC: FPGA Mezzanine Card
FPGA: Field-Programmable Gate Array
GUI: Graphical User Interface
HDL: Hardware Description Language
HLS: High Level Synthesis
LS: Least Square
LTE: Long Term Evolution
LTE-A: Long Term Evolution-Advanced
LTE-B: Long Term Evolution-Beyond
LUT: Look-Up Table
MBD: Model-Based Design
MBDK: Model-Based Design Kit
MIMO: Multiple Inputs Multiple Outputs
ML: Maximum Likelihood
MSA: Machine Séquentielle Algorithmique
NDA: Non Data-Aided
NMSE: Normalized Mean Square Error
OFDM: Orthogonal Frequency-Division Multiplexing
PCM: Plateform Control Model
PIM: Plateform Independent Model
PSM: Plateform Specific Model
QPSK: Quadrature Phase-Shift Keying
RF: Radio Frequency
ROM: Read Only Memory
RTL: Register Transfer Level
SDR: Software Defined Radio
SIMO: Single Input Multiple Outputs
SNR: Signal to Noise Ratio

TDD: Time Division Duplex
UML: Unified Modeling Language
VHDL: Verilog Hardware Description Language

Chapitre 1

Introduction générale

Le prototypage rapide est une méthode de conception et de production relativement récente. Grâce à ses multiples avantages, elle est utilisée dans différents secteurs allant de la conception de simples produits de consommation à la conception de pièces complexes de l'industrie lourde. Cette technique consiste en la création, d'une manière rapide et optimisée, de prototypes du produit désiré. Ces derniers sont testés en temps réel afin de détecter d'éventuelles erreurs de conception ou d'améliorer le produit final. Ainsi, elle suit une stratégie d'économie de temps et de coûts.

Du fait de son aspect évolutif et le changement constant de ses standards, le secteur des télécommunications a su adapter la technique du prototypage rapide à ses besoins. La raison principale étant le fait que cette méthode permet aux ingénieurs de vérifier et d'optimiser leurs designs, de mesurer l'impact de la variation des paramètres, d'essayer différents scénarios et cela tôt dans le processus de développement et en temps réel grâce au co-design et à la co-simulation [1].

C'est à cet effet que le concept de la radio logicielle (SDR) fut introduit au milieu des années 80 et est devenu largement répandu et nécessaire pour la fabrication de la plupart des appareils sans fil [2]. Ceci est principalement dû à la haute malléabilité et adaptabilité de cette technologie face à l'évolution du secteur et son adéquation avec la technique du prototypage rapide. Le domaine de la recherche scientifique en télécommunications a également su tirer avantage de la radio logicielle dans le but d'établir un schéma de recherche plus global qui va de la conception théorique à l'implémentation pratique. Ce schéma permet une validation de la théorie dans des conditions réelles,

ce qui offre un prototype fini. Ainsi, on assiste à un rapprochement bénéfique entre le domaine de la recherche et le domaine de l'industrie.

Le présent mémoire s'insère dans cette perspective. Nous avons travaillé sur l'implémentation et la validation d'un nouvel estimateur du rapport signal à bruit (SNR) [26] développé au sein du Wireless Lab. Cet estimateur de SNR se base sur le principe du maximum de vraisemblance (ML) et offre une architecture intégrée du système qui inclut la démodulation et la caractérisation du canal de transmission pour des canaux variables dans le temps. Il est constitué de l'association de deux estimateurs indépendants, ce qui lui confère une haute performance. Son degré de complexité et son innovation en font un estimateur idéal pour l'implémentation et la validation sur une radio logicielle en conditions réelles. Pour cela, l'approche de programmation basée sur un modèle (MBD) a été utilisée étant donné les avantages qu'elle propose comparée aux méthodes classiques.

Les différentes étapes de ce projet sont détaillées dans ce mémoire comme suit : Au chapitre 2, nous présentons la technologie de la radio logicielle, nous comparons l'approche MBD ainsi que ses outils aux différentes approches existantes et nous détaillons la méthodologie suivie pour accomplir ce travail. Au chapitre 3, nous justifions le choix de l'estimateur et nous détaillons sa théorie. Au chapitre 4, nous décrivons la phase de conception et d'implémentation de l'algorithme de l'estimateur en mettant l'accent sur les avantages de l'approche suivie. Au chapitre 5, nous présentons notre environnement expérimental ainsi que les résultats de la validation des différentes fonctions de l'estimateur sur la radio logicielle. Au dernier chapitre, nous tirons une conclusion générale sur le travail présenté et introduisons les futures perspectives de ce projet.

Chapitre 2

Prototypage rapide sur plateforme de radio logicielle: Approche de conception basée sur un modèle

2.1 Introduction

Le changement rapide et constant des systèmes de télécommunications et le besoin d'être à la fine pointe de la technologie a mené à une augmentation dans l'utilisation de la méthode du prototypage rapide. Cela peut être constaté à travers la multitude d'applications qui utilisent cette technique. Parmi ces projets, on trouve ceux qui traitent des schémas de modulation : BPSK [3], QPSK [4]. D'autres, plus élaborés, ont été dédiés à l'implémentation de systèmes de communication numériques et à l'architecture des formes d'ondes [5] ou à l'implémentation d'une couche physique complète du standard IEEE 802.15.4 (ZigBee) [6]. Ces projets ont été implémentés sur des plateformes de radio logicielle en utilisant les outils spécifiques disponibles.

Dans ce qui suit, nous introduisons le concept de la radio logicielle du côté matériel ainsi que du côté logiciel. Nous donnons un aperçu des approches et outils de programmation existants en mettant l'accent sur l'approche basée sur un modèle. A la fin nous présentons la méthodologie que nous avons suivie pour accomplir ce travail.

2.2 La radio logicielle

La radio logicielle (SDR) est un système sans fil reconfigurable qui peut supporter plusieurs standards de communication [7-8]. Il permet la reprogrammation du système (FPGA, DSP, carte radio, etc.) en utilisant uniquement des commandes logicielles [9]. Ainsi, dans une radio logicielle, la plupart des fonctionnalités de réception et de traitement seront définies et modifiées à travers des programmes logiciels écrits par l'utilisateur [3-10].

Le concept de radio logicielle a été proposé dans le but d'éviter les mises à jour coûteuses du matériel ou le remplacement d'appareils de communication dus aux changements rapides des standards et des protocoles des communications sans fil [11].

Contrairement aux approches traditionnelles, la radio logicielle ne nécessite aucune connaissance antérieure de langages de description du matériel (VHDL, Verilog, etc.) vu qu'elle peut être programmée avec des logiciels d'un haut niveau d'abstraction. Ainsi, elle constitue une très bonne alternative d'économie de temps, en adéquation avec le prototypage rapide.

Les plateformes matérielles sont généralement composées d'un modèle de DSP frontal, où résident les différents modules de contrôle de la conversion des données ainsi que les modules RF, et d'un modèle interne, où les données sont traitées [3]. Il existe plusieurs plateformes qui sont utilisées principalement dans des projets de recherche. Par exemple, la radio logicielle Small Form Factor de Lyrtech, qui est composée d'un module RF, d'un module de conversion des données et d'un module de traitement des données, a été utilisée pour implémenter un schéma de modulation BPSK [3]. Dans une autre étude [5], la plateforme URSP a été utilisée conjointement avec un General Purpose Processor sur ordinateur pour implémenter un système de communication numérique. Une autre plateforme développée par Lyrtech nommée SIGNALWAVe a été utilisée pour implémenter un schéma de modulation QPSK [4]. Un dernier exemple est celui de la radio logicielle de Nutaq Perseus 6010 qui a été utilisée pour l'implémentation de la couche physique ZigBee [6].

2.3 Outils de programmation de la radio logicielle

Le processus de programmation des plateformes à base de FPGA telles que la radio logicielle (SDR), qui va de la conception à partir des spécifications algorithmiques jusqu'à l'implémentation

sur machine du bitstream, a vu émerger différentes approches. Ces approches ont toutes pour but de palier la complexité croissante de programmation du matériel moderne et de raccourcir la durée de vie de ce processus.

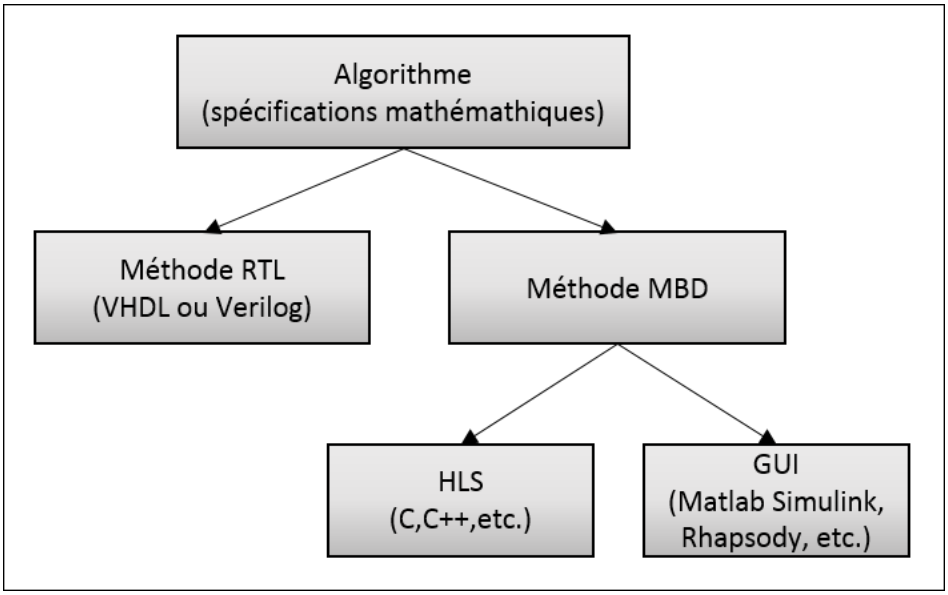


Figure 2.1 – Les différentes approches de programmation de FPGA.

L'approche basée sur un modèle (MBD) a su faire ses preuves comparée à celle traditionnelle basée sur la conception «Register Transfer Level » (RTL). Cette dernière nécessite une description détaillée des circuits (opérations logiques, signaux, etc.) et une traduction manuelle vers un des langages de description du matériel (HDL) tels que Verilog ou VHDL et de ce fait une connaissance du matériel ciblé. Alors que l'approche MBD reste une approche plus générale, plus accessible et plus rapide. En ces termes, elle s'insère plus dans le cadre du prototypage rapide. En effet, la MBD a pour caractéristique principale le fait qu'elle permet de séparer le «Ce que l'on veut faire» du «Comment arriver à le faire». En d'autres termes, elle permet de modeler le design sans avoir à se soucier des contraintes du matériel sur lequel ce design sera implémenté ou du langage avec lequel il sera décrit. Ainsi, elle améliore l'interopérabilité entre plateformes, standards ou langages [12].

Principalement, dans l'approche MBD, on rencontre deux types de modèles. D'abord, le modèle indépendant de la plateforme (PIM) qui, comme son nom l'indique, est développé sans prise en compte des contraintes matérielles et reste valide pour l'implémentation sur n'importe quelle plateforme. Ensuite, le modèle spécifique de la plateforme (PSM) est obtenu à partir du PIM et cible une plateforme et une technologie bien précises [12].

En ce qui concerne le développement du PIM, il existe deux méthodes différentes. La première se base sur des outils de haut niveau de synthèse (HLS) et la deuxième se base sur une interface graphique (GUI).

2.3.1 Outils de haut niveau de synthèse

L'utilisation des outils HLS pour la programmation de systèmes matériels à base de FPGA consiste à décrire le design avec un langage de haut niveau de spécification au lieu de passer par la méthode classique RTL. Cette alternative permet d'élever le niveau d'abstraction du design et ainsi contrôler la complexité et améliorer la productivité. Comme cette étude du NEC [13] le démontre, coder avec la méthode HLS réduit de 7 à 10 fois le nombre de lignes de code nécessaires comparée à la méthode RTL. Aussi, permet-elle de réduire de 11-31% l'utilisation des ressources FPGA [14].

La grande majorité des outils HLS utilisent des langages C tels que C, C++, SystemC [catapult] ou des langages C avec des extensions orientées vers le matériel tels que HardwareC [15], SpecC [16], Handel-C [17]. Quoique, il existe des outils qui utilisent d'autres langages tels que BlueSpec [18], Esterel [19], et MATLAB [20]. Parmi les applications de la méthode HLS dans le domaine des télécommunications, on trouve des systèmes sans fil 3G/4G [21], [22].

2.3.2 Outils à interface graphique

Les outils de programmation qui présentent une interface graphique (GUI) se basent soit sur le langage de modélisation unifié (UML) soit sur des librairies de blocs prédéfinis. Dans les deux cas, ils ne nécessitent pas de lignes de code tel que dans les outils HLS. Cela rend cette méthode encore plus accessible et moins sujette aux erreurs de codage. Dans la première catégorie on trouve, à titre d'exemple, Real Time Studio d'Artisan [12], Rhapsody d'I-logix [23]. Dans la deuxième, on retrouve le plus connu des outils de design rapide et de simulation à interface graphique, à savoir MATLAB Simulink [24].

Notre choix s'est porté sur ce dernier tant l'environnement de développement qu'il présente est accessible, varié, rapide et compatible avec différentes applications. Plus en détails, MATLAB Simulink combiné avec Xilinx System Generator représente un outil puissant de conception, test et implémentation de designs sur les systèmes embarqués. En effet, cet outil offre la possibilité d'utiliser

le langage de haut niveau d'abstraction de MATLAB et les blocks graphiques de Simulink et Xilinx simultanément. Il se base sur la méthode Drag-and-Drop pour créer une architecture hiérarchique du système. La simulation reste possible tout le long du cycle de développement, ce qui a pour avantage de permettre la détection d'erreurs et la vérification parallèle des résultats.

System Generator [25], qui fait partie de la librairie Xilinx, est un outil qui permet de traduire automatiquement un PIM construit en utilisant l'approche MBD GUI, en un PSM décrit en langage de bas niveau tel que VHDL en ciblant un matériel bien précis. Il génère aussi l'exécutable (bitstream) qui sera implémenté directement sur la plateforme choisie. Il offre d'autres options telles que l'analyse des contraintes temporelles, l'analyse des ressources nécessaires, l'analyse de la consommation de puissance, etc.

2.4 Méthodologie de conception et d'implémentation

Le passage des spécifications algorithmiques au test de l'exécutable en temps réel sur la plateforme se fait à travers une suite d'étapes qui présentent, chacune, un lot de défis bien spécifiques. Ces étapes sont illustrées dans la Figure 2.2.

- D'abord et comme au début de chaque conception, on se retrouve face aux spécifications et exigences de l'algorithme à implémenter. Il est important de bien comprendre ce dernier afin d'éviter les erreurs de conception à un stade avancé du processus de développement. Une fois les résultats de l'algorithme vérifiés sur MATLAB, il faut adapter le code utilisé afin de pouvoir comparer, graduellement, les résultats initiaux avec ceux du PIM. En effet, la structure du PIM est combinatoire, basée sur une construction par blocs et sous-systèmes hiérarchiques, et non séquentielle. De ce fait, il faut bien identifier les entrées/sorties de chaque bloc dans le code pour pouvoir les comparer. De plus, il faut adapter les opérations mathématiques complexes telles le calcul matriciel, adapter les fonctions MATLAB, etc.

- Ensuite, vient l'étape où l'on crée le PIM dans l'environnement GUI Simulink en utilisant les blocs et composants disponibles dans les librairies Xilinx. Le défi de cette étape étant de savoir allouer le nombre de bits nécessaires à chaque signal vu qu'on passe d'une représentation en virgule flottante à une représentation en virgule fixe. Allouer un nombre insuffisant de bits fausse les résultats, en allouer trop est une perte de ressources. Il faut bien veiller à

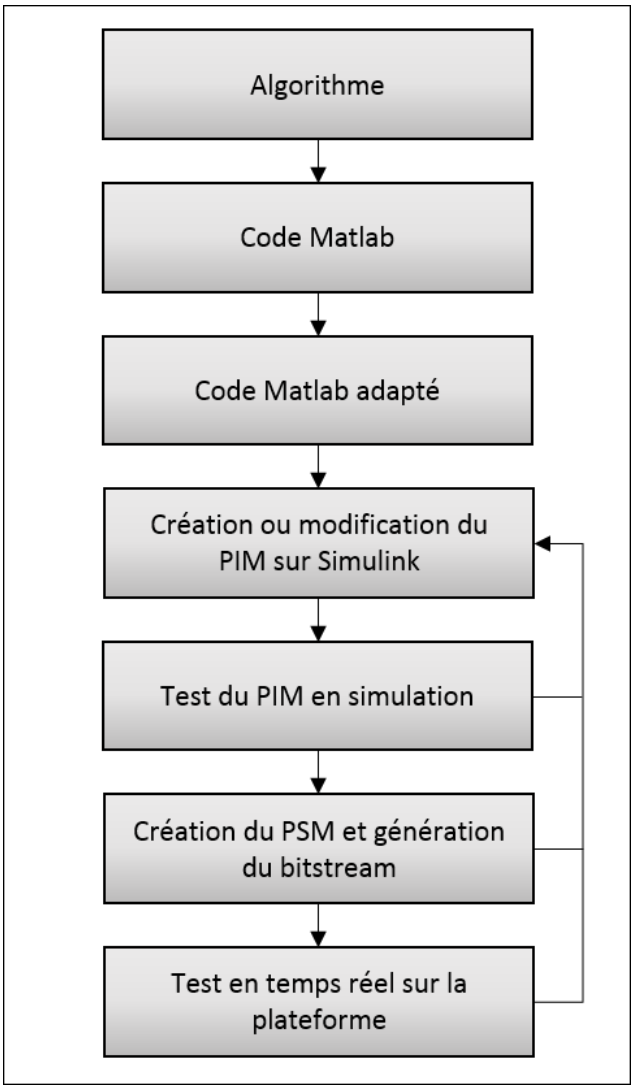


Figure 2.2 – Les étapes d'implémentation.

vérifier la concordance des résultats au fur et à mesure de l'avancement. Ceci se traduit par des simulations fréquentes. Si les tests en simulations ne donnent pas les résultats attendus, il faut remodifier le design et corriger les erreurs. Il faut s'attendre à une perte en précision due à la représentation en virgule fixe qu'il faut minimiser autant que possible.

- Une fois le PIM terminé et testé en simulations en donnant les mêmes résultats que le code MATLAB, on passe à l'étape de la création du PSM et de la génération du bitstream. Les problèmes rencontrés ici sont de deux types. Le premier étant de dépasser les ressources de la plateforme ciblée. Le deuxième est le problème classique du chemin critique. Ce problème se caractérise par le fait que le temps nécessaire pour un signal d'aller d'une mémoire à une

autre est plus long qu'un cycle d'horloge. Dans ce cas, il faut réduire ce temps en modifiant le PIM en insérant des bascules (représentées par des registres).

- La dernière étape consiste à tester en temps réel l'exécutable sur la plateforme et à vérifier qu'on obtient bel et bien les résultats attendus. Des problèmes d'initialisation (« reset ») de certains composants peuvent se produire. Il faut revenir à l'étape de la modification du PIM afin de les corriger.

2.5 Conclusion

Dans ce chapitre, nous avons défini le concept d'une radio logicielle et on a montré son importance pour les télécommunications de nos jours. Nous avons présenté les approches de programmation et avons justifié le choix de la technique MBD. Enfin, nous avons énuméré les étapes de l'implémentation de l'estimateur sur la plateforme SDR. Dans le prochain chapitre, nous détaillerons la théorie de cet estimateur.

Chapitre 3

Présentation de l'estimateur du rapport signal à bruit

3.1 Introduction

Estimer les paramètres du canal de communication sans fil est essentiel pour assurer la qualité de service requise. La littérature regorge d'estimateurs bien spécifiques à chaque paramètre ou caractéristique. Cependant, l'estimateur que nous présentons [26], quoique s'attaquant principalement à l'estimation du rapport signal sur bruit (SNR), peut être considéré comme étant une architecture intégrée du système. En effet, ajouté au fait qu'il estime le SNR, il démodule les symboles reçus et estime le canal simultanément. Ceci implique, indirectement, la possiblilité d'estimer d'autres paramètres tels que l'étalement Doppler ou le décalage en fréquence de porteuse (CFO).

Dans ce chapitre, nous présentons l'estimateur du SNR sur lequel nous avons choisi de travailler en justifiant ce choix. Nous exposerons ensuite son concept aussi bien que ses différentes versions.

3.2 Choix de l'estimateur

Les estimateurs classiques ne donnent pas de résultats satisfaisants lorsqu'ils sont utilisés avec les systèmes multi-antennes actuels ou futurs comme Long Term Evolution (LTE), Long Term

Evolution-Advanced (LTE-A) et Beyond (LTE-B) qui sont supposés assurer des communications de bonne qualité à de très hautes vitesses atteignant 500 Km/h [27]. Ceci est dû principalement au fait qu'ils considèrent que les canaux de communication varient lentement ou sont quasi-constants dans le temps, ce qui n'est pas le cas en réalité.

L'estimateur sur lequel nous avons choisi de travailler offre une nouvelle technique d'estimation à maximum de vraisemblance (ML) et cela pour les canaux à variations rapides dans le temps et à une seule entrée et plusieurs sorties (SIMO). En effet, au lieu de considérer le canal comme étant constant sur toute la période d'observation, ses variations sont localement mesurées avec une expansion polynomiale dans le temps. Les seuls travaux qui se sont attaqués à l'estimation du SNR sur des canaux variables dans le temps et une configuration SIMO se sont basés sur l'approche least-square (LS) [28], [29].

Cet estimateur est développé sous trois versions. La première est classée parmi les techniques d'estimation assistées par des données (DA) et est développée sous forme d'une expression analytique. Une connaissance de quelques-uns des symboles envoyés est nécessaire. Ces symboles sont appelés symboles pilotes. La deuxième est complètement non assistée par des données (NDA). Ceci signifie qu'aucune connaissance de la séquence envoyée n'est requise. Dans cette version, à cause de la non-linéarité de la fonction de vraisemblance de l'estimateur, une technique itérative basée sur le concept de l'Espérance-Maximisation (EM), qui converge en quelques itérations, est utilisée pour la maximiser. Étant donné le fait que cette technique est itérative, la manière avec laquelle elle est initialisée a un énorme impact sur le résultat final. Ceci nous amène à la troisième version dite hybride. Cette version utilise les résultats de l'estimateur DA pour initialiser celui du type NDA.

Les performances de cet estimateur dépassent celles du seul estimateur existant dans la littérature, appelé WGM, qui s'est attaqué à l'estimation ML basée sur le concept EM du SNR sur des canaux variables dans le temps [30]. Ceci est démontré dans la Figure 3.1 qui contient les courbes de l'erreur moyenne quadratique normalisée (NMSE) en fonction du SNR moyen par symbole (γ) pour différents nombres d'antennes.

Motivés par l'innovation de cet estimateur ainsi que par ses performances, nous nous sommes penchés sur sa complexité pour déterminer la faisabilité de son développement et son implémentation. L'absence de calculs mathématiques complexes se faisant en temps réel (Opérations matri-

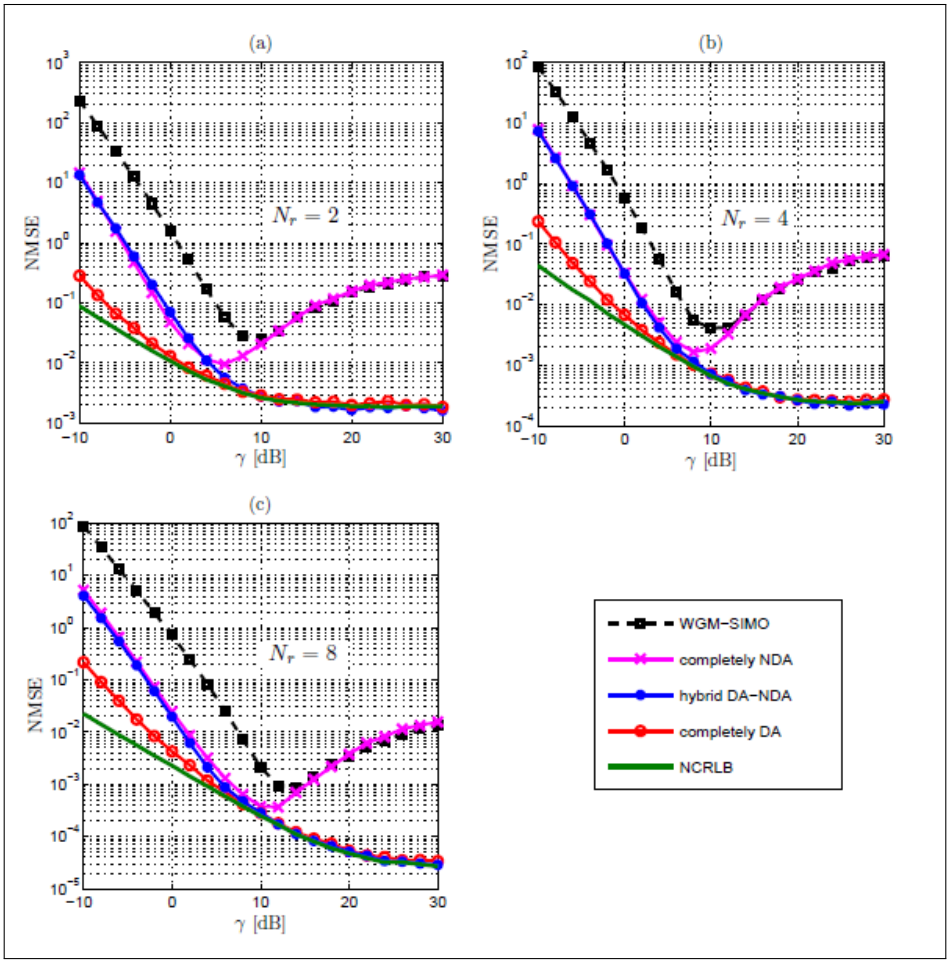


Figure 3.1 – Comparaison de l'estimateur choisi avec WGM-SIMO pour différents nombres d'antennes: (a) $N_r = 2$**, (b)** $N_r = 4$**, et (c)** $N_r = 8$ **, avec** $F_D T_s = 7 \times 10^{-3}$**,** $N = 112$**,** $\bar{N}_{DA} = 112$**,** $\bar{N}_{NDA} = 56$ **et** $L = 4$**, QPSK.**

cielles, etc.) et l'adaptabilité de son algorithme à notre environnement expérimental ont confirmé notre choix.

3.3 Description de l'estimateur

3.3.1 Modèle du système

Nous considérons la transmission numérique d'un signal modulé linéairement (modulation M-aire) à travers un système de communication à une seule entrée et plusieurs sorties (SIMO) dont les canaux à évanouissement plat variables dans le temps. Le signal reçu par l'antenne i, $i = 1, 2, \cdots, N_r$,

s'écrit :

$$y_i(t_n) = h_i(t_n)a(t_n) + w_i(t_n), \quad n = 1, 2, \cdots, N \tag{3.1}$$

où $\{t_n = nT_s\}_{n=1}^{N}$ est le $n^{ème}$ instant discret, T_s est la période du symbole et N est la taille de la fenêtre d'observation. On note par $a(t_n)$ le symbole modulé transmis, $y_i(t_n)$ le symbole reçu correspondant et par $h_i(t_n)$ le gain complexe du canal à travers chaque antenne i. Le bruit $w_i(t_n)$ est Gaussien blanc additif uniforme et la variance de chacune de ses parties complexes est σ^2.

Les coefficients du canal sont mesurés à travers une expansion polynomiale dans le temps d'ordre $(L-1)$ comme suit :

$$h_i(t_n) = \sum_{l=0}^{L-1} c_i^{(l)} t_n^l, \qquad i = 1, 2, \cdots, N_r. \tag{3.2}$$

Dans ce qui suit, nous présontons les versions DA, NDA et hybride de l'estimateur.

3.3.2 Estimateur DA

La technique d'estimation DA ML se base sur l'utilisation de symboles pilotes durant le processus d'estimation. De ce fait, nous supposons que N' symboles parmi N sont connus et qu'ils sont transmis tous les $T'_s = N_p T_s$ où $N_p \geq 1$ est un entier.
Les coefficients du canal à chaque position pilote, $t' = nT'_s$, sont :

$$h_{i,k}(t'_n) = \sum_{l=0}^{L-1} c_{i,k}^{(l)} t'^l_n, \qquad i = 1, 2, \cdots, N_r. \tag{3.3}$$

Pour nos futurs calculs, on définit les vecteurs suivants :

$$\mathbf{h}'_{i,k} = [h_{i,k}(t'_1), h_{i,k}(t'_2), \cdots, h_{i,k}(t'_{\bar{N}'_{\mathrm{DA}}})]^T, \tag{3.4}$$

$$\mathbf{w}'_{i,k} = [w_{i,k}(t'_1), w_{i,k}(t'_2), \cdots, w_{i,k}(t'_{\bar{N}'_{\mathrm{DA}}})]^T, \tag{3.5}$$

$$\mathbf{c}_{i,k} = [c_{i,k}^{(0)}, c_{i,k}^{(1)}, \cdots, c_{i,k}^{(L-1)}]^T. \tag{3.6}$$

$\mathbf{h}'_{i,k}$ contient les coefficients du canal aux positions pilotes, $\mathbf{w}'_{i,k}$ est le vecteur bruit correspondant et $\mathbf{c}_{i,k}$ contient les coefficients de l'expansion polynomiale locale. On peut écrire :

$$\mathbf{h}'_{i,k} = \mathbf{T}'\mathbf{c}_{i,k}, \qquad i = 1, 2, \cdots, N_r, \tag{3.7}$$

où

$$\mathbf{T}' = \begin{pmatrix} 1 & t'_1 & \cdots & {t'_1}^{L-1} \\ 1 & t'_2 & \cdots & {t'_2}^{L-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t'_{\bar{N}'_{\mathrm{DA}}} & \cdots & {t'}^{L-1}_{\bar{N}'_{\mathrm{DA}}} \end{pmatrix}. \tag{3.8}$$

On définit $\mathbf{A}'_k = \mathrm{diag}\{a_k(t'_1), a_k(t'_2), \cdots, a_k(t'_{\bar{N}'_{\mathrm{DA}}})\}$ comme une matrice diagonale qui contient les symboles connus de la $k^{ème}$ fenêtre d'approximation. Ceci donne :

$$\mathbf{y}'^{(k)}_{i,\mathrm{DA}} = \mathbf{A}'_k \mathbf{T}' \mathbf{c}_{i,k} + \mathbf{w}'_{i,k} = \mathbf{\Phi}'_k \mathbf{c}_{i,k} + \mathbf{w}'_{i,k}, \tag{3.9}$$

où $\mathbf{\Phi}'_k = \mathbf{A}'_k \mathbf{T}'$ est une matrice $(\bar{N}'_{\mathrm{DA}} \times L)$ connue. Avec $\mathbf{y}'^{(k)}_{\mathrm{DA}} = [\mathbf{y}'^{(k)T}_{1,\mathrm{DA}}\ \mathbf{y}'^{(k)T}_{2,\mathrm{DA}}\ \cdots\ \mathbf{y}'^{(k)T}_{N_r,\mathrm{DA}}]^T$ on obtient :

$$\mathbf{y}'^{(k)}_{\mathrm{DA}} = \mathbf{B}'_k \mathbf{c}_k + \mathbf{w}'_k, \tag{3.10}$$

où $\mathbf{c}_k = [\mathbf{c}^T_{1,k}\ \mathbf{c}^T_{2,k}\ \cdots\ \mathbf{c}^T_{N_r,k}]^T$ et $\mathbf{w}'_k = [\mathbf{w}'^T_{1,k}\ \mathbf{w}'^T_{2,k}\ \cdots\ \mathbf{w}'^T_{N_r,k}]^T$ sont des vecteurs colonnes de dimensions LN_r- et $\bar{N}'_{\mathrm{DA}} N_r$. $\mathbf{B}'_k = \mathrm{blkdiag}\{\mathbf{\Phi}'_k, \mathbf{\Phi}'_k, ..., \mathbf{\Phi}'_k\}$ est une matrice diagonale en blocs de dimensions $(\bar{N}'_{\mathrm{DA}} N_r \times LN_r)$.

Finalement, après le calcul de la fonction de log-likelihood du DA, on obtient les estimés ML des coefficients polynomiaux locaux sur toutes les antennes :

$$\widehat{\mathbf{c}}_{k,\mathrm{DA}} = \left(\mathbf{B}'^H_k \mathbf{B}'_k\right)^{-1} \mathbf{B}'^H_k \mathbf{y}'^{(k)}_{\mathrm{DA}}, \tag{3.11}$$

et par suite les coefficients du canal aux positions pilotes pour chaque fenêtre d'approximation $\{\widehat{\mathbf{h}}'^{(k)}_{i,\mathrm{DA}} = \mathbf{T}' \widehat{\mathbf{c}}^{(k)}_{i,\mathrm{DA}}\}_k$.

Les estimés ML de la variance du bruit sont obtenus pour chaque fenêtre d'approximation comme suit :

$$\widehat{\sigma^2}_{k,\mathrm{DA}} = \frac{1}{2\bar{N}'_{\mathrm{DA}} N_r} [\mathbf{y}^{(k)}_{\mathrm{DA}} - \mathbf{B}_k \widehat{\mathbf{c}}_{k,\mathrm{DA}}]^H [\mathbf{y}^{(k)}_{\mathrm{DA}} - \mathbf{B}_k \widehat{\mathbf{c}}_{k,\mathrm{DA}}], \tag{3.12}$$

puis moyennés pour donner :

$$\widehat{\sigma^2}_{\mathrm{DA}} = \frac{\bar{N}_{\mathrm{DA}}}{N} \sum_{k=1}^{N/\bar{N}_{\mathrm{DA}}} \widehat{\sigma^2}_{k,\mathrm{DA}}. \tag{3.13}$$

3.3.3 Estimateur NDA

Contrairement à l'estimateur DA qui s'écrit sous forme d'une expression analytique, l'estimateur NDA est itératif. Il est composé de deux étapes. L'étape d'espérance où la fonction de vraisemblance est calculée par rapport à tous les symboles transmis possibles a_m. S'ensuit l'étape de la maximisation par rapport à tous les paramètres inconnus.

La fonction de vraisemblance est définie par $L(\boldsymbol{\theta}_k|a_k(n) = a_m) = p(\mathbf{y}_k(n); \boldsymbol{\theta}_k|a_k(n) = a_m)$ où $p(\mathbf{y}_k(n); \boldsymbol{\theta}_k|a_k(n) = a_m)$ la fonction de densité de probabilité du vecteur reçu $\mathbf{y}_k(n)$ conditionnée par le symbole transmis $a_k(n)$. $\boldsymbol{\theta}_k$ est le vecteur qui contient tous les coefficients du canal inconnus et la puissance du bruit dans la $k^{\text{ème}}$ fenêtre d'approximation de longueur N_{NDA}.
Ensuite, les paramètres qui maximisent l'étape d'espérance sont calculés:

$$\widehat{\boldsymbol{\theta}_k}^{(q)} = \arg\max_{\boldsymbol{\theta}_k} \; Q\left(\boldsymbol{\theta}_k|\widehat{\boldsymbol{\theta}_k}^{(q-1)}\right), \tag{3.14}$$

où $Q\left(\boldsymbol{\theta}_k|\widehat{\boldsymbol{\theta}_k}^{(q-1)}\right)$ est la fonction objective qui est mise à jour itérativement commençant par une valeur initiale $\boldsymbol{\theta}_k^{(0)}$.
Pour une fenêtre d'observation, composée de k fenêtres d'approximation, l'algorithme est itéré q fois sur chaque fenêtre d'approximation k de taille N_{NDA}. Au final, les coefficients du canal, la puissance du bruit ainsi que le SNR sont obtenus sur chaque fenêtre d'observation. L'algorithme d'une itération q se présente comme suit:

- La probabilité *a posteriori* de a_m à l'itération $(q-1)$ est calculée :

$$P_{m,n,k}^{(q-1)} = \frac{P[a_m] P\left(\mathbf{y}_k(n)|a_m; \widehat{\boldsymbol{\theta}_k}^{(q-1)}\right)}{P\left(\mathbf{y}_k(n); \widehat{\boldsymbol{\theta}_k}^{(q-1)}\right)}, \tag{3.15}$$

avec

$$P\left(\mathbf{y}_k(n); \widehat{\boldsymbol{\theta}_k}^{(q-1)}\right) = \frac{1}{M} \sum_{m=1}^{M} P\left(\mathbf{y}_k(n)|a_m; \widehat{\boldsymbol{\theta}_k}^{(q-1)}\right), \tag{3.16}$$

et $P\left(\mathbf{y}_k(n)|a_m;\widehat{\boldsymbol{\theta}_k}^{(q-1)}\right)$ étant:

$$p(\mathbf{y}_k(n);\boldsymbol{\theta}_k|a_k(n)=a_m) = \frac{1}{(2\pi\sigma^2)^{N_r}} \times \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N_r}\left|y_{i,k}(n)-a_m\mathbf{c}_{i,k}^T\mathbf{t}(n)\right|^2\right\}. \tag{3.17}$$

- Les coefficients du canal sont ensuite estimés :

$$\widehat{\mathbf{c}}_{i,k}^{(q)} = \left(\sum_{n=1}^{\bar{N}_{\mathrm{NDA}}}\mathbf{t}(n)\mathbf{t}^T(n)\right)^{-1}\left(\sum_{n=1}^{\bar{N}_{\mathrm{NDA}}}\lambda_{i,n,k}^{(q-1)}\mathbf{t}(n)\right), \tag{3.18}$$

$$\lambda_{i,n,k}^{(q-1)} = [\widehat{a}_k^{(q-1)}(n)]^* y_{i,k}(n), \tag{3.19}$$

$$\widehat{a}_k^{(q-1)}(n) \;=\; \sum_{m=1}^{M} P_{m,n,k}^{(q-1)} a_m. \tag{3.20}$$

- La probabilité est recalculée en prenant en compte les nouveaux coefficients.
- La puissance du bruit est alors estimée par:

$$\widehat{2\sigma^2}_k^{(q)} = \frac{\sum_{i=1}^{N_r}\left(M_{2,k}^{(i)}+\eta_{i,k}^{(q-1)}\right)}{\bar{N}_{\mathrm{NDA}}N_r}, \tag{3.21}$$

où $M_{2,k}^{(i)}$ est le moment d'ordre deux des symboles reçus sur la $i^{ème}$ antenne.

$$\begin{aligned}\eta_{i,k}^{(q-1)} &= \sum_{n=1}^{\bar{N}_{\mathrm{NDA}}}\Big[\mathbf{t}^T(n)\left(\widehat{\mathbf{c}}_{i,k}^{(q-1)}\right)^*\left(\widehat{\mathbf{c}}_{i,k}^{(q-1)}\right)^T\mathbf{t}(n) + \\ &\qquad\qquad \alpha_{n,k}^{(q-1)} - 2\beta_{i,n,k}^{(q-1)}\left(\widehat{\mathbf{c}}_{i,k}^{(q-1)}\right)\Big] \\ &= \sum_{n=1}^{\bar{N}_{\mathrm{NDA}}}\left[\left|\mathbf{t}^T(n)\widehat{\mathbf{c}}_{i,k}^{(q-1)}\right|^2 + \alpha_{n,k}^{(q-1)} - 2\beta_{i,n,k}^{(q-1)}\left(\widehat{\mathbf{c}}_{i,k}^{(q-1)}\right)\right],\end{aligned} \tag{3.22}$$

et

$$\alpha_{n,k}^{(q-1)} = E_{a_m}\left\{|a_m|^2 \middle| \widehat{\boldsymbol{\theta}_k}^{(q-1)}, \mathbf{y}_k(n)\right\} \tag{3.23}$$

$$= \sum_{m=1}^{M} P_{m,n,k}^{(q-1)} |a_m|^2, \tag{3.24}$$

$$\beta_{i,n,k}^{(q-1)}(\mathbf{c}_{i,k}) = E_{a_m}\left\{\Re\{y_{i,k}^*(n) a_m \mathbf{t}^T(n)\mathbf{c}_{i,k}\} \middle| \widehat{\boldsymbol{\theta}_k}^{(q-1)}, \mathbf{y}_k(n)\right\}$$

$$= \sum_{m=1}^{M} P_{m,n,k}^{(q-1)} \Re\left\{y_{i,k}^*(n) a_m \mathbf{t}^T(n)\mathbf{c}_{i,k}\right\}. \tag{3.25}$$

Après q itérations, la puissance du bruit est moyennée sur toutes les fenêtres d'approximation comme suit :

$$\widehat{\sigma^2}_{\mathrm{NDA}} = \frac{\bar{N}_{\mathrm{NDA}}}{N} \sum_{k=1}^{N/\bar{N}_{\mathrm{NDA}}} \widehat{\sigma^2_{k,\mathrm{NDA}}}. \tag{3.26}$$

Finalement, le SNR est obtenu sur chaque antenne comme suit:

$$\widehat{\rho}_{i,\mathrm{NDA}} = \frac{\sum_{k=1}^{N/\bar{N}_{\mathrm{NDA}}} \sum_{n=1}^{\bar{N}_{\mathrm{NDA}}} |\widehat{a}_k(n)|^2 \left|\mathbf{t}^T(n)\widehat{\mathbf{c}}_{i,\mathrm{NDA}}^{(k)}\right|^2}{N\left(2\widehat{\sigma^2}_{\mathrm{NDA}}\right)}. \tag{3.27}$$

La version non biaisée du SNR est calculée ainsi :

$$\widehat{\rho}_{i,\mathrm{NDA}}^{\mathrm{UB}} = \frac{N_r N(1-\epsilon)-1}{N_r N}\widehat{\rho}_{i,\mathrm{NDA}} - \frac{\epsilon}{2}, \tag{3.28}$$

où $\epsilon = L/\bar{N}_{\mathrm{NDA}}$ et $(L-1)$ est l'ordre de l'expansion temporelle.

3.3.4 Estimateur hybride: Initialisation de l'algorithme EM

Étant itératif, l'algorithme EM a besoin d'être initialisé. Il existe deux manières de le faire. Soit il est initialisé arbitrairement, ce qui n'assure pas de bonnes performances. Soit il est initialisé avec l'algorithme DA pour constituer un estimateur hybride performant. Nous avons opté pour ce dernier choix.

Les coefficients estimés du canal à partir des symboles pilotes sont obtenus aux positions pilotes et non pilotes sur chaque fenêtre d'approximation DA de la manière suivante :

$$\widehat{\mathbf{h}}_{i,\mathrm{DA}}^{(k)} = \mathbf{T}_{\bar{N}_{\mathrm{DA}}} \widehat{\mathbf{c}}_{i,\mathrm{DA}}^{(k)}. \tag{3.29}$$

Les paramètres d'initialisation sont :

$$\begin{aligned} \widehat{\mathbf{c}}_{i,k}^{(0)} &= \left(\mathbf{T}_{\bar{N}_{\mathrm{NDA}}}^{T} \mathbf{T}_{\bar{N}_{\mathrm{NDA}}}\right)^{-1} \mathbf{T}_{\bar{N}_{\mathrm{NDA}}}^{T} \widehat{\mathbf{h}}_{i,\mathrm{DA}}^{(k)} \\ &\qquad \text{for} \quad k = 1, 2, \cdots, N/\bar{N}_{\mathrm{NDA}}, \end{aligned} \tag{3.30}$$

et

$$\widehat{\sigma^2}^{(0)} = \widehat{\sigma^2}_{\mathrm{DA}}. \tag{3.31}$$

3.4 Conclusion

Dans ce chapitre, nous avons présenté l'estimateur de SNR qui a été développé par notre équipe du Wireless Lab et nous avons détaillé son concept et ses versions DA, NDA et Hybride. Dans le chapitre suivant, nous présenterons la phase de conception et d'implémentation de cet estimateur et nous parlerons brièvement des problèmes rencontrés.

Chapitre 4

Conception et implémentation de l'estimateur du SNR

4.1 Introduction

L'implémentation de l'estimateur sur la plateforme SDR, comme tout processus de développement, passe par une phase de conception. Pendant cette phase, le système est divisé en sous-modules qui exécutent, chacun, une tâche spécifique. Les modules sont définis par rapport aux fonctions algorithmiques de l'estimateur. Ces fonctions sont constituées des équations mathématiques qui définissent les entrées et les sorties de chaque sous-module.

Dans ce qui suit, nous définissons les différents paramètres avec lequels nous avons configuré l'estimateur pour pouvoir procéder aux étapes de conception et d'implémentation qui seront décrites. Nous mettons l'accent sur les notions de chemin des données et chemin de contrôle. Ensuite, nous introduisons quelques solutions d'optimisation à travers les blocs personnalisés. À la fin, nous parlons de la génération de l'exécutable et de sa configuration.

4.2 Paramétrage de l'estimateur

Pour le développement du design de l'estimateur, nous avons configuré les paramètres de manière à prendre en considération la complexité du système ainsi que les ressources matérielles disponibles. Nous avons choisi de travailler dans le scénario pratique de l'Uplink LTE. Un symbole pilote est inséré à chaque $7^{ème}$ position de la sous-séquence envoyée (contenant 14 symboles). La fenêtre d'observation est composée de huit sous-séquences consécutives. Le nombre d'antennes est fixé à un pour la transmission et deux pour la réception (configuration SIMO). Nous avons aussi choisi de travailler avec une fréquence Doppler maximale $F_D T_s \leq 7 \times 10^{-3}$ et avec une modulation QPSK. Ceci implique une configuration des paramètres de l'estimateur telle que montrée dans le tableau suivant:

Tableau 4.1 – Configuration des paramètres de l'estimateur

Taille de la fenêtre d'observation	N	112
Taille de la fenêtre d'approximation NDA	N_{NDA}	56
Taille de la fenêtre d'approximation DA	N_{DA}	112
Nombre de symboles de la constellation	M	4 $(QPSK)$
Ordre du polynôme temporel NDA: $(L_{NDA} - 1)$	L_{NDA}	4
Ordre du polynôme temporel DA: $(L_{DA} - 1)$	L_{DA}	4
Période du symbole	T_s	7.1429×10^{-5}
Période du symbole pilote	$T'_s = 7 \times T_s$	$7 \times 7.1429 \times 10^{-5}$
Nombre d'itérations NDA	q	10
Fréquence Doppler maximale	F_D^{max}	100

4.3 Conception et implémentation

La phase de conception permet de spécifier, dans un premier lieu, le chemin des données et, en conséquence, le chemin de contrôle. En effet, le chemin des données est caractérisé par l'ensemble des ressources nécessaires pour traiter les tâches de chaque sous-module telles que les ressources mémoires, les additionneurs, les multiplicateurs, etc. L'architecture du chemin des données étant combinatoire, il est crucial de contrôler le flow de ces données à l'aide de signaux afin de synchroniser le fonctionnement des différentes unités. Ceci détermine le chemin de contrôle.

4.3.1 Chemin des données

Architecture globale

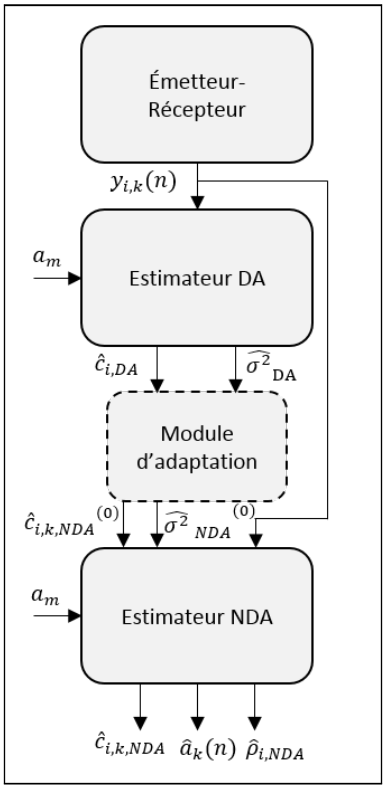


Figure 4.1 – Diagramme en blocs du design global.

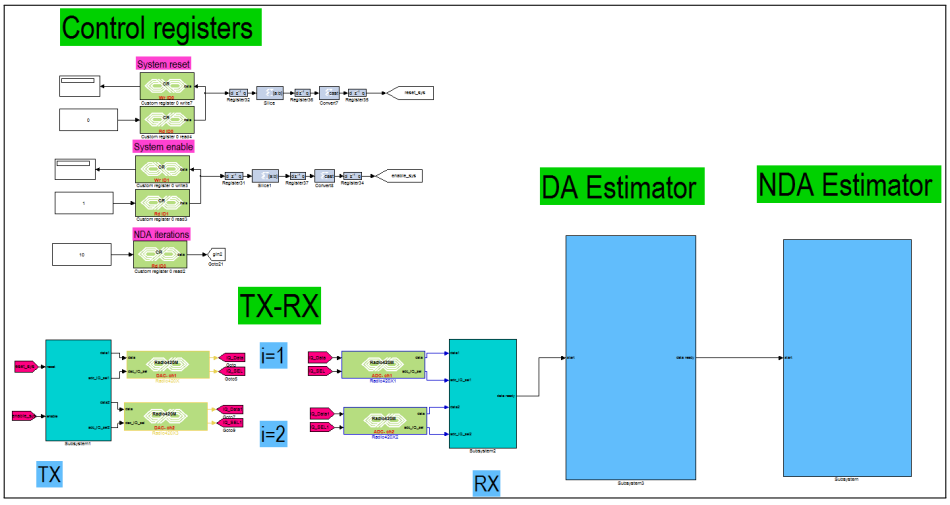


Figure 4.2 – Implémentation dans l'environnement MATLAB/Simulink et Xilinx du design global.

Les Figures 4.1 et 4.2 montrent, respectivement, le diagramme en blocs du design global et son implémentation dans l'environnement MATLAB/Simulink et Xilinx. Le design est constitué, principalement, de quatre modules :

- Le module de l'émetteur-récepteur
- Le module de l'estimateur DA
- Le module d'adaptation
- Le module de l'estimateur NDA

Ces quatre modules constituent le chemin des données de notre design. Leurs entrées et leurs sorties sont bien déterminées à la Figure 4.2.

Le module de l'émetteur-récepteur

Ce module a pour rôle d'envoyer et recevoir, à travers la carte radio de la SDR, une séquence de symboles QPSK. Un symbole QPSK étant constitué d'une partie I (partie en phase) et une partie Q (partie en quadrature de phase), ces deux parties sont envoyées entrelacées à travers le CNA qui les transforme en un signal RF. Après le passage par le canal de transmission sans fil, ce signal est reçu par le CAN et les parties I et Q sont démêlées et enregistrées pour traitement. Aussi, ce module permet-il l'extraction et l'enregistrement des symboles pilotes pour l'estimation DA.

Le module de l'estimateur DA

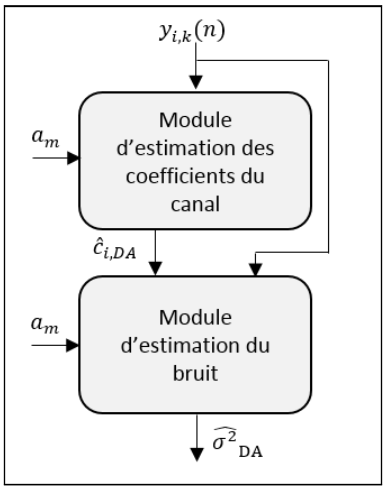


Figure 4.3 – Diagramme en blocs de l'estimateur DA.

Ce module exécute l'algorithme de l'estimateur DA qui se présente sous forme d'une expression analytique décrite dans les sections précédentes. Ce module prend comme entrées les symboles reçus aux positions pilotes. Une connaissance des symboles a_m de la constellation QPSK est nécessaire.

Dans un premier temps, les coefficients du canal sont estimés par le sous-module d'estimation des coefficients du canal selon l'équation (3.11). Ensuite, ils sont passés au deuxième sous-module d'estimation du bruit qui estimera la puissance du bruit selon les équations (3.12) et (3.13).

Dans le but d'éviter les calculs matriciels complexes inhérents aux équations de l'estimateur DA, il suffit, en plus du paramétrage de l'estimateur, de fixer au préalable la valeur des symboles pilotes. Cela n'influe aucunement sur les résultats de l'estimateur étant donné que ces symboles pilotes fixés seront injectés aux positions pilotes au fur et à mesure de l'envoi de la séquence. Ainsi, les calculs matriciels sont faits « offline » et sont chargés directement dans les unités mémoires du design.

Le module d'adaptation

Le module d'adaptation vient s'insérer entre le module de l'estimateur DA et celui NDA. Son rôle étant d'initialiser l'estimateur NDA avec les paramètres estimés par celui DA selon les équations (3.30) et (3.31).

Le module de l'estimateur NDA

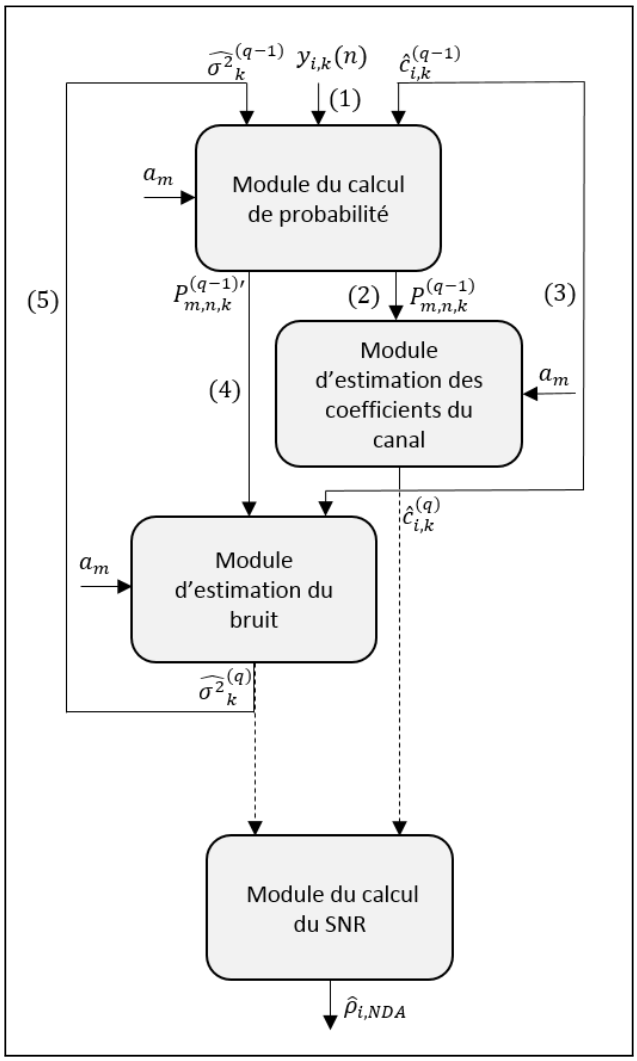


Figure 4.4 – Diagramme en blocs de l'estimateur NDA.

Le diagramme de la Figure 4.4 montre les quatre différents sous-modules du module responsable de l'estimation NDA, leurs entrées et sorties ainsi que le déroulement des itérations.

Une itération comprend trois sous-modules qui sont interconnectés. L'enchaînement d'une itération se fait comme suit :

1. Le sous-module du calcul de probabilité reçoit les paramètres d'initialisation du module d'adaptation lors de la première itération et ces mêmes paramètres sont mis à jour lors des itérations suivantes. Il transmet la probabilité du symbole $P_{m,n,k}^{(q-1)}$ au sous-module d'estimation des coefficients du canal.

2. Le sous-module suivant estime les coefficients du canal $\widehat{\mathbf{c}}_{i,k}^{(q)}$ à partir de la probabilité $P_{m,n,k}^{(q-1)}$.
3. La probabilité $P_{m,n,k}^{(q-1)\prime}$ est recalculée en prenant en compte les nouveaux coefficients du canal.
4. Le module d'estimation du bruit reçoit la nouvelle probabilité $P_{m,n,k}^{(q-1)\prime}$ ainsi que les coefficients estimés $\widehat{\mathbf{c}}_{i,k}^{(q)}$ pour estimer la puissance du bruit du canal $\widehat{\sigma^2}^{(q)}$.
5. Si le nombre d'itérations q n'est pas encore atteint, les nouveaux paramètres estimés $\widehat{\sigma^2}^{(q)}$ et $\widehat{\mathbf{c}}_{i,k}^{(q)}$ sont réinjectés au sous-module de calcul de probabilité afin de lancer une nouvelle itération.

A la fin de toutes les itérations, le dernier sous-module calcule le SNR $\widehat{\rho}_{i,\mathrm{NDA}}$ sur chaque antenne i.

4.3.2 Chemin de contrôle

Le chemin de contrôle de notre design global, Figure 4.5, est composé essentiellement de machines séquentielles algorithmiques (MSA). Les MSA développées sont des machines de Moore dont les sorties sont contrôlées uniquement par leur états présents.

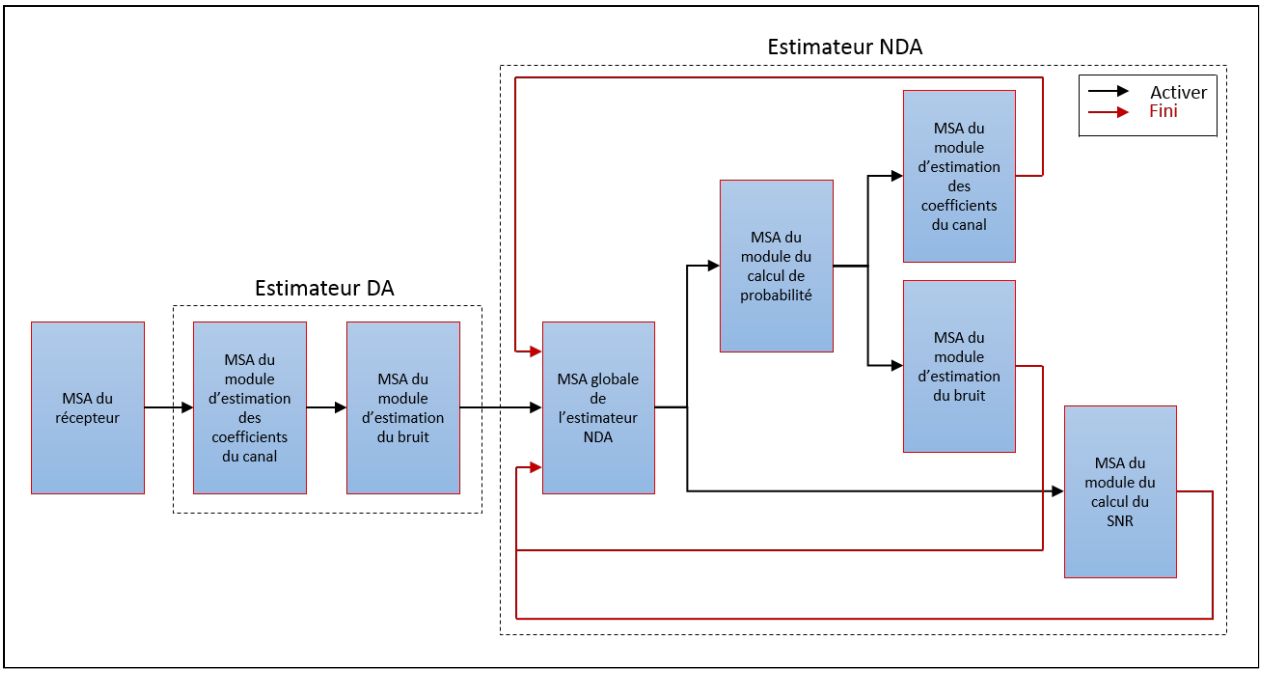


Figure 4.5 – Chemin de contrôle du design global.

Chaque sous-module est principalement contrôlé par une MSA qui s'occupe d'assurer la synchronisation entre ses différents éléments. De plus, elle permet d'activer celle en aval lorsque le fonctionnement du sous-module correspondant est fini.

Le besoin d'avoir une MSA globale pour l'estimateur NDA vient du fait qu'il soit itératif et que ses sous-modules doivent suivre un enchainement chronologique particulier. En effet, cette MSA globale synchronise les MSA des différents sous-modules entre elles en se basant sur leurs signaux de retour.

Le contrôle du design doit aussi se faire en temps réel lorsque l'algorithme est entrain de rouler sur la plateforme SDR. Cette partie importante du chemin de contrôle est assurée par les registres de contrôle qui permettent une communication entre la plateforme et l'unité de commande (ordinateur). Les fonctions principales de ces registres étant de démarrer le système ou de le réinitialiser.

4.3.3 Blocs de modélisation personnalisés

La librairie Xilinx dans l'environnement MATLAB/Simulink offre un ensemble de blocs qui permettent de modéliser des opérations logiques ou mathématiques plus ou moins simples pour le traitement des signaux. Néanmoins, lors du développement du design, on se retrouve face à des opérations plus complexes et récurrentes telles que la sommation, la multiplication vectorielle ou matricielle, les fonctions mathématiques, etc. Ceci nous amène à créer notre propre librairie qui est constituée de blocs personnalisés, fonctionnels et réutilisables dans plusieurs designs. Cette librairie ajoute un niveau d'abstraction et facilite ainsi la construction de designs. De plus, cette approche nous évite de re-modéliser à chaque fois les mêmes fonctions et offre, par conséquent, un gain de temps très important. Dans la suite, nous présenterons quelques-uns des blocs créés et décrirons leurs fonctions premières.

Multiplication de matrices complexes

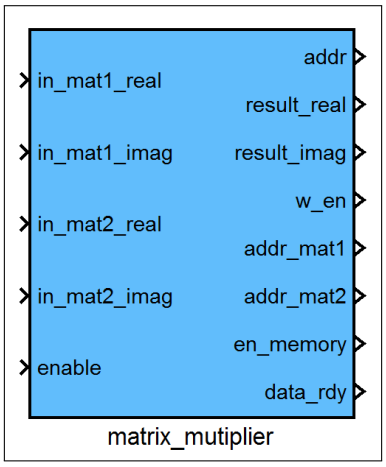


Figure 4.6 – Bloc de multiplication de matrices complexes.

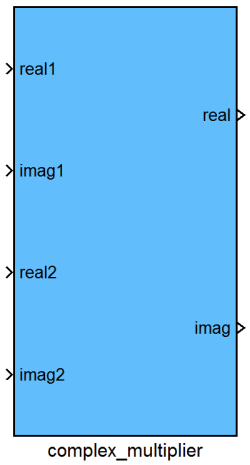
Les algorithmes des estimateurs développés contiennent plusieurs opérations de multiplication matricielle et vectorielle réels ou complexes. Le bloc montré dans la Figure 4.6 permet de faire de telles opérations. Il est paramétrable selon les dimensions des matrices et leur type d'éléments (réels ou complexes). Ce module est géré par une MSA qui contrôle les entrées et les sorties et synchronise les tâches séquentiellement. Il est principalement constitué d'accumulateurs, de multiplicateurs, d'additionneurs, de compteurs et de registres.

Il prend comme entrées les valeurs des éléments matriciels qui sont enregistrés dans des unités de mémoires. Les adresses et les signaux de contrôle de ces dernières ainsi que ceux de l'unité où le résultat sera enregistré sont données comme sorties. Ce bloc est actionné par un signal d'activation «enable» et possède un signal «data_rdy» qui passe à «1» quand les résultats sont prêts.

Opérations sur les nombres complexes

Pour faciliter les opérations récurrentes avec les nombres complexes, nous avons créé certains blocs qui exécutent des tâches spécifiques telles que la multiplication de nombres complexes ou le calcul du module au carré d'un nombre complexe. Le but étant de pouvoir les réutiliser là où nécessaire.

(a) Bloc de multiplication complexe.



(b) Éléments Xilinx du bloc de multiplication complexe.

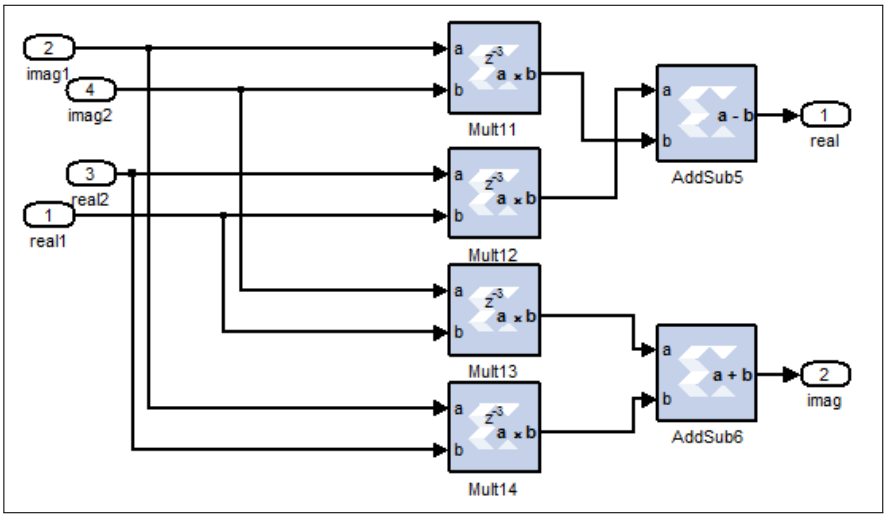
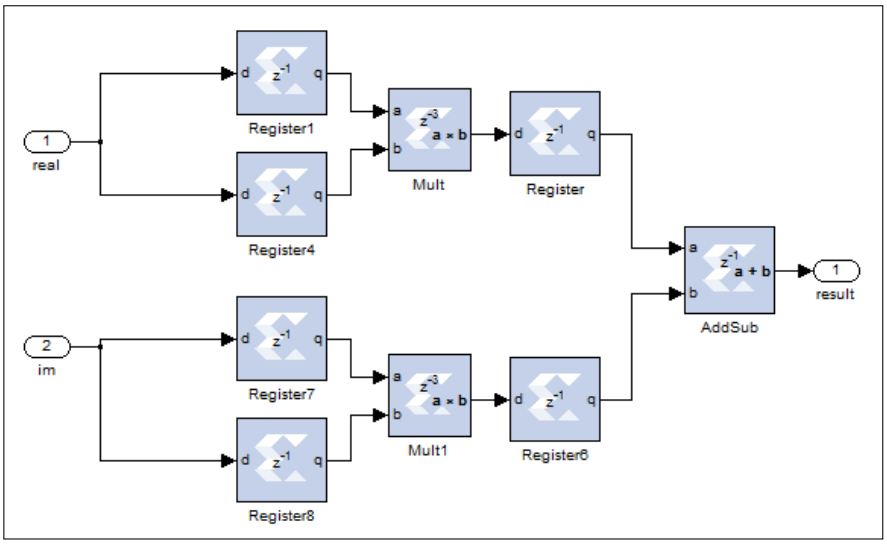


Figure 4.7 – Bloc de multiplication complexe et ses composants.

(b) Éléments Xilinx du bloc de calcul du module au carré.



(a) Bloc de calcul du module au carré.

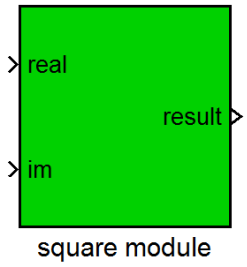


Figure 4.8 – Bloc de calcul du module au carré et ses composants.

Ces deux blocs, illustrés aux Figures 4.7 et 4.8, prennent comme entrées les parties imaginaires et réelles séparées des nombres complexes et donnent à la sortie le résultat demandé. Ils sont relativement simples et ne nécessitent pas de MSA.

Sommation

(b) Éléments Xilinx du bloc de sommation.

(a) Bloc de sommation.

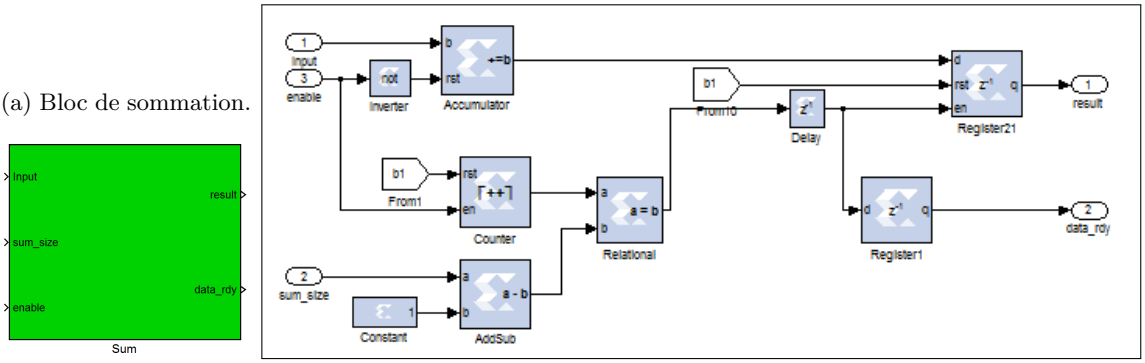


Figure 4.9 – Bloc de sommation et ses composants.

Le bloc de sommation permet de faire des sommes de données de taille (sum_size) réglable. La somme n'est autre qu'une accumulation de la valeur à l'entrée du bloc à chaque coup d'horloge. Ce bloc est actionné par un signal d'activation «enable» et possède un signal «data_rdy» qui passe à «1» quand les résultats sont prêts.

Les fonctions mathématiques

L'implémentation de fonctions mathématiques dans le design se base essentiellement sur les LUTs. En effet, si on veut implémenter une fonction f où $x \in [\alpha, \beta]$ est notre variable et $n \in [0, N-1]$ est la taille de l'unité de mémoire utilisée (i.e. ROM), alors x peut être écrit comme suit :

$$x = \alpha + \frac{\beta - \alpha}{N - 1} \times n. \tag{4.1}$$

Ce qui donne :

$$n = (x - \alpha) \times \frac{N - 1}{\beta - \alpha}. \tag{4.2}$$

D'abord, l'unité de mémoire est remplie « offline » avec un vecteur initial dont les composants sont enregistrés un par un dans une adresse mémoire. Chaque composant a une expression MATLAB définie par $f(x) = f(\alpha + \frac{\beta-\alpha}{N-1} \times n)$. Pendant que le système est en train de rouler, n est calculée à partir de l'entrée x et est injectée comme adresse. L'unité de mémoire donne comme sortie la vraie valeur de $f(x)$. Naturellement, plus la taille de la mémoire est grande, plus le résultat est

précis. Dans notre cas, nous avons utilisé deux blocs d'implémentation de fonctions : Un bloc pour la fonction exponentielle ($f(x) = exp(x)$) et un bloc pour la fonction inverse ($f(x) = \frac{1}{x}$).

4.4 Génération du bitstream

L'étape suivante dans le processus d'implémentation de l'algorithme sur la plateforme SDR est l'étape de la génération du PSM à partir du PIM déjà développé et testé en simulation. La génération du HDL et du fichier bitstream, qui est exécutable sur machine, se fait automatiquement. Il est nécessaire, cependant, de spécifier le type de matériel ciblé. Les caractéristiques de ce dernier doivent avoir été préalablement installées et configurées dans le HDL Coder, System Generator dans notre cas. Pour cela, il suffit d'ajouter le bloc fourni avec la MBDK de la plateforme SDR au design comme le montre la Figure 4.10 (a). Ce bloc permet, entre autres, de choisir l'horloge source du système.

(a) Configuration du Perseus 601x. (b) Paramétrage de System Generator.

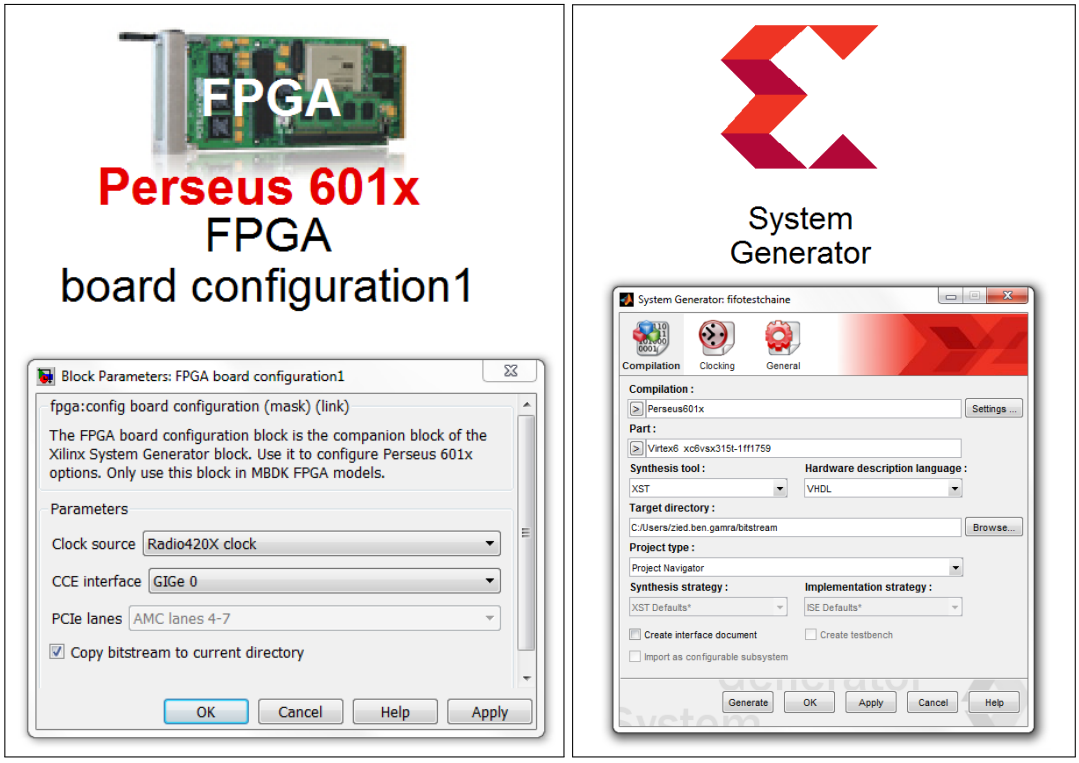


Figure 4.10 – Génération du bitstream.

Il ne reste, ensuite, qu'à paramétrer le bloc System Generator et lancer la génération, comme le montre la Figure 4.10 (b). Les configurations les plus importantes étant :

- Le type de compilation choisi
- La carte FPGA ciblée
- Le langage de description de matériel voulu
- L'outil de synthèse

Les problèmes auxquels nous avons fait face dans cette étape ont été abordés précédemment. Le problème principal étant celui du chemin critique qui reste relativement facile à résoudre. La fréquence d'horloge de notre plateforme est égalle à 80 MHz. La solution à ce problème consiste en l'insertion de registres là où nécessaire afin d'améliorer le chemin critique et adapter le design à cette fréquence.

4.5 Conclusion

Dans ce chaptire, nous avons détaillé les phases de conception et d'implémentation de l'estimateur dans l'environnement MATLAB/Simulink et Xilinx System Generator en présentant les différents blocs ainsi que leurs entrées et sorties tant bien que les chemins des données et de contrôle. Nous avons introduit quelques blocs personnalisés qui nous ont beaucoup aidés dans le développement du design en termes de gain en temps. Finalement, nous avons montré la facilité de la génération de l'exécuatble pour l'implémenter sur la machine grâce à l'approche MBD choisie. Dans le chapitre suivant, nous décrirons notre station de travail, nous évaluerons les performances du PSM implémenté sur la plateforme SDR et les comparerons à celles de l'algorithme sur MATLAB.

Chapitre 5

Description de l'environnement de travail et validation des performances

5.1 Introduction

L'étape ultime dans notre projet de développement est l'étape de test et de validation. Cette étape nous permet de savoir si l'algorithme implémenté sur la plateforme a le même comportement et donne les mêmes résultats que celui développé sur logiciel. La résolution des problèmes reliés à cette étape a été abordée dans les sections précédentes. Ceci nous a permis de passer à la validation à proprement dite de l'implémentable.

Dans ce chapitre, nous commençons par la présentation de notre environnement de travail expérimental. Ensuite, nous évaluons le comportement de l'algorithme implémenté en ce qui concerne les différentes fonctions de l'estimateur choisi.

5.2 Environnement de travail

Notre environnement expérimental se compose de la plateforme SDR PicoSDR de Nutaq et de l'émulateur de canaux EB Propsim d'Anite. La plateforme est reliée à l'émulateur selon une configuration SIMO 1x2: une antenne émettrice et deux antennes réceptrices.

5.2.1 PicoSDR2X2 de Nutaq

Matériel

Figure 5.1 – Plateforme PicoSDR2X2 de Nutaq.

La PicoSDR de Nutaq est une plateforme SDR de simulation qui est utilisée pour modéliser, tester et déployer rapidement des algorithmes de télécommunication sans fil [31]. Le modèle de plateforme sur lequel nous avons travaillé est le PicoSDR2X2. Son architecture interne est montrée dans la Figure 5.2.

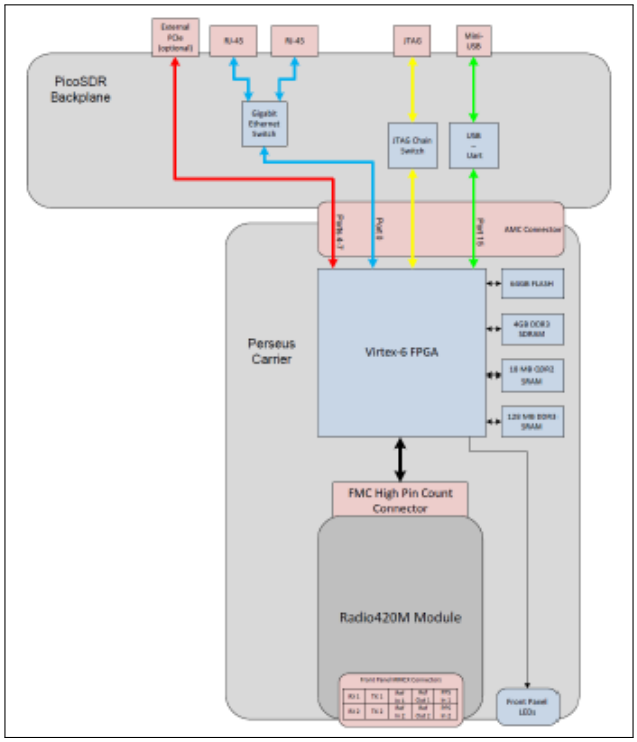


Figure 5.2 – Architecture interne de la plateforme PicoSDR2X2 de Nutaq.

Comme illustré, ce modèle de plateforme SDR est composé essentiellement de :

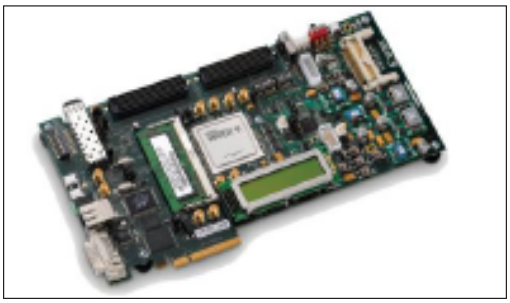


Figure 5.3 – Carte FPGA Virtex-6.

Une large carte FPGA Virtex-6 qui offre plus de ressources de traitement du signal numérique.

Figure 5.4 – Carte AMC Perseus601x.

Une carte AMC Perseus601x de traitement du signal numérique conçue pour de hautes performances, de larges bandes passantes et des applications de traitement à faible latence. Elle est idéale pour la réduction de la taille, de la complexité et des coûts associés aux télécommunications de pointe [32].

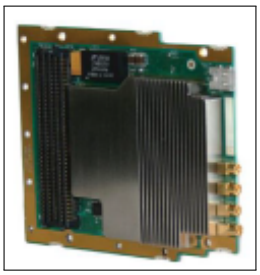


Figure 5.5 – Carte radio FMC 420m.

Une carte radio FMC 420m qui est un module émetteur-récepteur multimode de la radio logicielle. Il supporte une couverture large bande, les modes full duplex TDD et FDD et une bande

passante allant de 1.5 à 28 MHz avec une excellente sélectivité du canal. Il opère dans des fréquences comprises entre 300 MHz et 3 GHz [33].

Logiciel

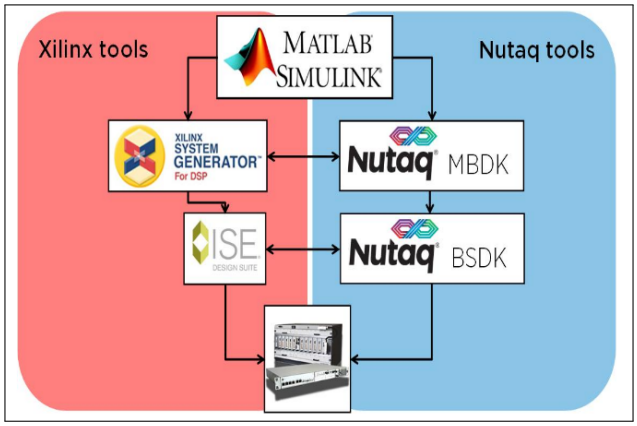


Figure 5.6 – Agencement des outils de Xilinx et des outils de Nutaq.

Nutaq propose des outils de développement qui s'alignent avec notre choix de développer le Design dans l'environnement GUI MATLAB/Simulink et Xilink System Generator. Un ensemble d'outils logiciels (BSDK et MBDK) est fourni avec la plateforme. Son MBDK vient s'ajouter à la librairie de fonctions DSP de Xilinx pour permettre à l'utilisateur de contrôler les interfaces entrées/sorties de la plateforme telles que des convertisseurs A/N haut débit, les interfaces mémoires, etc [34].

Ces outils permettent, également, le contrôle à distance ainsi que l'échange de données entre la plateforme et l'ordinateur. Ils offrent aussi une intégration en temps réel des entrées/sorties de la plateforme pendant la co-simulation [31].

5.2.2 Émulateur de canaux EB Propsim d'Anite

Figure 5.7 – Émulateur de canaux EB Propsim d'Anite.

Dans le but de simuler les propriétés du canal, on a décidé d'utiliser l'émulateur de canaux EB Propsim commercialisé par Anite. Il offre un grand choix de paramétrage des caractéristiques des canaux radios telles que l'affaiblissement de propagation, l'évanouissement multi-trajet, l'étalement Doppler, la polarisation, la corrélation ainsi que le réglage des paramètres spatiaux nécessaires pour la configuration MIMO et la performance des systèmes multi-radio. Avec ses 8 canaux RF, il est capable de générer un canal 4x4 MIMO bidirectionnel pour tester les performances sans fil des appareils mobiles et étudier différentes technologies telles que 4G, LTE, etc [35].

5.3 Validation des performances du PSM sur la plateforme PicoSDR2X2

La dernière étape de l'implémentation de l'estimateur sur la plateforme SDR est la validation du PSM généré. Cette étape est réalisée par co-simulation. Les outils du MBDK fournis par Nutaq assurent cette fonction en permettant la création d'un modèle de co-simulation (PCM). Le PCM ainsi créé contient, en plus de tous les éléments Xilinx qu'on peut trouver dans un PIM, des éléments de contrôle spécifiques à la plateforme PicoSDR2X2. Ceci nous permet d'interagir avec la plateforme en contrôlant ses entrées/sorties ainsi que d'autres paramètres tels que ceux de la carte radio ou du PSM implémenté. On peut, ainsi, injecter des données à traiter et enregistrer les résultats directement sur l'ordinateur afin de les analyser et exploiter.

Le but de la validation du PSM étant de montrer que celui-ci donne les mêmes résultats que l'algorithme de l'estimateur précédemment développé avec MATLAB.
La validation du PSM se basera sur trois des fonctions que l'estimateur offre à savoir :

- L'estimation du SNR.
- la démodulation des symboles QPSK.
- l'identification du canal de transmission.

5.3.1 Validation de l'estimation du SNR

Afin de déterminer si le PSM implémenté sur la PicoSDR2X2 donne les mêmes résultats d'estimation du SNR que ceux théoriques, nous avons injecté au PSM les mêmes fenêtres d'observation obtenues avec MATLAB. Celles-ci sont constituées chacune de 112 symboles QPSK reçus $y_i(t_n)$ ayant passé à travers un canal de transmission de Rayleigh à SNR et étalement de Doppler paramétrables. Pour obtenir les résultats, nous avons effectué une simulation Monte-Carlo de 5000 réalisations.

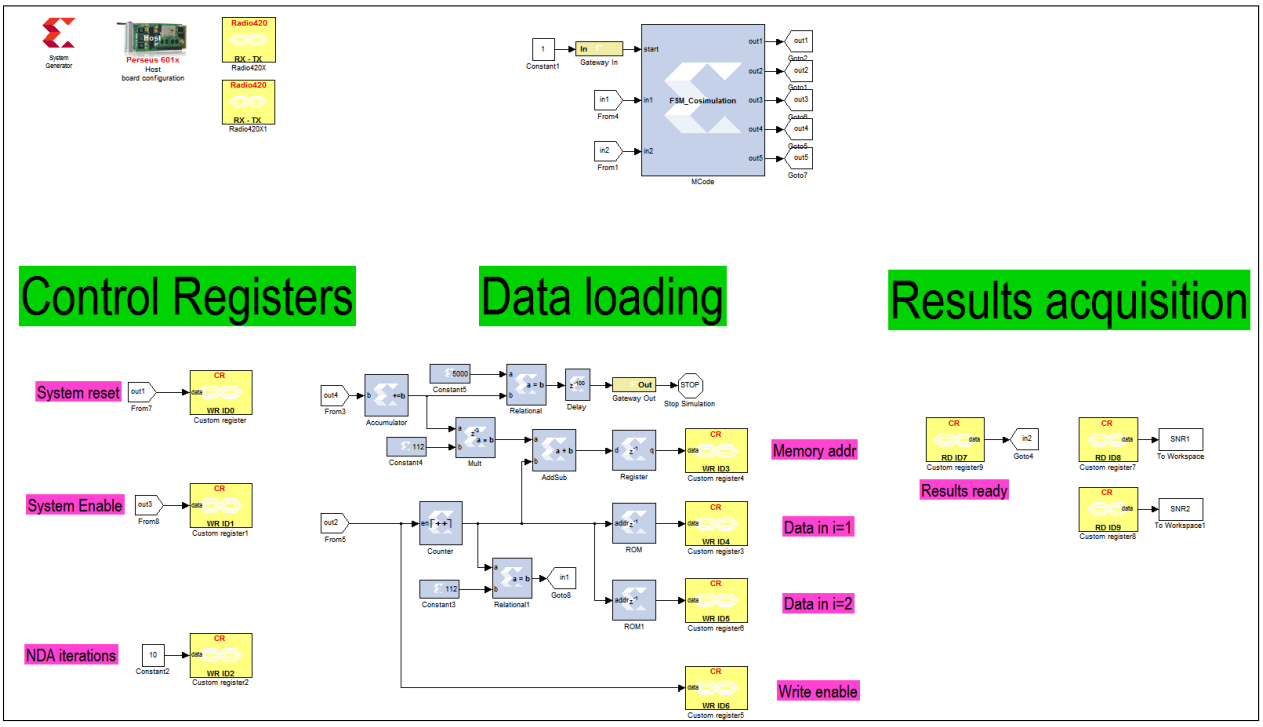


Figure 5.8 – PCM de la validation de l'estimation du SNR.

Le PCM créé, illustré par la Figure 5.8, assure ce traitement. Il est composé, entre autres, de registres de contrôle (registres jaunes). Ceux-ci permettent l'écriture/la lecture des données dans/de la plateforme PicoSDR2X2. Le PCM contient une MSA qui synchronise l'exécution et la succession des 5000 itérations.

Les étapes d'une itération de l'algorithme se déroule comme suit :

- Un signal de réinitialisation du système (System Reset) est envoyé vers la plateforme.
- Les 112 symboles QPSK $y_i(t_n)$ constituant la fenêtre d'observation de la $j^{ème}$ itération sur chaque antenne sont injectés dans les unités mémoires de la plateforme.
- Un signal de démarrage du système (System Enable) est envoyé vers la plateforme.
- Un signal indiquant la fin du traitement (Results Ready) est reçu de la plateforme. Les résultats du SNR estimé sont enregistrés sur l'ordinateur.
- Le compteur des itérations est augmenté de un. S'il atteint $j = 5000$, la simulation est arrêtée. Sinon, une autre itération est lancée.

Après le traitement des données enregistrées, nous avons produit les courbes de l'erreur moyenne quadratique normalisée (NMSE) en fonction du SNR pour différentes valeurs de la fréquence normalisée de l'étalement Doppler $F_D T_s$ et ceci pour les deux versions NDA et hybride de l'estimateur. Ces courbes sont montrées dans les figures ci-dessous.

Pour la version NDA:

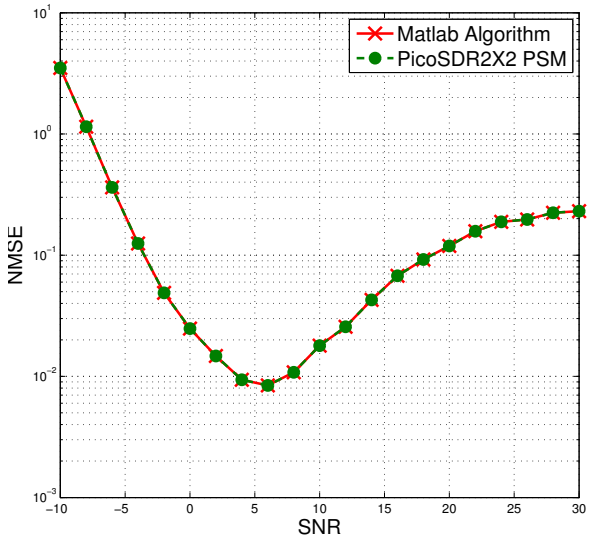


Figure 5.9 – NMSE de l'algorithme sur MATLAB de l'estimateur NDA comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK.

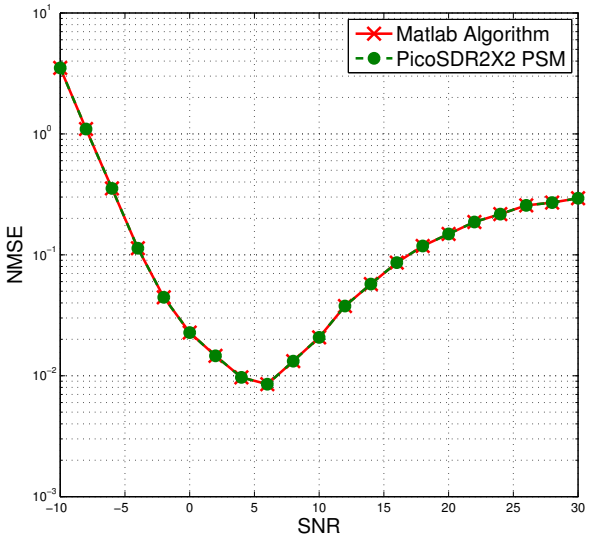


Figure 5.10 – NMSE de l'algorithme sur MATLAB de l'estimateur NDA comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK.

Pour la version hybride:

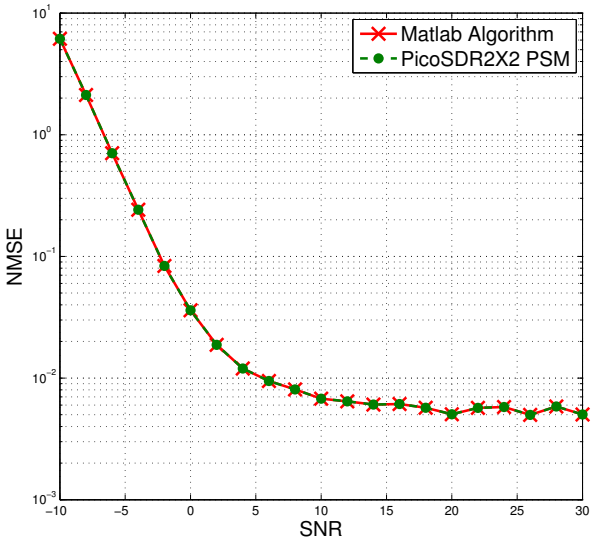


Figure 5.11 – NMSE de l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK.

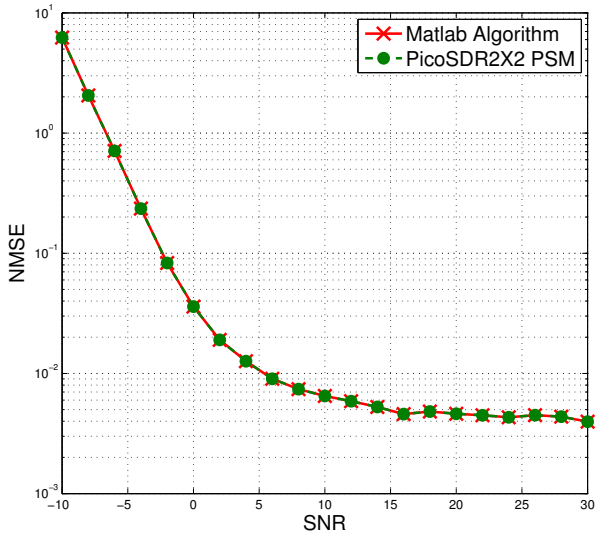


Figure 5.12 – NMSE de l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK.

Premièrement, nous constatons que les performances de l'estimateur sont nettement améliorées dans la version hybride, surtout au niveau des hauts SNRs: [5 dB. . . 30 dB]. Deuxièmement, les performances des deux versions restent valides et semblables pour les deux différentes valeurs de $F_D T_s$. Troisièmement, les deux courbes de la NMSE produites par MATLAB et produites sur la plateforme PicoSDR2X2 coïncident en tout point dans chaque figure. Ceci nous permet de conclure que le PSM se comporte exactement comme l'algorithme sur MATLAB et donne les mêmes résultats malgré la petite perte en précision due au passage de la virgule flottante à la virgule fixe. Ainsi, l'estimation du SNR est validée sur la plateforme PicoSDR2X2.

5.3.2 Validation de la démodulation des symboles QPSK

La validation de la démodulation des symboles QPSK se déroule selon le même protocole que celle de l'estimation du SNR. Au total, nous étudierons la démodulation de $N = 112 \times 5000 = 560000$ symboles QPSK, 112 étant le nombre de symboles par fenêtre d'observation et 5000 étant le nombre des réalisations de la simulation Monte-Carlo.

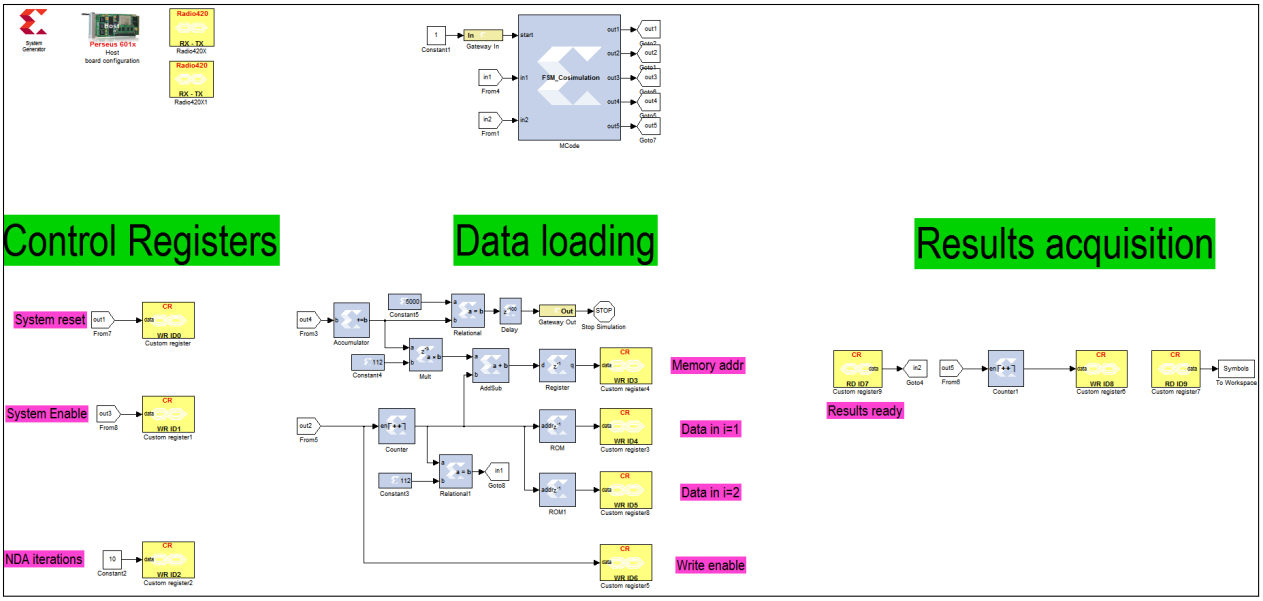


Figure 5.13 – PCM de la validation de la démodulation des symboles QPSK.

Le PCM, illustré par la Figure 5.13, est le même que celui de la validation de l'estimation du SNR à l'exception de l'acquisition des résultats. La différence réside dans le fait que nous devons lire les symboles démodulés d'une unité mémoire dans la plateforme. Pour cela, un compteur d'adresse est actionné quand les résultats sont prêts. Le contenu des cases mémoires est enregistré au fur et à mesure sur l'ordinateur.

Le taux d'erreurs binaires (BER) est ensuite calculé et les courbes du BER en fonction du SNR sont produites à différentes valeurs de $F_D T_s$ avec la version hybride. Les résultats sont montrés dans les Figures 4.14 et 4.15.

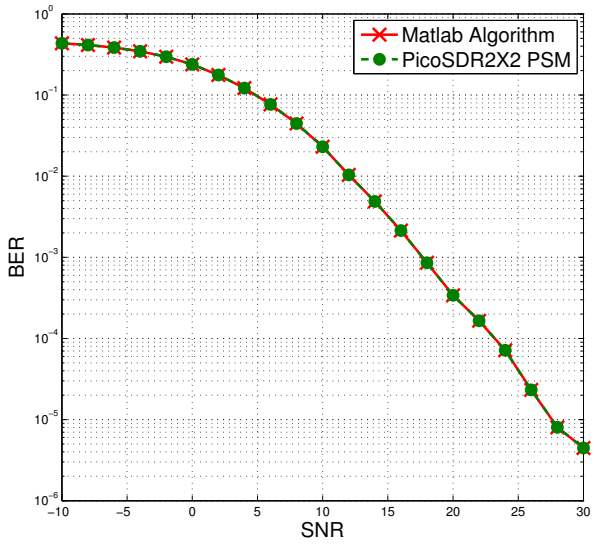


Figure 5.14 – BER de l'algorithme sur MATLAB de l'estimateur hybride comparé à celui du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 3.5 \times 10^{-3}$, QPSK.

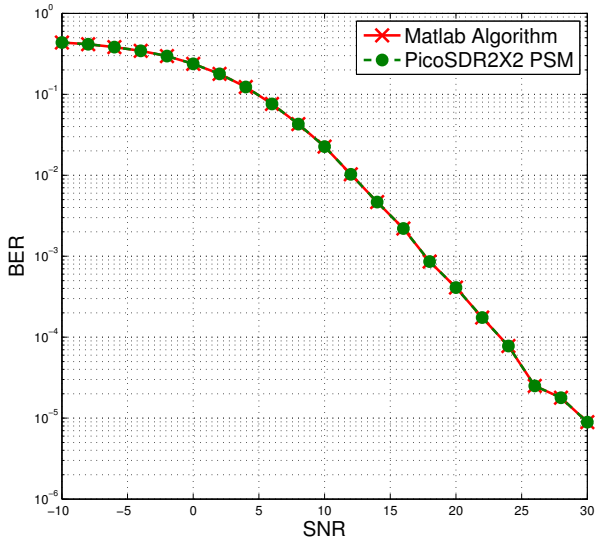


Figure 5.15 – BER de l'algorithme sur MATLAB de l'estimateur hybride comparé à celui du PSM du même estimateur sur PicoSDR2X2 en fonction du SNR avec $F_D T_s = 7 \times 10^{-3}$, QPSK.

De même que pour la validation du SNR, les performances de l'estimateur sont valides et semblables avec les deux différentes valeurs de $F_D T_s$. Plus important encore, les courbes produites sur MATLAB et sur la plateforme sont identiques. Ceci confirme que la démodulation QPSK faite par le PSM sur la PicoSDR2X2 est quasi-identique à celle faite avec MATLAB. Ce résultat est attendu étant donné le fait que l'estimation du SNR a été validée. En effet, cette dernière exploite la démodulation QPSK pour fonctionner.

5.3.3 Identification du canal de transmission

Dans le but d'identifier le canal de Rayleigh, dans un premier lieu, nous avons suivi le même mode opératoire que précédemment, mais sur $N = 112 \times 4 = 448$ symboles QPSK, *i.e.* quatre fenêtres d'observation. Dans la suite, nous avons émulé un canal de transmission en utilisant l'émulateur de canaux EB Propsim d'Anite que nous avons estimé sur le même nombre de fenêtres.

Canal de transmission créé avec MATLAB

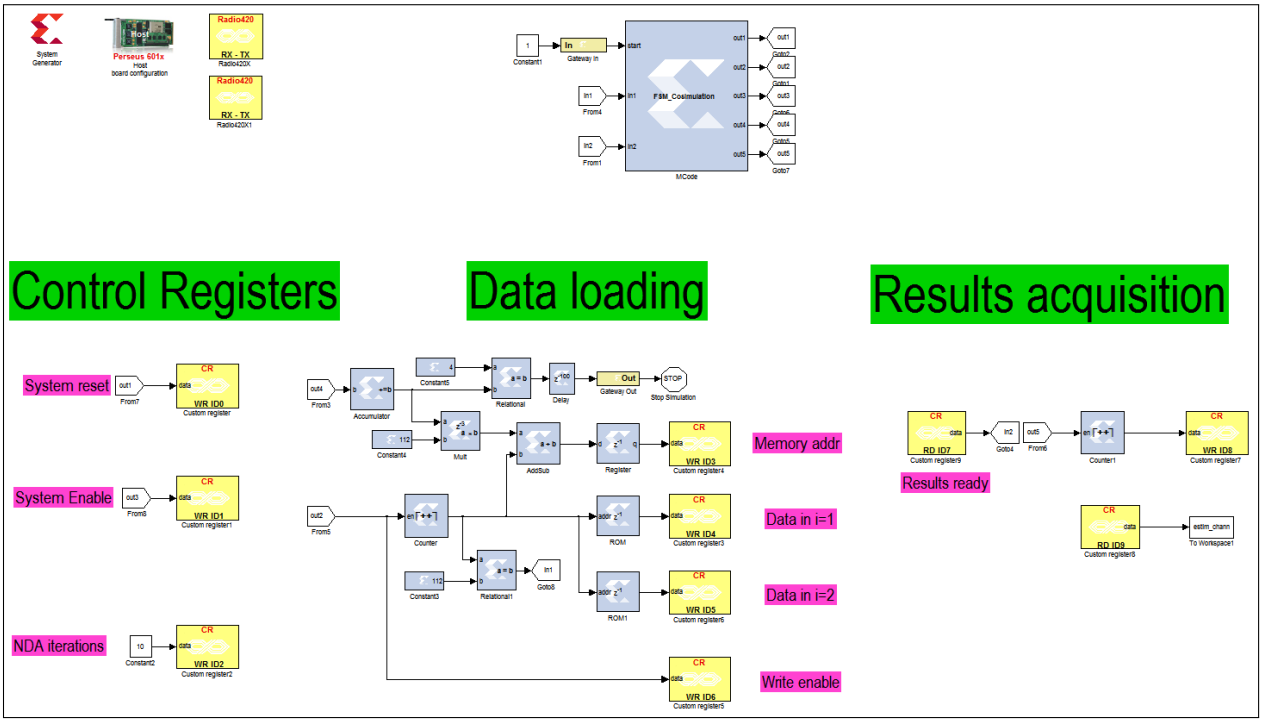


Figure 5.16 – PCM de la validation de du canal de transmission créé avec MATLAB.

Le PCM, illustré à la Figure 5.16, sert à injecter quatre fenêtres d'observation et à récupérer le canal estimé. Il suit le même protocole que les PCMs précédents.

Pour différentes valeurs du SNR, nous avons comparé le vrai canal produit par MATLAB avec celui estimé par l'algorithme de l'estimateur sur MATLAB et sur la PicoSDR2X2 avec une valeur de $F_D T_s$ fixée à 0.007. Nous nous sommes intéressés à une seule antenne i choisie arbitrairement vu que les deux canaux sont décorrélés.

Pour un SNR de 24 dB on obtient les figures ci-dessous:

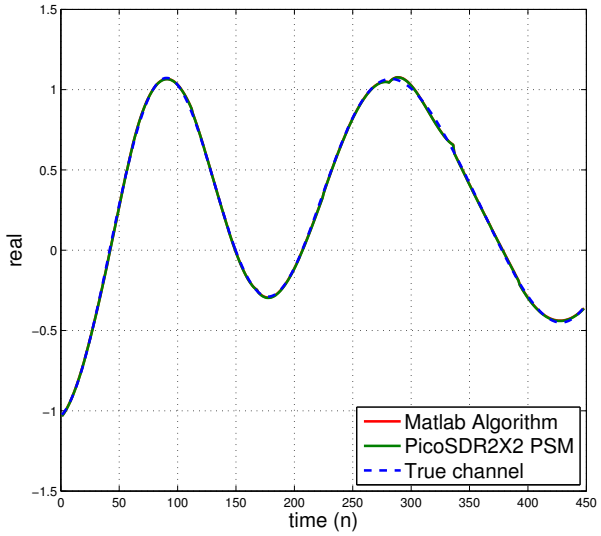


Figure 5.17 – Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

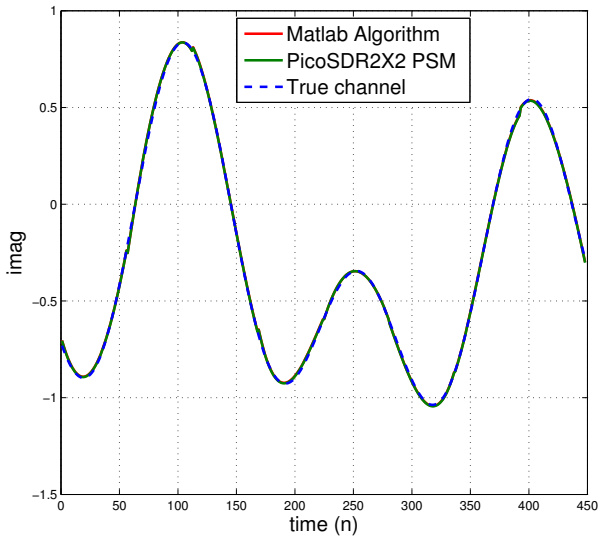


Figure 5.18 – Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

Ensuite avec un SNR de 14 dB:

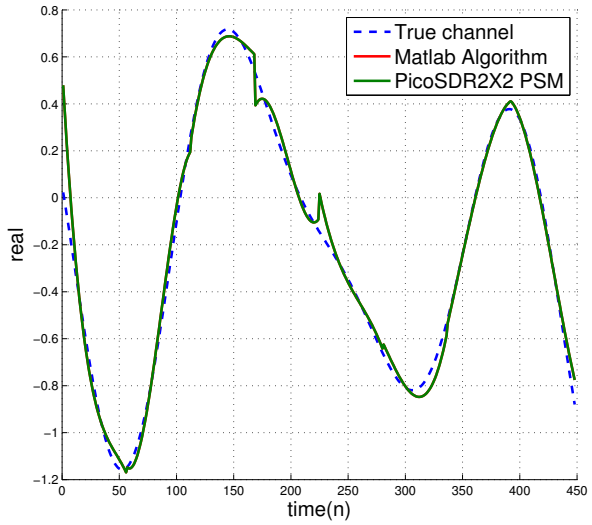


Figure 5.19 – Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

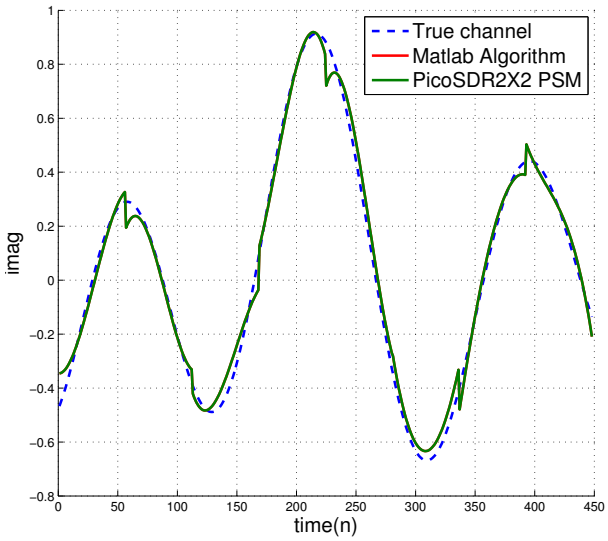


Figure 5.20 – Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

Et finalement avec un SNR de 10 dB:

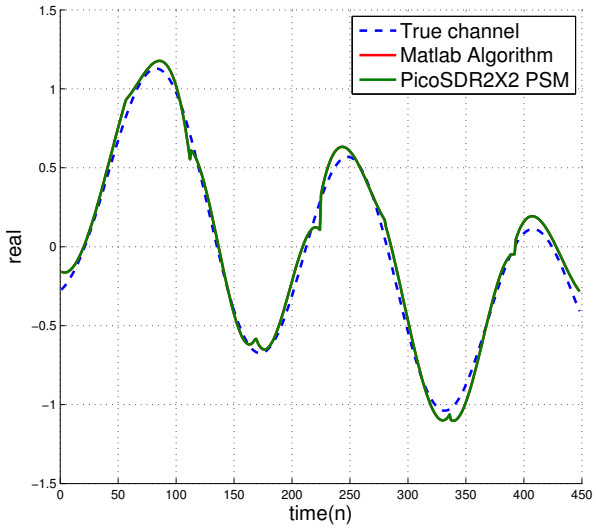


Figure 5.21 – Identification de la partie réelle du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

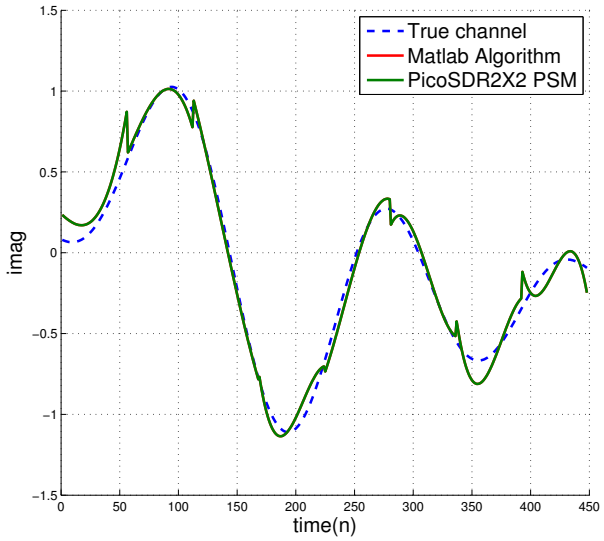


Figure 5.22 – Identification de la partie imaginaire du canal de MATLAB par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

La performance de l'estimateur dans l'identification du canal s'améliore, bien évidemment, avec l'amélioration du SNR jusqu'à épouser parfaitement la forme du vrai canal. L'identification sur

MATLAB et sur la PicoSDR2X2 restent identiques malgré une perte minime de précision relativement invisible. De ce fait, l'identification du canal est validée pour le PSM.

Canal de transmission créé avec EB Propsim

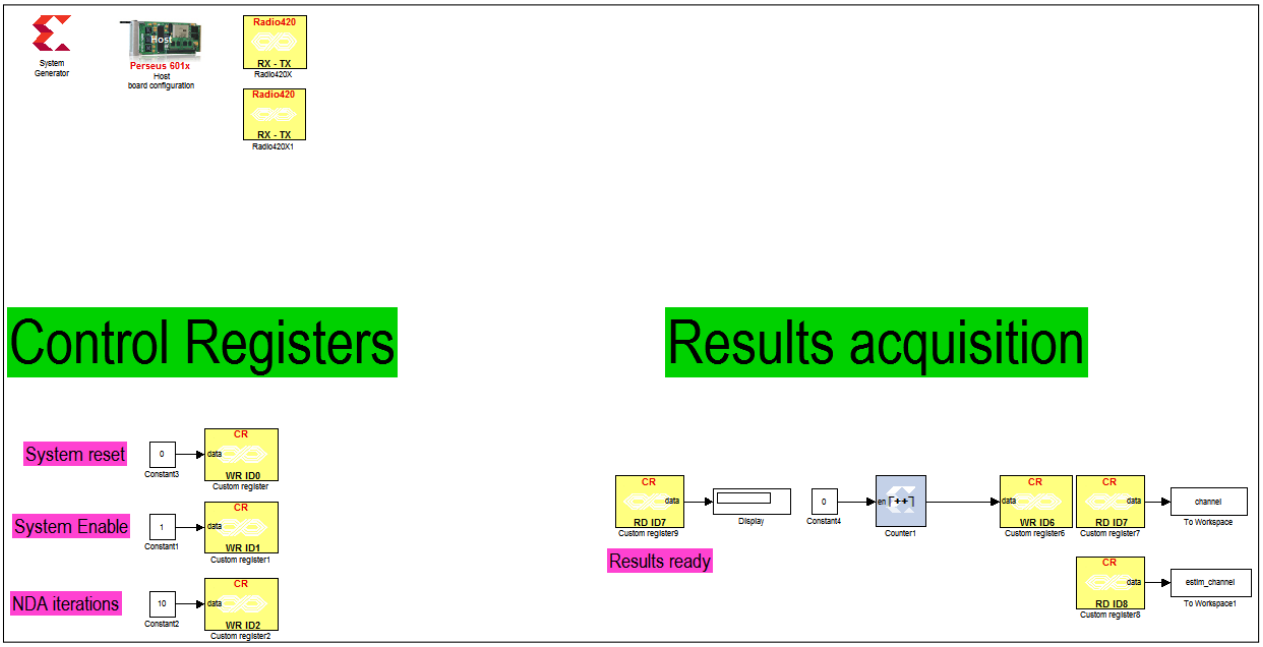


Figure 5.23 – PCM de l'identification du canal de transmission émulé avec EB Propsim.

Ce PCM, illustré à la Figure 5.23, est plus simple. Il ne contient pas de MSA et donc il est opéré manuellement. Après réinitialisation, le système est lancé. À ce moment, les symboles QPSK passent à travers l'émulateur du canal via l'émetteur-récepteur de la carte radio de la plateforme PicoSDR2X2 suivant une configuration SIMO 1X2. Quand le traitement de l'estimateur est fini et que le signal indique que les résultats sont prêts, le compteur d'adresse mémoire est activé et les résultats s'enregistrent sur l'ordinateur au fur et à mesure.

Après validation sur un canal de Rayleigh créé par MATLAB de la fonction d'identification de l'estimateur, l'étape suivante était d'identifier, en temps réel et sur les ondes, un canal de Rayleigh émulé par le EB Propsim d'Anite.

Pour cela, nous avons émulé un canal de Rayleigh avec $F_D T_s = 0.007$ pour différentes valeurs du SNR. L'émulateur est relié directement à la plateforme dans une configuration SIMO 1X2. Nous avons, à chaque fois, extrait le canal bruité à identifier en envoyant une séquence connue pour le

comparer à son estimé car l'émulateur n'offre pas la possibilité, comme avec MATLAB, d'avoir le vrai canal. Nous avons choisi les mêmes valeurs du SNR que celles des expériences faites avec le canal créé avec MATLAB.

Pour un SNR de 24 dB on obtient les figures ci-dessous :

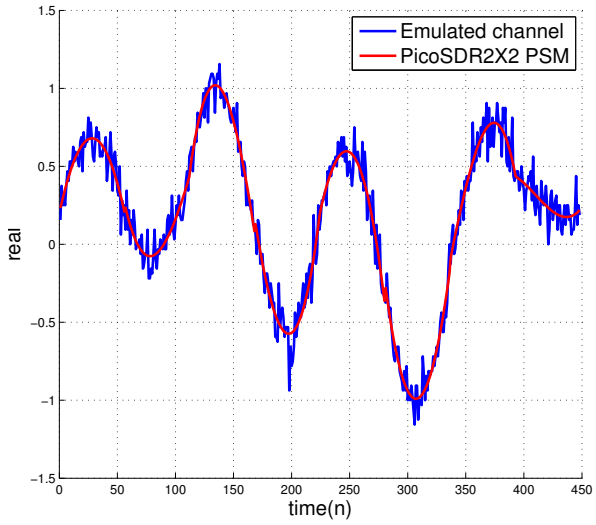


Figure 5.24 – Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

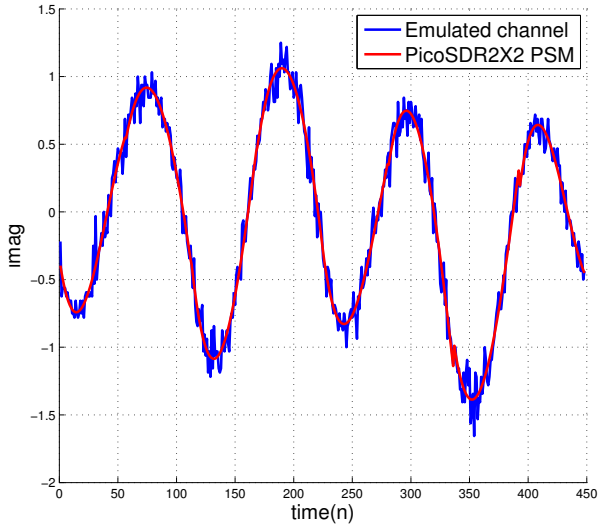


Figure 5.25 – Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $24dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

Ensuite avec un SNR de 14 dB:

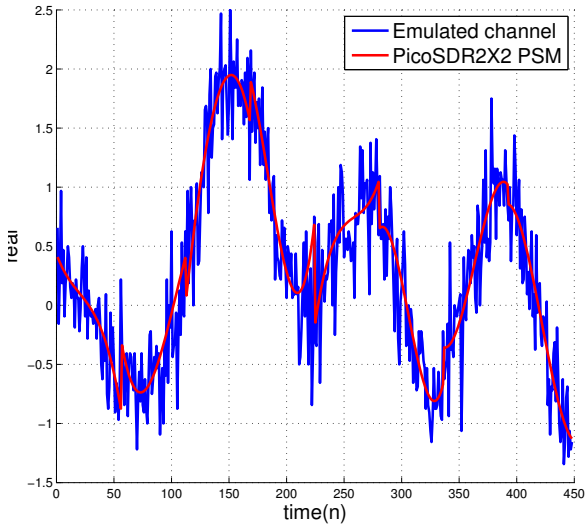


Figure 5.26 – Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

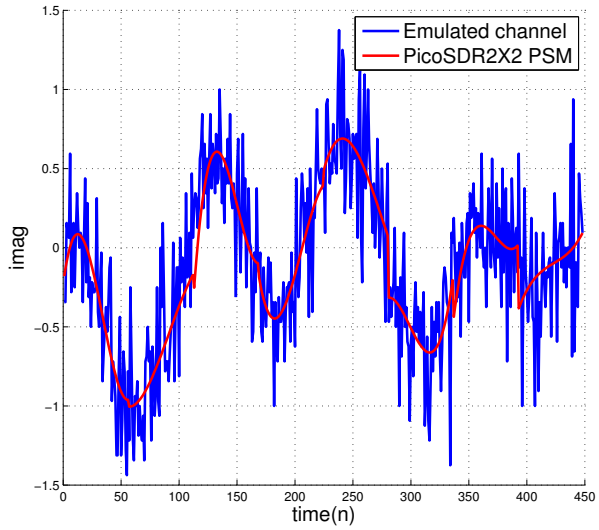


Figure 5.27 – Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $14dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

Et finalement avec un SNR de 10 dB:

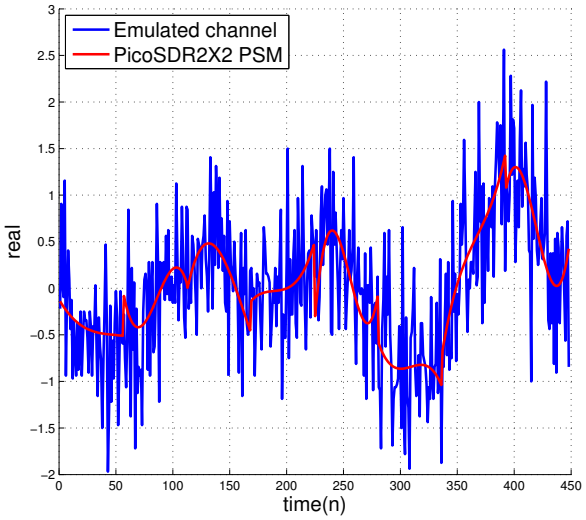


Figure 5.28 – Identification de la partie réelle du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

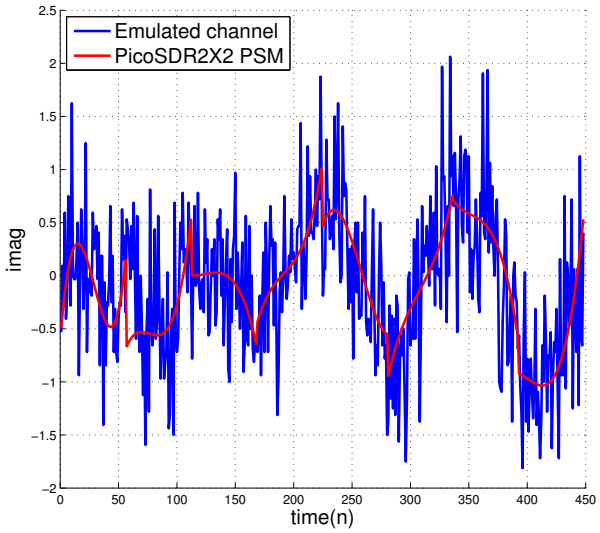


Figure 5.29 – Identification de la partie imaginaire du canal d'EB Propsim par l'algorithme sur MATLAB de l'estimateur hybride comparée à celle du PSM du même estimateur sur PicoSDR2X2 avec un SNR de $10dB$ et $F_D T_s = 7 \times 10^{-3}$, QPSK.

Nous remarquons une petite perte en performances dans les résultats des expériences avec l'émulateur EB Propsim d'Anite comparés à ceux obtenus avec le canal créé par MATLAB. Ceci s'ex-

plique, pour un même SNR, par le bruit ajouté par différents composants: câbles, connecteurs, etc., ainsi que par le bruit thermique de la plateforme et les performances des convertisseurs de la carte radio. Ceci étant, le canal estimé suit les variations du canal émulé. Bien entendu, l'identification est meilleure à des SNRs plus élevés.

5.4 Conclusion

Dans ce chapitre, après avoir présenté notre station de travail sur laquelle nous avons testé notre algorithme implémenté, nous avons pu valider et nous assurer du fait que ce dernier donnait les mêmes résultats que la version développée sur MATLAB et ceci en ce qui concerne l'estimation du SNR, la démodulation des symboles QPSK et l'identification du canal de transmission. Pour cette dernière fonction, nous avons pu évaluer les performances de l'estimateur implémenté dans des conditions pratiques sur les ondes d'un émulateur de canaux.

Chapitre 6

Conclusion générale

Le travail présenté dans ce mémoire consiste en l'implémentation et validation d'un estimateur du rapport signal sur bruit (SNR) sur une plateforme de radio logicielle. Cet estimateur de SNR à maximum de vraisemblance a été choisi car sa complexité algorithmique ne présentait pas d'obstacles à sa programmation matérielle, mais aussi et surtout grâce à ses hautes performances et son originalité. En effet, et comme mentionné dans les sections précédentes, il se base sur une nouvelle technique d'estimation ML pour estimer le SNR, démoduler les symboles reçus et identifier le canal de transmission et ceci pour les canaux SIMO variables dans le temps.

L'approche de programmation que nous avons choisie est celle basée sur un modèle. Cette approche nous a permis de faire un gain en temps de développement grâce aux outils proposés. Nous avons conçu et implémenté le design de l'estimateur dans l'environnement de travail logiciel de MATLAB/Simulink en utilisant les composants de la librairie Xilinx, la MBDK de Nutaq ainsi que System Generator pour générer l'exécutable sur la plateforme PicoSDR2X2 de Nutaq. Cette radio logicielle ainsi que l'émulateur de canaux EB Propsim d'Anite ont constitué notre environnement de travail expérimental.

Les résultats obtenus à l'issue des expériences sur le matériel ont permis de valider le PSM de l'estimateur dans toutes ses fonctions offertes et de rendre compte des performances de ce dernier. En effet, l'algorithme implémenté sur la PicoSDR2X2 a donné les mêmes résultats que celui développé sur MATLAB en dépit du passage à une représentation binaire à virgule fixe et des aléas du matériel dans les domaines bande de base et RF.

Dans ce travail, nous avons implémenté les trois versions de l'estimateur DA, NDA et hybride mais nous avons dû nous limiter dans le paramétrage de l'estimateur à travailler dans un domaine de fréquence Doppler normalisée $F_d T_s \leq 0.007$, à adopter une seule modulation QPSK, à travailler dans une configuration SIMO 1X2, etc. Une extension de ce projet sur laquelle nous travaillerons déjà vise une intégration matérielle plus ambitieuse qui prend en compte un plus large éventail de configurations. Aussi, nous sommes en train de développer une chaine de transmission OFDM sur laquelle nous grefferons cet estimateur afin de nous rapprocher davantage du standard LTE.

Références

[1] Matlab. Rapid prototyping. [Online]. Available: http://www.mathworks.com/rapid-prototyping

[2] Nutaq. Short history of the SDR technology. [Online]. Available: http://www.nutaq.com/blog/short-history-software-defined-radio-sdr-technology

[3] Muzammil, Rehan, et al. "Design and implementation of BPSK transmitter and receiver for Software Defined Radio on a Model Based Development Platform." *Industrial Electronics and Applications (ISIEA), 2011 IEEE Symposium on*. IEEE, 2011.

[4] Rodriguez, Anton, et al. "Model-based software-defined radio (SDR) design using FPGA." *Electro/Information Technology (EIT), 2011 IEEE International Conference on*. IEEE, 2011.

[5] Schwall, Michael, et al. "Model-based waveform design for the universal software radio peripheral with simulink." *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*. IEEE, 2011.

[6] Bhatnagar, Vishal, et al. "An fpga software defined radio platform with a high-level synthesis design flow." *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*. IEEE, 2013.

[7] Louet, Yves, and Al Ghouwayel, Ali. "FPGA implementation of a reconfigurable FFT for software radio multistandard systems." *International Conference on Consumer Electronics*. 2009.

[8] Ahmadian, Mansour, et al. "Model-based design and SDR", DSP enabled Radio, 2005, *the 2ndIEE/EURASIP Conference on* (Ref No. 2005/11086), pp 19/1-19/6.

[9] Zlydareva, Olga, and Sacchi, Claudio. "Multi-standard wimax/umts system framework based on sdr." *Aerospace Conference, 2008 IEEE*. IEEE, 2008.

[10] Alluri, Veerendra Bhargav, et al. "A new multichannel, coherent amplitude modulated, time-division multiplexed, software-defined radio receiver architecture, and field-programmable-gate-array technology implementation." *Signal Processing, IEEE Transactions on* 58.10 (2010): 5369-5384.

[11] Alluri, Veerendra Bhargav, et al. "A multi-channel coherent amplitude modulated time division multiplexed software defined radio receiver architecture for programmable-logic-device technology implementation." *Radio and Wireless Symposium, 2009. RWS'09*. IEEE. IEEE, 2009.

[12] ARTiSAN Software Tools, "Artisan real-time studio support for Model-Driven Architecture (MDA)," ARTiSAN Software Tools, Whitepaper, 2002. [Online]. Available: http://www.omg.org/mda/mda_files/ArtisanRealtimestudio.pdf

[13] Wakabayashi, Kazutoshi. "C-based behavioral synthesis and verification analysis on industrial design examples." *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*. IEEE Press, 2004.

[14] Cong, Jason, et al. "High-level synthesis for FPGAs: From prototyping to deployment." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 30.4 (2011): 473-491.

[15] Ku, D., and De Micheli, Giovanni. "HardwareC: A language for hardware design (version 2.0)," *Comput. Syst. Lab., Stanford Univ., Stanford, CA, Tech.* Rep. CSL-TR-90-419, 1990.

[16] Gajski, Daniel, et al. "SpecC: specification language and methodology" *Springer Science & Business Media*, 2012.

[17] Solutions, Agility Design. "Handel-C language reference manual. Agility Design Solutions." (2007).

[18] BlueSpec, Inc. [Online]. Available: http://www.bluespec.com

[19] Edwards, Stephen. "High-level synthesis from the synchronous language Esterel." *IWLS*. 2002.

[20] Haldar, Malay, et al. "A system for synthesizing optimized FPGA hardware from MATLAB." *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*. IEEE Press, 2001.

[21] Guo, Yuanbin, et al. "Rapid industrial prototyping and SoC design of 3G/4G wireless systems using an HLS methodology." *EURASIP Journal on Embedded Systems* 2006.1 (2006): 18-18.

[22] Sun, Yang, et al. "Scalable and low power LDPC decoder design using high level algorithmic synthesis." *SOC Conference, 2009. SOCC 2009*. IEEE International. IEEE, 2009.

[23] Hoffmann, Hans-Peter. "Harmony-SE/SysML Deskbook: Model-Based systems engineering with Rhapsody." (2006).

[24] MathWorks. (2015) Model-based design. [Online]. Available: http://www.mathworks.com/help/simulink/gs/model-based-design.html

[25] System Generator for DSP user guide, December 2, 2009 [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/sysgen_user.pdf

[26] Bellili, Faouzi, et al. "Maximum likelihood SNR estimation of linearly-modulated signals over time-varying flat-fading SIMO channels." *Signal Processing, IEEE Transactions on* 63.2 (2015): 441-456.

[27] Tingting, Gao, and Bin, Sun. "A high-speed railway mobile communication system based on LTE." *Electronics and Information Engineering (ICEIE), 2010 International Conference On*. Vol. 1. IEEE, 2010.

[28] Bellili, Faouzi, et al. "SNR estimation of QAM-modulated transmissions over time-varying SIMO channels." *Wireless Communication Systems. 2008. ISWCS'08. IEEE International Symposium on*. IEEE, 2008.

[29] Stéphenne, Alex, et al. "A decision-directed SNR estimator for QAM signals over time-varying SIMO channels." *Communications, 2008 24th Biennial Symposium on*. IEEE, 2008.

[30] Wiesel, Ami, et al. "SNR estimation in time-varying fading channels." *IEEE Transactions on Communications* 54.5 (2006): 841-848.

[31] PicoSDR-User's Guide-v1.0, Nutaq, June 2013.

[32] Nutaq, Nutaq Perseus 601X user guide. [Online]. Available: http://www.nutaq.com/wp-content/uploads/2015/12/Perseus601_04122015_web-datasheet.pdf

[33] Nutaq Radio420X, March 2014 [Online]. Available: http://www.nutaq.com/wp-content/uploads/2015/07/Radio420x_V1.3_03_24_2014_web.pdf

[34] Nutaq. Model-Based Design Kit (MBDK). [Online]. Available: http://nutaq.com/en/software/model-based-design-kit

[35] Propsim quick guide, Anite Telecoms, 2013. [Online]. Available: http://www.gigacomp.de/pdfs/Anite_Propsim_Quick_guide.pdf