وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختـــار- عنابـــــة

Faculté des Sciences de l'ingénieur
Département d'électronique

Année 2013

THÈSE

Présentée en vue de l'obtention du diplôme de **DOCTORAT**

IMPLEMENTATION SUR **FPGA** D'UN ALGORITHME DE DEBRUITAGE EN UTILISANT **1D-DWT**

Option
TRAITEMENT DE SIGNAL

Par
ABDELHAKIM SAHOUR

DIRECTEUR DE THÈSE : Mohamed BENOUARET M.C Université. Annaba

DEVANT LE JURY

PRESIDENT	**: H.A. ABBASSI**	**Professeur**	**Université .Annaba**
EXAMINATEURS	**: A.H. BOUKROUCHE**	**Professeur**	**Université .Guelma**
	: A.FAROUKI	**Professeur**	**Université .Constantine**
	: M. FEZARI	**M.C**	**Université .Annaba**
	: A.R. LACHOURI	**M.C**	**Université .Skikda**

REMERCIEMENT

J’exprime mes sincères remerciements à Monsieur le Professeur **H.A. ABBASSI,** pour avoir accepté de présider le jury de cette thèse.

Je tiens également à exprimer mes sincères remerciements à Monsieur les membres de jury pour l’intérêt qu’ils ont porté à ce travail en acceptant de le rapporter.

Cette thèse n’aurait pas vu le jour sans la patience et la générosité de mon directeur de thèse Monsieur **M. BENOUARET**, je veux le remercier chaleureusement d’avoir accepté de diriger cette thèse, de la confiance qu’il m’a accordée, de son encadrement et son suivi et ses conseils tout au long de ces années.

Sans oublier mes parents pour leurs *Daâouat* qui m’ont guidé pendant toute ma vie.

Une dédicace toute spéciale à ma femme **ISMAHANE** qui m’a entouré de son affection illimitée, de ses encouragements qui m’ont permis de garder mon optimisme et ma volonté dans les moments de stress, ainsi qu’à mes filles **LINA-DJIHANE**, **IMENE** et **RYM-NOUR**

Les membres de ma famille et ma belle famille qui m’ont comble de gentillesse.

SAHOUR Abdelhakim

RESUME

Le traitement du signal fait l'objet de recherche dans la plus part des laboratoires d'électronique, souvent depuis leurs premières années d'existence. Les travaux sont intensifiés avec l'apparition du traitement numérique du signal. Le débruitage est une méthode indispensable dans le traitement du signal. L'implémentation de la transformé en ondelettes ainsi que le choix de la fonction de seuillage ont été décortiquées et étudiées dans un souci de satisfaire les contraintes de l'application ciblée. Ces contraintes sont généralement le temps réel, latence ou débit des traitements, mais encore le coût de l'architecture dédiée ou la consommation du système mis en œuvre.

Quelle que soit la cible, FPGA ou DSP…. Il reste une phase de transformation sémantique difficile qui consiste à passer du type abstrait des variables manipulées par notre algorithme (variables réelles, complexes, entières ..etc) à un type logique comme, par exemple, le vecteur de bits qui sera admis pour le système RTL (register transfert logic).

Aujourd'hui, la maitrise des nouvelles technologies submicroniques, qui permet l'intégration à haute densité de dizaines de millions de transistors sur le même mono-chip, a induit l'apparition d'une nouvelle étape de conception qui repose sur la synthèse comportementale pouvant être assimilée à la conception de code source sur les dernières générations de DSP et FPGA. Elle consiste, à partir de la spécification comportementale d'un algorithme, à générer une représentation interne (Elément Logique) au niveau des Registres Logiques de Transfert (RTL). Le généré relève des techniques de compilation (analyse lexicale et syntaxique en VHDL, propagation de constants, etc…) Tandis que les transformations et les conversions reposent sur des méthodes comme l'ordonnancement et l'affectation ou l'assignement de composants en vue de satisfaire les contraintes de cette application, en particulier, le traitement en temps réel.

Les logiciels de programmation et de synthèse des FPGA offrent des outils conviviaux pour une implémentation facile. Pour une implémentation efficace l'utilisation des LUTs élimine la nécessité des multiplicateurs classiques longs, afin de réduire le temps d'exécution c.à.d. augmenter les performances en termes de fréquence (débit d'échantillonnage). Réduire la taille des LUTs gourmandes en termes d'éléments logiques, est une solution efficace pour remédier à ce type de problème, ce qui nous amène à optimiser l'utilisation des ressources matérielles.

MOTS CLES : Transformée en Ondelettes, DWT, Débruitage, Filtre de daubechies, Banc de filtres, VHDL, FPGA, Modelsim, Altera, DE2, Quartus, Matlab.

ABSTRACT

The signal processing is a research object in the most electronics laboratories, often since for their first years of existence. Work is intensified with the appearance of the digital signal. The denoising is an essential method in signal processing. The implementation of the discrete wavelet transforms (DWT), the choice of the thresholds and the functions of thresholding were peeled and studied in the purpose of satisfying the constraints of the targeted application. These constraints are generally the real time, latency or flow of the treatments, but still the cost of dedicated architecture or the cost of the implemented system.

Whatever the target is FPGAs or DSPs.... it remains a difficult semantic phase of transformation which consists of passing from the abstract type of the variables handled by our algorithm (real variables, complex, integer, etc…) to a logical type, such as, the bits vector admissible for RTL system (register logic transfer).

Today, the maitrise of new submicronic technologies, which allows high density integration of tens of millions of transistors on the same mono-chip, the appearance of a new stage of design based on the behavioral synthesis which can be assimilated to the design of source code on the last generations of DSP and FPGA. It consists, starting from the behavioral specification of an algorithm, in the generation of internal representation (Logic elements) at the level of Registers Transfer Logic (RTL). The generated code depends on the compilation techniques of (lexical analysis and syntactic in VHDL, constant propagation etc…) whereas that the transformations and conversions lie on the methods like scheduling and the assignment or task of components in order to satisfy the constraints of this application, in particular, real time processing.

The programming and synthesis software of the FPGA offer convivial tools for an easy and effective implementation. For an efficient implementation, the use of LUTs eliminate the necessity of slow classical multiplier, in order to decrease the execution time, i.e to increase the performance in terms of frequency (sampling throughput). Reducing the LUTs size, heavy consumer of logic elements, is an efficient solution to remediate at this type of problem. This leads to optimize the use of hardware resources

KEY WORD: wavelet transform, DWT, denoising, Daubechies filter, Filter banc, VHDL, FPGA, modelsim, Altera, DE2, Quartus, Matlab.

ملخـــص

تمثل معالجة الإشارة مواضيع بحث في معظم مختبرات الإلكترونية، منذ السنوات الأولى لوجودها. هذه الأعمال تكاثفت مع ظهور المعالجة الرقمية للإشارة، تطهير الإشارة عملية لا غنى عنها في معالجة ألإشارة، نصب تحويلات الموجيات و اختيار العتبة و دالة العتبة تم مناقشته و دراسته لغرض إرضاء متطلبات التطبيق المرغوب، هته المتطلبات هي على العموم الزمن الحقيقي، الكمون أو تدفق المعالجة، ولكن تكلفة البنية المخصصة أو استهلاك الطاقة للنظام.

مهما كانت المرمى FPGA أو DSP .... تبقى مرحلة تحويل صعبة تتمثل في الانتقال من المتغيرات المجردة المستعملة من قبل خوارزيتمنا (المتغيرات الحقيقية، المركبة، الكلي .. الخ) إلى نوع منطقي، على سبيل المثال، شعاع الخانات الذي يتم قبوله من طرف نظام RTL (سجل تحويل منطقي).

اليوم، التمكن من التكنولوجيات الحديثة ل submicron، والذي يسمح بكثافة تكامل عالية لعشرات الملايين من الترانزستورات على نفس الرقاقة الأحادية، بفعل ظهور مرحلة تصميم جديدة تقوم على شمللة السلوك لكي يمكن استيعابهم في تصميم الدارات المبرمجية على أحدث جيل من DSP وFPGA. إنه يتألف، من مواصفات السلوكية للخوارزمية، لإنشاء التمثيل الداخلي (العنصر المنطقي) على مستوى سجلات التحويل المنطقية (RTL). الشفرة المنشاءة من تقنيات التجميع (تحليل المعجمية والنحوية لـ VHDL، نشر الثوابت، الخ ...) في حين تستند التحولات والتحويلات على الطرق مثل الجدولة والتوزيع تعتمد من اجل تلبية تنقضات التطبيق، على وجه الخصوص، المعالجة في الوقت الحقيقي.

البرامج وأدوات التجميع FPGA تقدم تنفيذ سهل الاستعمال. من اجل تنفيذ فعال لاستخدام LUTs يلغي الحاجة للمضاعفات التقليدية البطيئة، من اجل التقليل من وقت التنفيذ و زيادة أداء من حيث التردد. تقليل حجم LUT ت المستهلكة للعناصر المنطقية، حل فعال لحل هذا النوع من المشاكل، وهو ما يقودنا إلى تحقيق الاستخدام الأمثل للموارد المادية.

كالمات مفتاح : التطهير، الوافلات، مصفاة دوب شيز، مجموعة مصفاة ، VHDL ، FPGA, Modelsim, Altera, DE2, Quartus, Matlab.

,

Listes des figures **I**

Notations **V**

CHAPITRE I : INTRODUCTION GENERALE

1. Introduction générale **1**

CHAPITRE II : CIRCUITS LOGIQUES PROGRAMMABLES

2.1 Introduction : **6**
2.2 Histoire de la logique programmable **7**
2.3 Les FPGAs **11**
2.4 Architecture des FPGAs **14**
2.5 Fournisseurs des circuits logiques programmables **16**
2.6 Outils de Développement **16**
2.6.1 Description de l'application **17**
2.6.1.1 Outils de saisie d'un schéma **17**
2.6.1.2 Les langages de description de matériel **17**
a. Le Verilog : historique et description **19**
b. VHDL : historique et description **19**
c. Structure d'une description VHDL **21**
2.6.2 Traduction et optimisation **22**
2.6.3 Partitionnement placement et routage **23**
2.6.4 Simulation **24**
2.6.5 Configuration **24**
2.7 Les atouts des FPGA **25**
2.7.1 Performances **25**
2.7.2 Temps de mise sur le marché **25**
2.7.3 Cout **25**
2.7.4 Fiabilité **26**
2.7.5 Maintenance **26**
2.8 Les tendances courantes **26**
2.9 Le choix des FPGAs **27**
2.10 La carte de développement **28**
2.10.1 Présentation **28**

2.10.2 CODEC AUDIO 29

CHAPITRE III : ARITHMETIQUE SUR FPGA

3.1 Introduction 31
3.2 Représentation des nombres 31
3.2.3 Les nombres en virgule fixe 32
3.2.3.1 Représentation des nombres non signé (positifs) 32
3.2.3.2 Représentation des nombres signés 32
3.2.3.3 Représentations biaisées 33
3.2.3.4 Complément à deux (2C) 34
3.2.3.5 Complément à un (1C) 35
3.2.3.6 un-conventionnel en virgule fixe 36
3.2.3.7 Nombres à Chiffre Signé 37
3.2.3.8 Le code (CSD) Fractionnel 37
3.2.4 les nombres en virgule flottante 38
3.3 Operateurs arithmétiques spécifiques pour les FPGAs 39
3.3.1 Introduction 40
3.3.2 Addition binaire 40
3.3.2.1 Semi additionneur 40
3.3.2.2 Additionneur complet 41
3.3.2.3 Additionneur par propagation de retenue 41
3.3.2.4 Additionneur par anticipation de retenue 42
3.3.2.5 Additionneur par l'utilisation d'une LUT 43
3.3.2.6 L'addition et la soustraction en complément à deux 44
3.3.3 La multiplication 45
3.3.3.1 La multiplication graphique 45
3.3.3.2 La multiplication d'accumulateur (MAC) et somme de produit (SOP) 47
3.3.3.3 La multiplication distribuée 48
3.3.3.4 Look-Up Table (LUT) Multiplication 51

CHAPITRE IV : THEORIES DE LA TRANSFORMEE EN ONDELETTE

4.1 Introduction 54
4.2 Transformée en ondelette 55
4.3 Définition d'une ondelette 57

4.4 La transformée discrète en ondelette 58
4.5 L'analyse multi résolution (MRA) 59
4.6 L'algorithme pyramide de Mallat 61
4.7 Propriété des ondelettes 62
4.7.1 Support compact 62
4.7.2 Symétrie 62
4.7.3 Nombre de moment nuls 62
4.7.4 Régularité 63
4.8 Panorama d'ondelettes 63
4.8.1 Base d'ondelettes de daubechies 64
4.9 Application 66
4.10 Banc de filtre 67
4.10.1 Introduction 67
4.10.2 Conversion de la fréquence 68
4.10.2.1 Sous échantillonnage 68
4.10.2.2 Sur échantillonnage 69
4.10.3 La décimation 69
4.10.4 L'interpolation 69
4.10.5 Les identités nobles 70
4.10.6 Utilisation du Bancs de filtre 72
4.11 Débruitage de signaux 73
4.11.1 Introduction 73
4.11.2 Position du problème 73
4.11.3 Débruitage par des filtres linéaires 74
4.11.4 Débruitage par les ondelettes 74
4.11.5 Débruitage par seuillage 75
4.11.6 Méthode de seuillage 76
4.11.6.1 Seuillages dur 77
4.11.6.2 Seuillage doux 77
4.11.6.3 Seuillage dur modifie 77
4.11.6.4 Sélection du seuille 78

CHAPITRE V : DEBRUITAGE DU SIGNAL PRINCIPE & IMPLEMENTATION

5.1 Introduction 80

5.2 Design méthodologie de développement **80**
5.2.1 ADC/DAC CODEC AUDIO **81**
5.2.2 Module d'horloge **82**
5.2.3 L'opération de débruitage **83**
5.2.4 Principe de débruitage avec la transforme d'ondelettes **83**
5.2.4.1 Choix des ondelettes et niveau de décomposition **84**
5.2.4.2 Banc de filtres d'analyse (décomposition) **86**
5.2.4.3 Banc de filtre de synthèse (reconstitution) **90**
5.2.4.4 Le seuillage **93**
5.2.4.5 Filtre décimateur de daubechies **93**
5.2.5 Implémentation de l'opération de débruitage **94**
5.2.5.1 La transformée en ondelettes discrète **94**
5.2.5.2 Analyse multi-résolution **95**
5.2.5.3 La multiplication par une constante **96**
5.2.5.4 Représentation binaire des nombres décimaux **101**
5.2.5.5 Un niveau de décomposition **101**
5.2.5.6 Le seuillage **107**
5.2.5.7 Un niveau de reconstitution **110**
5.2.5.8 Filtre moyenneur **115**
5.2.6 Simulation **116**
5.2.7 Implémentation hardware **118**
5.3 Conclusion **119**

CONCLUSION ET PERSPECTIVES

1C	One’s complement
2C	Two’s complement
ADCs	analogue-to-digital converter
ALM	Altera Logic Module
ASG	Analysis Standardisation Group
ASIC	Application Specific Integrated Circuit
BDTI	Berkeley Design Technology Inc
BGA	Ball Grid Array
CD	Compact Disc
CLBs	configurable logic blocks
CPA	Carry-Propagate Adder
CPLD	Complex Programmable Logic Device
CSD	Canonic Signed Digit System
CSD	canonic signed digit
CWT	Continuous Wavelet Transform
DA	Distributed Arithmetic
DACs	digital-to-analogue converter
DSP	Digital Signal Processor
DWT	Discrete Wavelet Transform
EEPROM	Electrically
EPROM	erasable PROM
FA	full-adder
GPS	Global Positioning system
HA	Half-Adder
HDL	Hardware Description Language
HDL	Hardware Description Language
I/O	Input/Output
I²C	Inter Integrated Circuit
iid	indépendant et identiquement distribué
IP	Intellectual Property
IrDA	Infrared Data Association
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display

LE	Logic Element
LED	light-emitting diode
LSB	Least significant bit
LUT	Look up table
MAC	Multiply-Accumulator
MMI	Monolithic Memories Inc
MRA	Multi-Resolution Analysis
MSB	Most significant bit
NRE	Non-Recurring Engineering
OFDM	Orthogonal Frequency Division Multiplexing
OLMC	Output Logic Macro Cell
OT	Objets Techniques
OVI	Open Verilog International
P.L.D	Programmable Logic Device
PCI	Personel Computer Interface
PDA	Personal Digital Assistants
PLA	Programmable Logic Array
PLI	Progamming Language Interface
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTL	Register Transfer Level
SD	Signed Digit
SNR	Rapport signal sur bruit
SoC	System on Chip
SOP	Sum Of Product
SPLD	Simple Programmable Logic Devices
SRAM	Static Random Access Memory
UAL	Unite Arithmetic Logic
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHDL-AMS	VHDL Analog and Mixed Signal
VHSIC	Very High Speed Integrated Circuit

Figure2.1 Les déférentes catégories des circuits intégrées numériques
Figure2.2 La structure d'un PLA avant la programmation
Figure2.3 un PLA réalisant un OU exclusif
Figure2.4 Structure de base d'un PLD
Figure2.5 Schéma d'un CPLD
Figure2.6 physionomie d'un FPGA
Figure2.7 Un élément logique de l'FPGA
Figure2.8 Exemple de LUT
Figure2.9 Exemple de LUT ALTERA Cyclone II
Figure2.10 Une ALM de stratix III
Figure2.11 Virtex-II Pro Slice (Top Half)
Figure2.12 Cellule SRAM
Figure2.13 Résumé graphique des familles de P.L.D
Figure2.14 Les étapes du développement pour les PLD
Figure2.15 Schéma de l'additionneur incomplet
Figure2.16 L'arbre des séries d'Altera
Figure2.17 La carte DE2
Figure2.18 Le Bloc diagramme du WM8731
Figure 3.1 Aperçu des représentations de nombres
Figure 3.2 Représentation en signe et valeur absolue de nombres de quatre bits.
Figure 3.3 Représentation biaisée (R = 23) de nombres entre -8 et +7, avec des mots de quatre bits.
Figure 3.4 Représentation en complément à deux de nombres entre -8 et +7, avec des mots de quatre bits.
Figure 3.5 Représentation en complément à un de nombres entre -7 et +7, avec des mots de quatre bits.
Figure 3.6 Résumé des diverses représentations des nombres étudiées.
Figure 3.7 résultats de simulation.
Figure 3.8 Schéma et symbole d'un Semi Additionneur (half-adder)
Figure 3.9 Schéma et symbole d'un Additionneur Complet (Full-adder
Figure 3.10 La structure d'un additionneur à retenue propagé
Figure 3.11 la structure d'une cellule S, G et P.
Figure 3.12 structure d'un Additionneur avec une LUT

Figure 3.13 Soustraction en complément à deux réalisée avec un additionneur à retenue propagée
Figure 3.14 Multiplication traditionnelle de deux nombres
Figure 3.15 Multiplicateur basé sur l'algorithme de décalage et d'addition
Figure 3.16 Les deux réalisations pour la constante 124.
Figure 3.17 Multiplicateur avec un coefficient 124
Figure 3.18 Structure d'une multiplication distribue
Figure 3.19 Structure d'une multiplication distribue à base d'une LUT
Figure 3.20 Schéma bloc d'une LUT (Entrée (5 bits) * 67, sortie(15 bits))
Figure 3.21 Schéma bloc d'un exemple d'une LUT (Entrée = 25, sortie = 1675)
Figure 4.1 Analyse temps-échelle par la transformée en ondelette
Figure 4.2 L'onde sinusoïdale est symétrique et régulière et l'ondelette est une vibration asymétrique et irrégulière
Figure 4. 3 L'évolution d'une Ondelette
Figure 4. 3 Décomposition en banc de filtre d'une analyse multirésolution
Figure 4. 4 Type d'ondelettes
Figure 4. 6 Ondelette de Daubechies à 2 moments nuls (4 coefficients)
Figure 4. 7 Ondelette de Daubechies à 6 moments nuls (12 coefficients) pour 2 facteurs d'échelle différents.
Figure 4. 8 L'application de la transformée en ondelettes dans le traitement du signal
Figure 4. 5 Le sous-échantillonnage
Figure 4. 6 Le sur-échantillonnage
Figure 4. 7 La décimation
Figure 4. 8 l'interpolation
Figure 4.9 Les identités nobles
Figure 4.14 La structure polyphasé
Figure 4.15 La structure polyphase efficace pour la décimation
Figure 4.16 Le décimateur polyphase avec un interrupteur à l'entrée.
Figure 4.17 La structure polyphase efficace pour l'interpolation
Figure 4.18 L'interpolateur polyphase avec un interrupteur à la sortie
Figure 4.19 La structure d'un banc de filtre standard
Figure 4.20 Seuillages dur et doux.
Figure 5.1 Bloc diagramme des différentes étapes

Figure 5.2 Les protocoles des lignes de bus I2C de WM8731 Wolfson
Figure 5.3 Résultas de simulation avec modelsims; (a) signal d'horloge 50MHz; (b) signal d'horlogel 'XCK' ; (c) signael ADC LR; (d) signal Bit-Stream Clock
Figure 5.4 Schéma bloc de la fonction débruitage
Figure 5.5 Banc de filtre à deux canaux
Figure 5.6 décomposition à deux canaux d'un banc de filtre
Figure 5.7 Banc de filtre de décomposition à deux canaux
Figure 5.8 Décomposition polyphase de $H_a(Z)$ et $G_a(Z)$
Figure 5.9 la structure polyphase d'analyse d'un banc de filtres
Figure 5.10 Banc de filtre de reconstitution
Figure 5.11 la structure polyphase de $H_s(z)$ et $G_s(Z)$
Figure 5.12 banc de filtre polyphase à deux canaux de la phase de synthèse
Figure 5.13 Filtre daubechies sous la structure polyphase
Figure 5.14 L'évolution de la structure polyphase du filtre de daubechies
Figure 5.15 Les trois niveaux de décomposition
Figure 5.16 les trois niveaux de synthèse (reconstitution)
Figure 5.17 Schéma bloc de la fonction débruitage
Figure 5.18 l'implémentation d'une LUT en utilisant le concept KCM
Figure 5.19 l'implémentation d'une LUT en utilisant le concept KCM
Figure 5.20 Résultat de la simulation par Quartus ® II de la LUT du coefficient 0.8365
Figure 5.21 Architecture RTL donnée par Quartus ® II du multiplicateur par le coefficient 0.8365
Figure 5.22 L'Architecture d'un niveau décomposition
Figure 5.23 Schéma bloc du diviseur
Figure 5.24 Machine à états finis et programme VHDL du Diviseur
Figure 5.25 Le résultat de la simulation par Modelsim du diviseur
Figure 5.26 Vue interne (RTL) du diviseur.
Figure 5.27 Vue interne (RTL) des deux filtres décomposition
Figure 5.28 Le résultat de la simulation par Quartus ® II d'un niveau de décomposition
Figure 5.29 Simulation par ModelSim d'un niveau décomposition
Figure 5.30 Architecture du composant de seuillage
Figure 5.31 Vue interne (RTL) du composant de seuillage
Figure 5.32 Simulation par ModelSim de l'architecture de seuillage

Figure 5.33 L’architecture d’un niveau de reconstitution
Figure 5.34 L’Architecture de l’interpolateur
Figure 5.35 Vue interne (RTL) de l’interpolateur.
Figure 5.36 Digramme de la procédure d’interpolation
Figure 5.37 Vue interne (RTL) des deux filtres reconstitution
Figure 5.38 Simulation par ModelSim de l’architecture d’un niveau de reconstitution
Figure 5.39 Schéma bloc de la phase de décomposition
Figure 5.40 Vue interne (RTL) d’un exemple de bloc de retard
Figure 5.41 L’Architecture du filtre moyenneur
Figure 5.42 Vue interne (RTL) du filtre moyenneur
Figure 5.43 Simulation par ModelSim du filtre
Figure 5.44 Le résultat de la simulation avec ModelSim; (a) signal original ; (b) signal original retardé; (c) signal débruité; (d) signal débruité et filtré.
Tableau 2.1 Table de vérité d’un additionneur incomplet
Tableau 4. 1 Les pseudo-fréquences et les échelles correspondantes
Tableau 5.1 Coefficients des différents filtres de daubechies
Tableau 5.2 résultats Matlab d’un niveau de décomposition
Tableau 5.3 Hardware Resource Consumption

Chapitre I

Introduction Générale

1 Introduction :

Tous les signaux obtenus en tant qu'une réponse instrumentale sont affectés par un bruit. Le bruit dégrade l'exactitude et la précision de l'analyse. Le débruitage du signal est donc fortement souhaitable et indispensable en même temps dans l'optimisation analytique de la réponse instrumentale. Pour les applications à grand intérêt, dans les télécommunications et le multimédia : téléphones mobiles, PDA (Personal Digital Assistants), systèmes GPS (Global Positioning system), jeux et lecteurs vidéo… le bruit est principalement à haute fréquence, alors que le signal effectif est principalement de basse fréquence. Ainsi compte tenu que la décomposition en ondelettes décompose le signal d'une manière ordonnée en coefficient d'approximation (de basse fréquence) et les coefficients de détail (haute fréquence), les coefficients de détail contiendront une grande partie du bruit. Ceci suggère une méthode pour débruiter le signal. Qui consiste à réduire simplement la taille des coefficients de détail avant de les employer pour reconstruire le signal. Cette approche s'appelle seuillage (thresholding) ou rétrécissement des coefficients de détail. Naturellement, nous ne pouvons pas éliminer entièrement les coefficients de détail; car ils contiennent toujours quelques composantes importantes du signal informatif original. Nous avons proposé différents types de seuillage. Il est à noter que le type de seuillage dépend de l'application. Les deux différentes approches qui sont habituellement appliquées au débruitage à seuillage sont le seuillage dur et le seuillage doux. La méthode de seuillage dure consiste en plaçant tous les coefficients de l'ondelette au-dessous d'une valeur du seuil donnée égale à zéro, alors que dans le seuillage doux, les coefficients de la transformée d'ondelette sont réduits par une quantité égale à la valeur du seuil [1].

Le Débruitage de signal en utilisant la DWT comprend les trois étapes suivantes :

- la transformée d'ondelette du signal observé qui consiste à la décomposition du signal en deux classes les détails et les approximations.
- des coefficients empiriques de d'ondelette en moyen de l'opération de seuillage.
- l'ondelette inverse des coefficients modifiés. en vue de restituer l'information utile ayant subi efficacement l'opération de débruitage.

La recherche précédente sur le Débruitage du signal en utilisant la décomposition en ondelette est de nature off-line; ce qui signifie que le signal est prélevé en temps réel, puis stocker dans la mémoire ou sur le disque dur, et il sera débruité ou traiter plus tard sur un

ordinateur à l'aide d'un outil software tel que le logiciel Matlab. Cette approche à l'avantage d'être facile à la programmation et un temps de réalisation réduit. Par contre à un temps d'exécution lent et un coût minimum (CPU-RAM-E/S). Cependant, beaucoup d'applications exigent le traitement en temps réel du signal, dans lequel le signal doit être traité juste à son arrivé. Ces applications en temps réel exigent que le signal soit traité au même taux qu'il est produit ; en d'autres termes, la sortie des échantillons du système de débruitage doit être égale aux données entrant dans le système. Avec un peu de retard, ce qu'est acceptable (et nécessaire, puisque les calculs ne peuvent pas être faits instantanément). C'est le cas des ASICs qui offrent l'avantage d'avoir une exécution très rapide, et une consommation optimisée. Le but de cette thèse est de démontrer que le débruitage de signal peut être fait efficacement et économiquement en temps réel, en utilisant un circuit programmable de type FPGA. Au contraire des ASICs pour des raisons de coût lié à la fabrication des masques, le taux d'erreurs et le manque de flexibilité avec un temps de développement important, les architectures à base de FPGAs (Field Programmable Gate Arrays) deviennent à présent des alternatives possibles. Ces circuits inventés par la société Xilinx au milieu des années 80, selon les prévisions, le marché mondial du FPGA devrait passer d'1.9 Milliard de dollars en 2005 à 5.6 milliards en 2014 [2,3].

Les FPGAs ont connu d'importantes évolutions architecturales. Résultant essentiellement de l'augmentation des capacités d'intégration. Les principaux atouts de la technologie FPGA sont :

- Exécution très rapide ;
- Facilité et flexibilité d'utilisation ;
- Adapté au prototypage rapide ;
- Adaptabilité aux futures évolutions grâce à la possibilité de reconfiguration dynamique du circuit ;
- Fabrication économique ;
- Maintenance à long terme ;

- Facilité de test.

2 L’état de l’art :

Dans les dernières décades d’année, les travaux ont été intensifié sur l'implémentation de la transformées en ondelettes discrètes (DWT) sur la base des processeurs de traitements numériques des signaux (DSP), des circuits logique programmable de type FPGA. Les premiers circuits spécifiques utilisés pour le calculer la DWT étaient les DSP, qui ont un processus de grande puissance de calcul, à une vitesse élevée et, normalement, s'est avéré en premier temps une solution parfaite. Les DSPs contient des unités spéciaux, tels que Multiplicateur à Accumulateur (MAC) pour améliorer les performances de l’implémentation de la transformée en ondelettes discrète Néanmoins, ces dispositifs étant coûteux, pas de compatible avec le matériel et avec la possibilité d'utiliser un seul programme de traitement à la fois, les tâches ainsi doivent être programmé de manière séquentielle non parallèle (pas de notion de parallélisme).

L'un des premiers auteurs à signaler l'utilisation de ces dispositifs étaient Bahoura et al (1997) [4]. Ils développent un algorithme basé sur la transformée en ondelettes adaptée pour l’implémentation en temps réel. Cet algorithme a été implémenté sur un DSP (SPORC-1400) avec une fréquence d'horloge de 50 MHz dans le but de détecter les caractéristiques du signal ECG. Une application nécessite la DWT et la norme JPEG2000 pour la compression d'image a été décrite par Gnavi et al[5]. Les auteurs ont développé un algorithme d'ondelettes sur une plate-forme DSP. Après ce travail, Jichang et al [6], décrit un algorithme pour qu’il doit être implémenté dans le DSP TMS3320C3X à l’aide des instructions de multiplication à Accumulateur parallèle en langage assembler. Tous les travaux mentionnés avants ont été une implémentation de la transformée en ondelette discrète où les donnée été des entiers. Quand la représentation en virgule flottante des coefficients de la DWT est nécessaire. En ce sens, bien qu'il soit possible d’implémenter des opérations en virgule flottante sur les architectures DSP à virgule fixe, le principal inconvénient de cette approche est lié aux temps d’exécution et de taille de la mémoire nécessaire [7].

Afin d'offrir une flexibilité pour l’implémentation de la DWT plusieurs implémentations sur la base des FPGAs ont été proposées. Le premier travail enregistré, décrivant une application en temps réel de la transformée en ondelettes discrètes pour la compression audio et vidéo est rapportée par Motra et al. [8]. Dans ce travail, l'architecture indiqué a été programmé en Verilog-HDI dans un FPGA. La vitesse et la surface occupée sont

prises en compte. Un autre travail similaire est présenté par Zhang et Hu. Ici, les auteurs ont proposé un algorithme de DWT qui été programmé en langage VHDL [9].

Dans le même contexte, des algorithmes de débruitage du signal utilisant la DWT ont été implémentés sur des plates-formes de type FPGAs. Nous pouvons Citer le travail de Marchi et al [10], qui consiste à la comparaison de l'implémentation d'un algorithme de débruitage par ondelettes sur un DSP et un FPGA. Les performances ont été justifiées par la comparaison en termes de fréquences d'horloge, la consommation énergétique et le rapport signal-sur-bruit (SNR) causé par l'effet de l'utilisation de l'arithmétique en virgule fixe sur le FPGA et en virgule flottante sur le DSP.

Un autre travail par M Bahoura basé sur la représentation de nouvelles architectures en temps réel de l'implémentation de la transformée directe/inverse d'ondelette discrète et son application dans le débruitage du signale. Ces propositions ont l'avantage d'assurer une reconstruction parfaite en égalisant les retards de propagation des filtres. Différentes architectures de ces algorithmes sont mis en œuvre sur une carte XUP Virtex-II Pro de Xilinx utilisant des DSP. Ces architectures sont évaluées et comparées en termes d'erreur de reconstruction, les performances de débruitage et de l'utilisation des ressources matérielles [11].

Ce bref aperçu sur l'état de l'art et les tendances récentes de l'implémentation de la DWT appliquée pour le débruitage du signal, menées par différents groupes de recherche. Les implémentations sur FPGAs présentent des avantages qui permettre la préservation de l'architecture parallèle en une seule puce, ce qui augmente les performances de la vitesse d'exécution ainsi que l'optimisation de l'architecture, et le code source peut être modifiée par l'utilisateur avec une relative simplicité. Un inconvénient bien connu de ces systèmes, est l'utilisation des opérateurs arithmétiques de multiplication classique

3 Plan de la thèse :

Le deuxième chapitre est consacré à la présentation des différents circuits logiques programmables disponibles sur le marché. Ensuite, nous abordons aussi la programmation de ces circuits logique programmable.

Le troisième chapitre détaillera l'arithmétique utilisée par les FPGAs, nous proposons dans une première partie la représentation des nombres. Dans une deuxième partie, nous

présentons la réalisation des opérations algébriques et arithmétique implémenté dans des plateformes de type FPGAs.

Le chapitre IV rappelle les fondements de l'analyse multi résolution par ondelettes et dresse un panorama de nouvelles représentations multi résolution. Nous allons donner un aperçu sur le débruitage des signaux dans lequel on expliquera les principales techniques de débruitage par ondelette. On limitera notre description à la partie du débruitage par seuillage.

Le cinquième chapitre présente la vision système de l'implémentation du débruiteur de signal.

Le chapitre VI est le dernier chapitre, c'est en même temps la conclusion de ce travail, et l'ensemble des perspectives qui peuvent être les sujets d'autres études.

Chapitre II

Circuits Logiques Programmables

2. Circuits logiques programmables

3.1 Introduction :

Actuellement les Objets Techniques (O.T.) utilisent de plus en plus la logique programmée (µP, Mémoires, µContrôleur, ...). Ces systèmes électroniques peuvent comprendre au sein d'une même puce aussi bien des technologies numériques et analogiques exigeant des arbitrages matériel–logiciel. Elles utilisent généralement pour réaliser ces fonctions des fonctions logiques de base élémentaires, compteurs, registres , … Le nombre de circuits nécessaires pour remplir ces fonctions peut devenir très vite important. Pour diminuer les coûts de fabrication, de développement et de maintenance, les fabricants de circuits intégrés ont donné naissance aux Circuits Logique Programmable ou encore P.L.D. (Programmable Logic Device).

Ces circuits sont capables pour un O.T. de réaliser plusieurs fonctions logiques dans un seul circuit. Si ces fonctions étaient réalisées à base des circuits de logique classique, il en faudrait plusieurs circuits. Un autre avantage, l'évolution des fonctions d'un l'O.T. s'effectue par programmation comparée à une solution classique où il faut refaire un circuit imprimé si on veut modifier le fonctionnement. Les déférentes catégories des circuits numériques sont présentées dans la figure 2.1

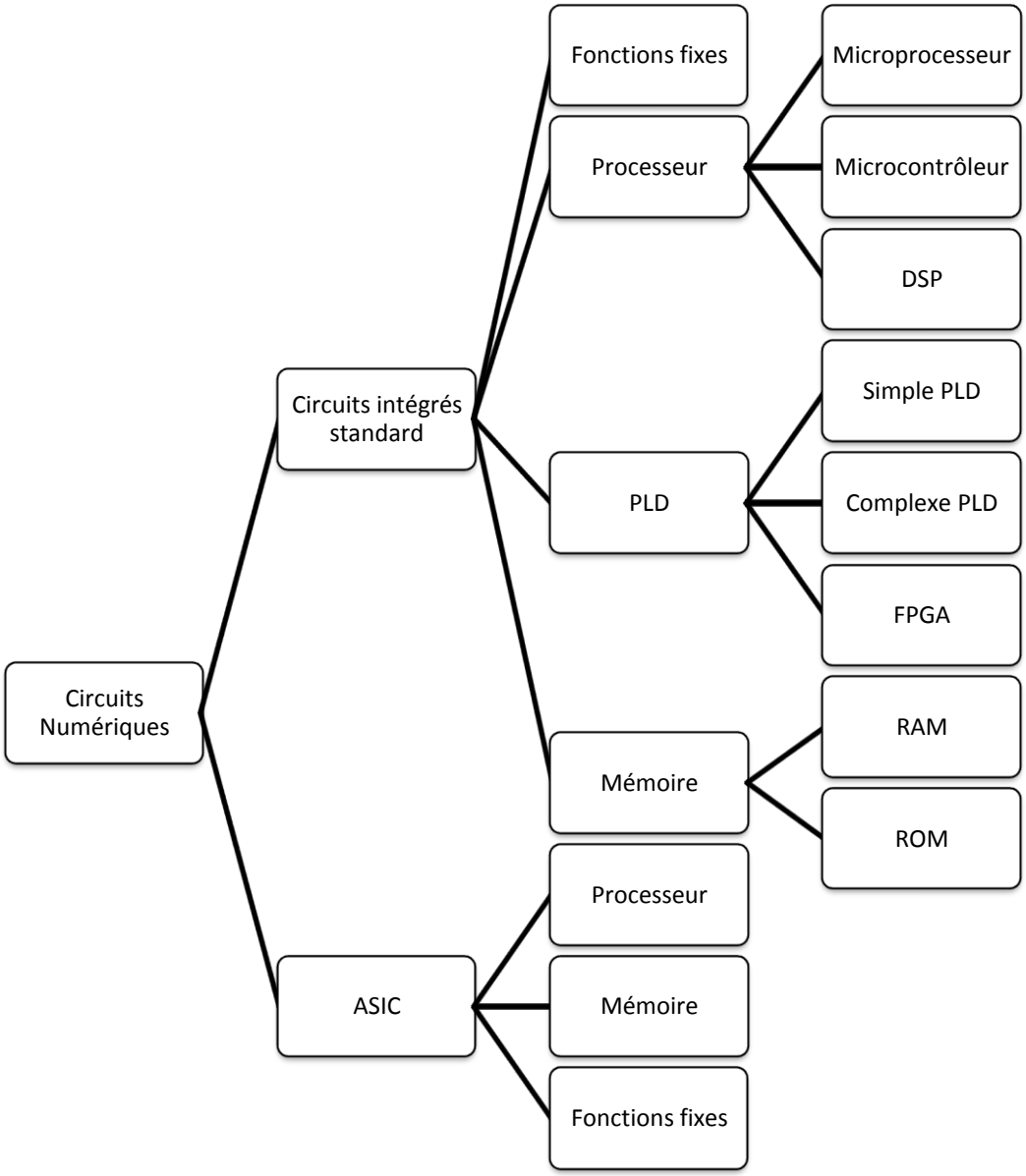


Figure 2.1. Les déférentes catégories des circuits intégrées numériques

Ce chapitre détaille l'histoire des circuits logiques programmables, depuis leurs inventions jusqu'aux leurs complexes architectures modernes. Des tendances courantes telles que les blocs inclus de DSP sont discutées, aussi bien que les langages de description et les outils matériels qui sont employés pour les programmer.

3.2 Histoire de la logique programmable :

Un circuit logique programmable se définit comme un composant discret contenant des modules de logique combinatoires et séquentiels dont les interconnexions sont désignées par programmation. Il peut être configuré et reconfiguré par l'utilisateur pour la réalisation de fonctions logiques [2.1]. Le premier circuit programmable été les réseaux logique

programmable (PLA, pour programmable logic array) inventé par Monolithic Memories Inc. (MMI) l'année 1975 [12] Utilise le fait que toute fonction logique combinatoire peut se mettre sous forme d'une somme (OU logique) de produits (ET logique), c'est ce que l'on appelle classiquement la première forme normale, ou forme disjonctive. ou sous forme d'un produit des sommes en utilisant la loi de De Morgan, la structure du PAL est plutôt intuitive. Elle se compose généralement des entrées avec des inverseurs menant dans une série de portes ET leurs sorties mènent vers une série des portes OU. Ceci fait les produits de n'importe quelle combinaison des entrées et de leurs compléments disponibles sur les portes OU pour les sommer.

Le schéma de la figure 2.2 représente une structure de PLA simple vierge avant programmation, avec tous les connections possibles qui sont réaliser typiquement par des fusibles. La programmation du circuit Pour mettre en application une conception demandé, consiste à supprimer certaines des connexions marquées d'une croix, un programme est employé pour bruler les fusibles avec courant fort afin de supprimer les raccordements non désirés. Si une connexion est supprimée, une valeur constante '1' est appliquée à l'entrée correspondante de la porte ET, c'est ce que symbolise le réseau de résistances relié à cette valeur constante.

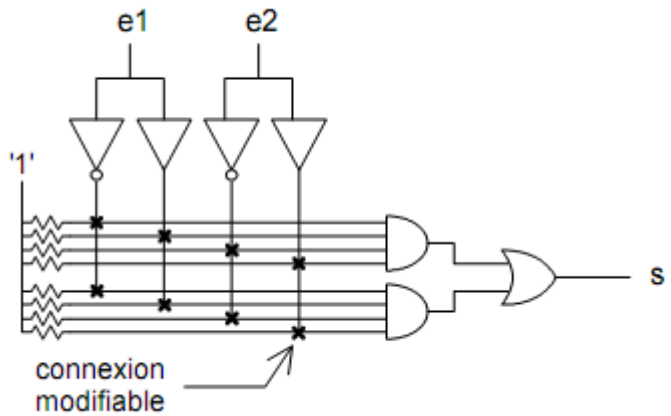


Figure 2.2. La structure d'un PLA avant sa programmation

Un opérateur ou exclusif, par exemple, obéit à l'équation :

$$e1 \oplus e2 = \overline{e1} \times e2 + e1 \times \overline{e2}$$

D'où la programmation du PLA de la figure 2.3

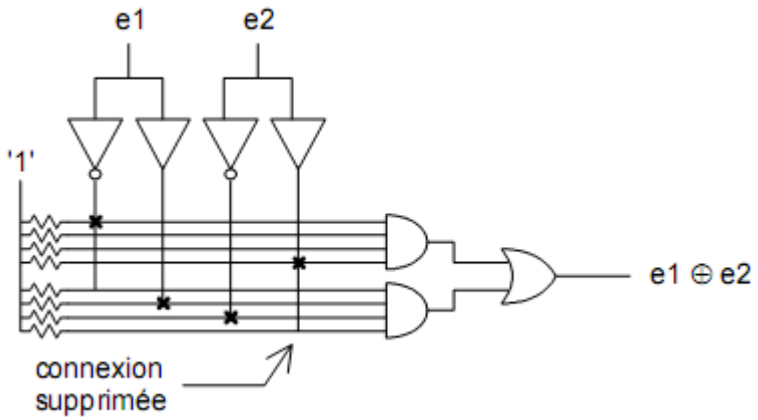


Figure 2.3. Un PLA réalisant un OU exclusif

La plupart des PLDs suivent la structure suivante [13]:

- Un bloc d'entrée qui permet de fournir au bloc combinatoire l'état de chaque entrée et de son complément.
- Un ensemble d'opérateurs « ou » sur les quels les sortie des opérateurs « ET » sont connectées.
- Un bloc de sortie.
- Un bloc d'entrée-sortie, qui comporte une porte à 3 états et une broche d'entrée-sortie.

Le deuxième et le troisième ensemble forment chacun ce qu'on l'appelle une matrice. Les interconnections de ces matrices doivent êtres programmables, et ceci est réalisé par des fusibles qui sont grillés lors de la programmation. Lorsque un PLD est vierge toutes les connections sont assurées.

Le bloc de sortie est souvent appelé macro-cellule que l'on nomme OLMSC (output logic Macro Cell) signifiant macro-cellule logique de sortie comporte : [14]

- Une porte OU exclusif, une bascule D.
- Des multiplexeurs qui permettent de définir différentes configuration et un dispositif de rebouclage sur la matrice ET.
- Des fusibles de configuration (dans les FPGAs on utilise plutôt des cellules de commande des points de connexion.

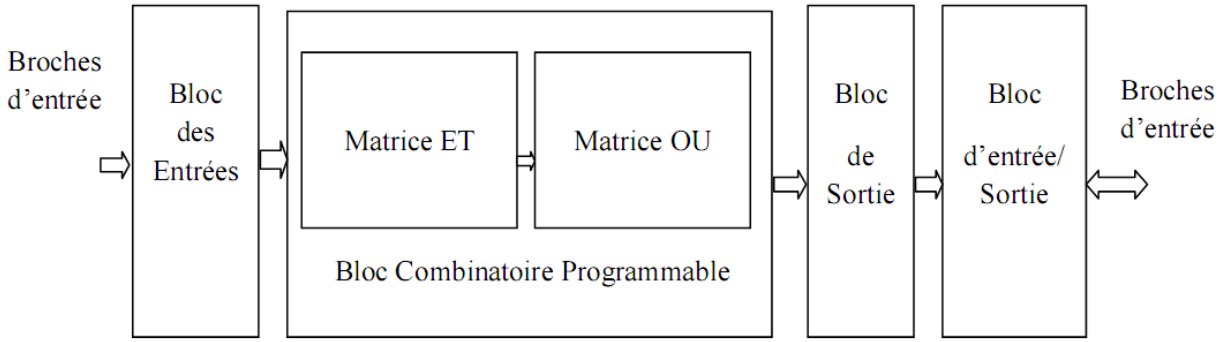


Figure 2.4. Structure de base d'un PLD [12]

Une amélioration du PAL qui sont étudiés jusqu'à maintenant sont connus comme SPLD (Simple Programmable Logic Devices), est venue avec l'introduction du circuit logique programmable complexe (complex programmable logic device CPLD), qui tient compte des circuits logiques plus complexes. Un CPLD figure 2.5 se compose d'un certain nombre de SPLD partagent une matrice d'interconnexion programmable commune. Tandis que les SPLD sont programmés avec un programmeur, un CPLD est programmé par le fabricant ou par un programmeur à travers un câble JTAG relié à un ordinateur.

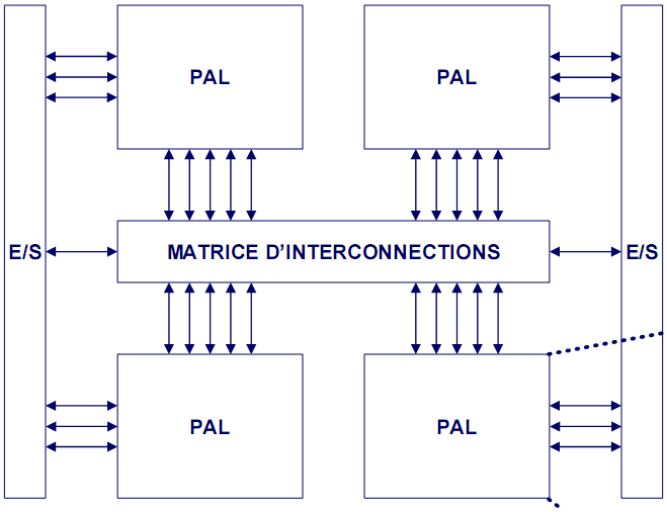


Figure 2.5 : Schéma d'un CPLD

2.3 Les FPGAs :

En 1984 Xilinx a présenté le FPGA pour (field programmable gate array FPGA). C'est un type alternatif du PLD s'est développé plus récemment. Ces circuits en silicium programmables sont devenus au milieu des années 90 des circuits plus complexes grâce à l'intégration de ressources spécifiques dédiées, associées à une mémoire interne et à des ports d'entrées/sorties flexibles [15].

Les FPGAs se composent d'une matrice de blocs logiques élémentaires (CLB) permettant de réaliser des fonctions combinatoires et des fonctions séquentielles, Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées/sorties IOB dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs. Et de ressources d'interconnexion (programmable interconnect) totalement flexibles contrairement au CBLDs [14] la figure 2.6 montre cette structure des FPGAs [16]. Dont le programme, appelé « bitstream » n'a pas vocation à être exécuté par un microprocesseur, mais à configurer des portes logiques et une logique d'interconnexions permettent de relier ces portes logiques entre elles.

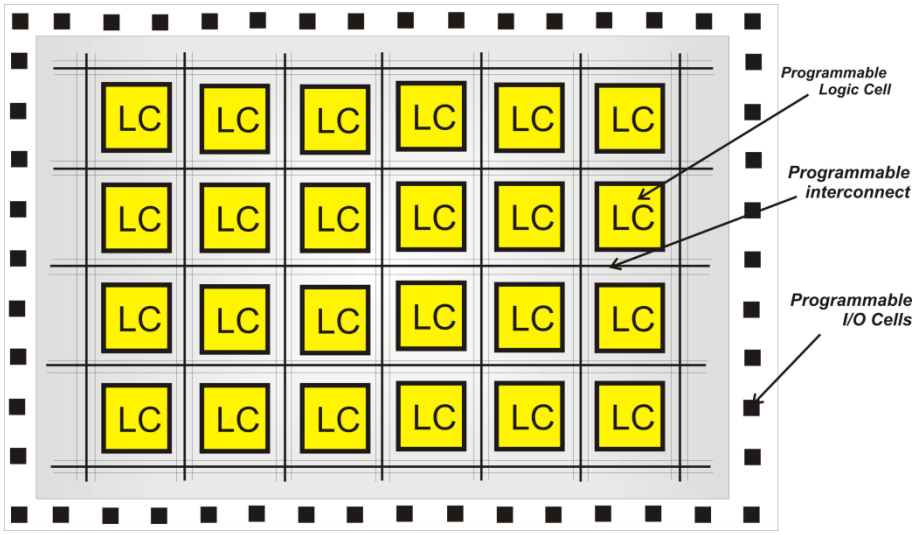


Figure 2.6 : physionomie d'un FPGA [16]

La partie fondamentale du FPGA est appelée « look up table », ou LUT qui agit en tant que générateur de fonction. Historiquement chez la plupart des fabricants de FPGA une LUT contient 16 bits, et permettait donc de réaliser n'importe quelle fonction combinatoire 4 vers 1. A chaque LUT peur être adjoint un registre piloté par une horloge et un multiplexeur permettant d'utiliser le registre ou non. Chaque fabriquant de FPGA a par la suite réalisé des regroupements de LUTs. Et d'ajout de capacité supplémentaire. Aujourd'hui, la structure la

plus utilisée est basée sur une look-up-Table (RAM) pour implémenter une fonction combinatoire plus une bascule D

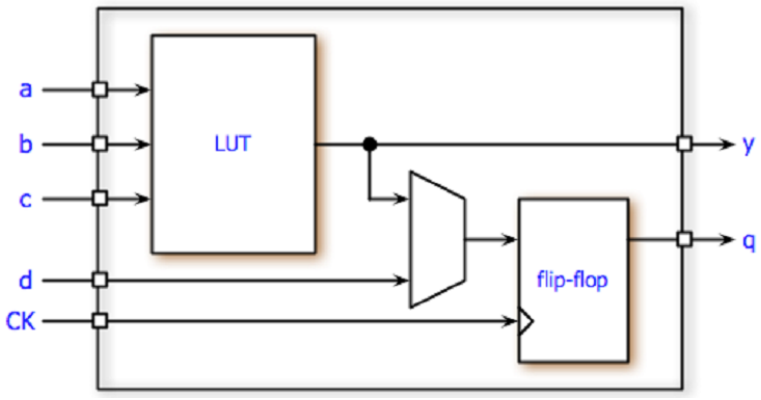


Figure 2.7. Un élément logique de l'FPGA

La fonction de la LUT est de stocker la table de vérité de la fonction combinatoire à implémenter dans la cellule comme il est montré sur la figure suivante :

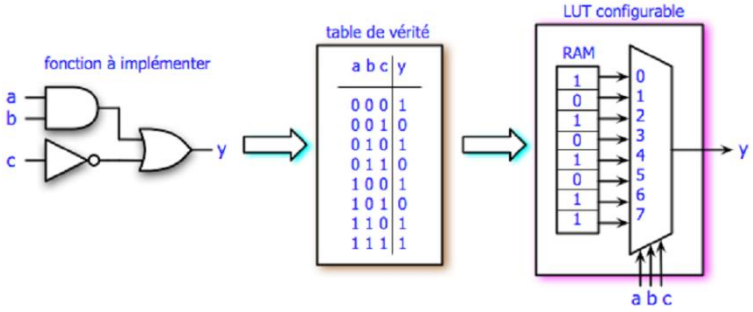


Figure 2.8. Exemple de LUT

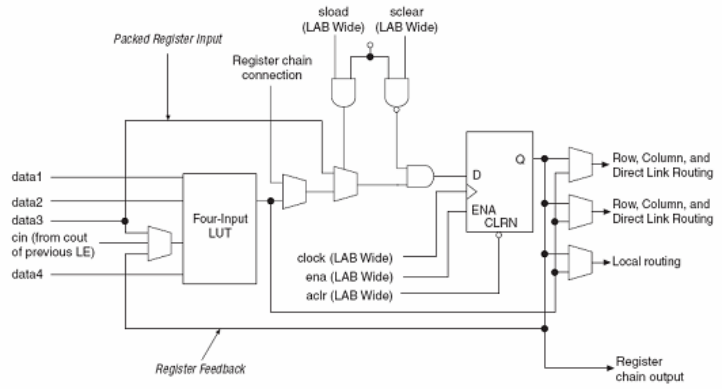


Figure 2.9. Exemple de LUT ALTERA Cyclone II

Chaque fabricant de FPGA a par la suite réalisé des choix de regroupement de LUTs, et d'ajout de capacités supplémentaires.

A titre d'exemple, et parce que ce type de FPGA sera utilisé durant cette thèse, la figure représente un « Altera Logic Module » ou ALM constituant la logique élémentaire du Stratix III. Ainsi , l'LAM est constitue d'une série de LUT, Altera les nomes élément logique (LE) tandis que les FPGAs de Xilinx ont des blocs logique configurables (CLBs), ils pouvant être configuré de nombreuses façons. Le circuit marron (*Cout*) permet une propagation de retenu rapide pour permettre d'accélérer l'arithmétique des nombres, tandis que le circuit bleu (LUT) permet de générer aisément des registres à décalage. Pour plus de détails le lecteur peut se reporter au mauel [17] si l'on prend l'exemple du stratix III, 8 ALMs sont regroupés dans un logic array block (LAB) via une interconnexion locale haute performance [14]. Ces LABs sont regroupés en une matrice d'interconnexion rapide 4X4 (incluent 128 ALM). Chacun de ces groupes est maillé dans une matrice d'interconnexion plus lente 5X3 (soit 1920 ALM). Ces blocs sont ensuite maillés dans une interconnexion globale lente et de taille variable. Cette taille dépend du modèle dans chaque technologie, et se trouve dans nomenclature du composant.

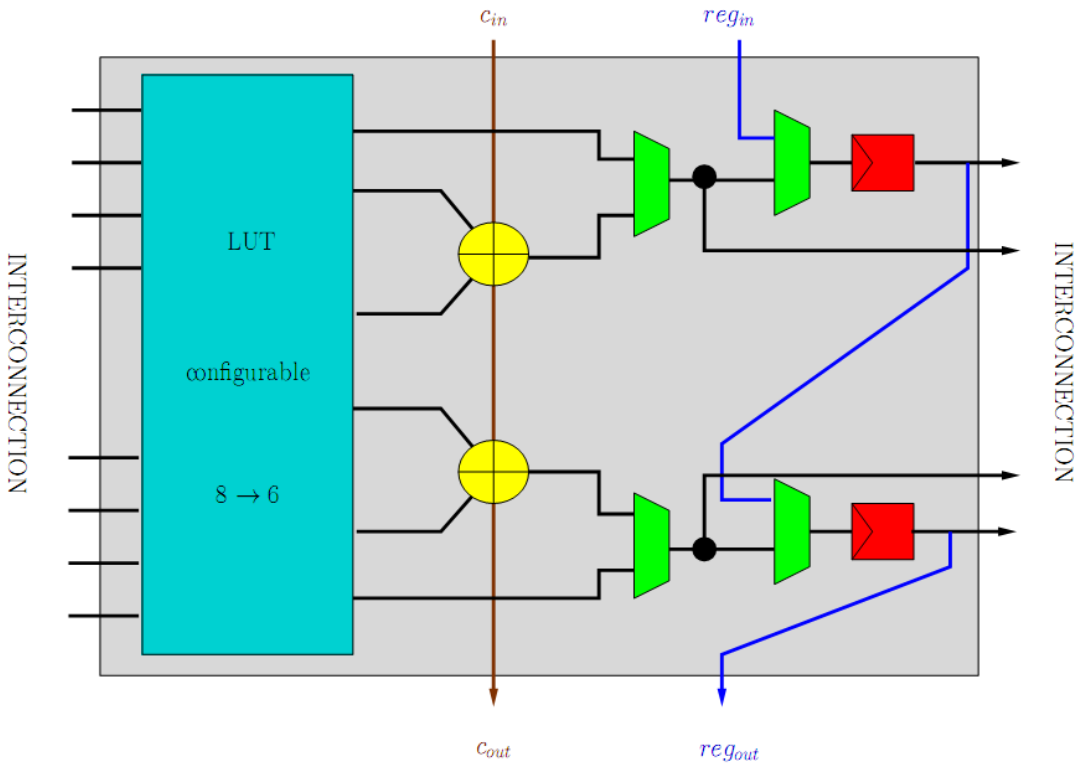


Figure 2.10. Une ALM de stratix III [18]

Chaque CLB dans un FPGA de Xilinx à quatre cellules logique, qui contiennent à leur tour deux 4-entrée générateurs de fonction, carry logique, des portes arithmétiques logique, des multiplexeurs et deux éléments mémoire [19] (figure 2.11).

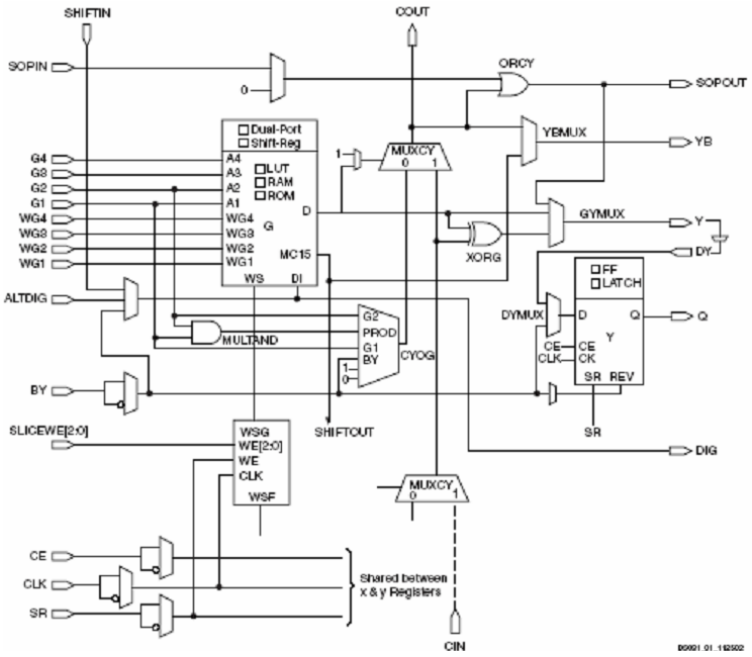


Figure 2.11. Virtex-II Pro Slice (Top Half) [20]

2.4 Architecture des FPGAs :

Les PLD sont configurés par le téléchargement d'une séquence de valeurs numériques (0 ou 1) contenu dans un fichier appelé bitstream. Ces valeurs sont stockées dans la mémoire du circuit, où la mémoire peut être volatile ou non-volatile.

Mémoire volatile : Quand les données sont stockées dans la mémoire, ils seront maintenus dans la mémoire aussi longtemps que la mémoire est reliée à l'alimentation d'énergie. Une fois que l'alimentation est enlevée, alors le contenu de la mémoire (les données) est perdu.

Mémoire non-volatile : Quand des données sont stockées dans la mémoire, ils seront maintenus dans la mémoire même lorsque l'alimentation est enlevée. Un certain FPGAs utilise la technologie d'anti-fuse pour stocker la configuration du FPGA; les nouvelles générations des FPGAs utilisera également les mémoires flash. Les CPLDs utilisent les mémoires non volatiles tels que les EPROM, EEPROM, et les mémoires flash.

La majorité des FPGAs sont à base des mémoires SRAM et donc elles sont volatiles et ils peuvent être programmés aussi facilement que les SRAM standard, mais, elles doivent être programmées chaque fois qu'on les mit sous tension**.** C'est pour ça on les couples avec un autre circuit de mémorisation du programme de configuration, tel qu'une mémoire PROM.

La cellule SRAM (Figure 2.12) à un mode d'écriture et un autre de lecture. Elle est couplé aux points de configuration dans le FPGA.

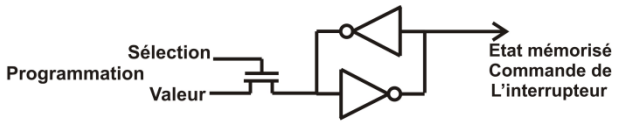


Figure 2.12 : Cellule SRAM

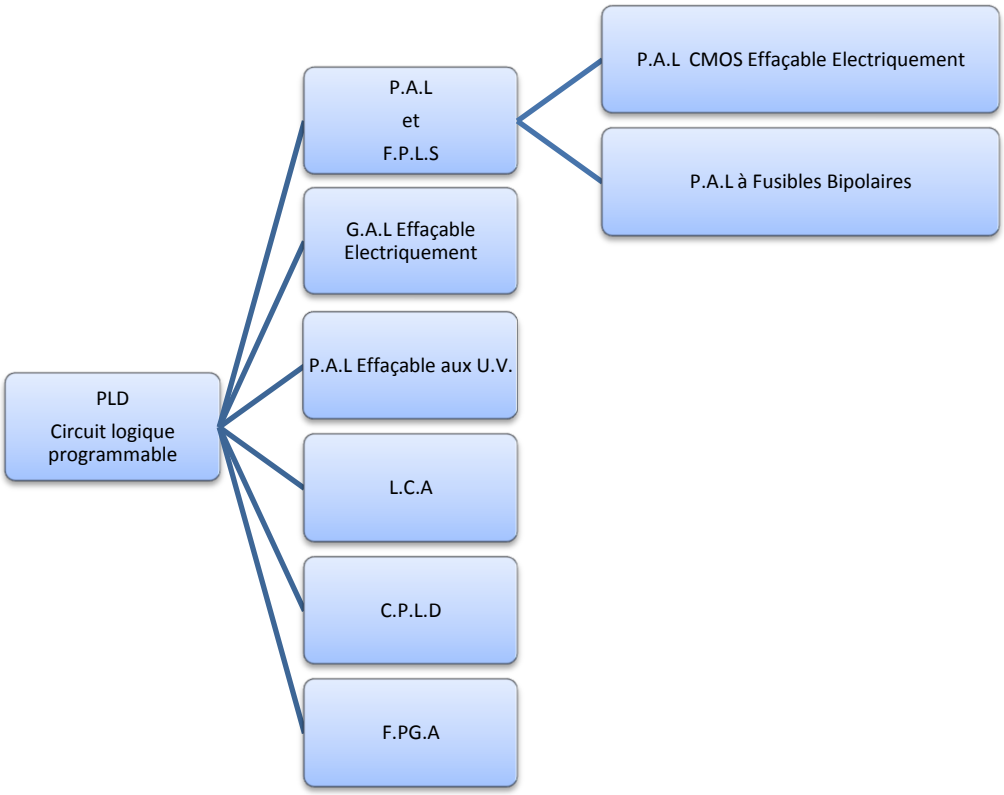


Figure 2.13 : Résumé graphique des familles de P.L.D

2.5 Fournisseurs des circuits logiques programmables :

Les PLDs sont offerts par un ensemble de fournisseurs, dont chacun fournit une famille de PLD Basé sur les SPLD, CPLD ou FPGA. Ils fournissent également un ensemble d'outils EDA (Electronic Design Automation) d'aide pour le processus de conception de puis la saisie, la simulation et vérification jusqu'a la synthèse. Les principaux fabricants des

FPGAs dans le monde on peut citer : Xilinx, Altera, Actel, Atmel, QuickLogic, Lattice et d'autres.

D'après Ed Lepkowski (L-Mar Associates) cité par Electronics Weekly, Altera aurait vu sa part du marché de la logique programmable passer de 35,3 % à 40,7 % entre 2009 et 2010. L'américain aurait ainsi grignoté son compatriote Xilinx qui, lui, passerait de 52,8 % à 48,3 % de part de marché et reste le leader, ainsi qu'Actel (N°1 du marché des FPGAs Antifusibles et FLASH [21]) passera de 6 % à 5 % désormais dans le giron de Microsemi. Seul Lattice aurait également grappillé quelques parts de marché, passant de 5,7 % à 6,1 %.

Dans leur globalité, les ventes de FPGA et CPLD ont explosé en 2010, passant de 3,38 à 4,78 milliards de dollars, permettant à tous les acteurs de voir leurs revenus croître fortement. [22]

2.6 Outils de Développement :

L'objectif du processus de développement d'une application utilisant les PLD passe par un certain nombre d'étapes allant de la description de l'application à la programmation du composant. Le développement sur les différents circuits logiques peut être désigné par la Figure 2.14.

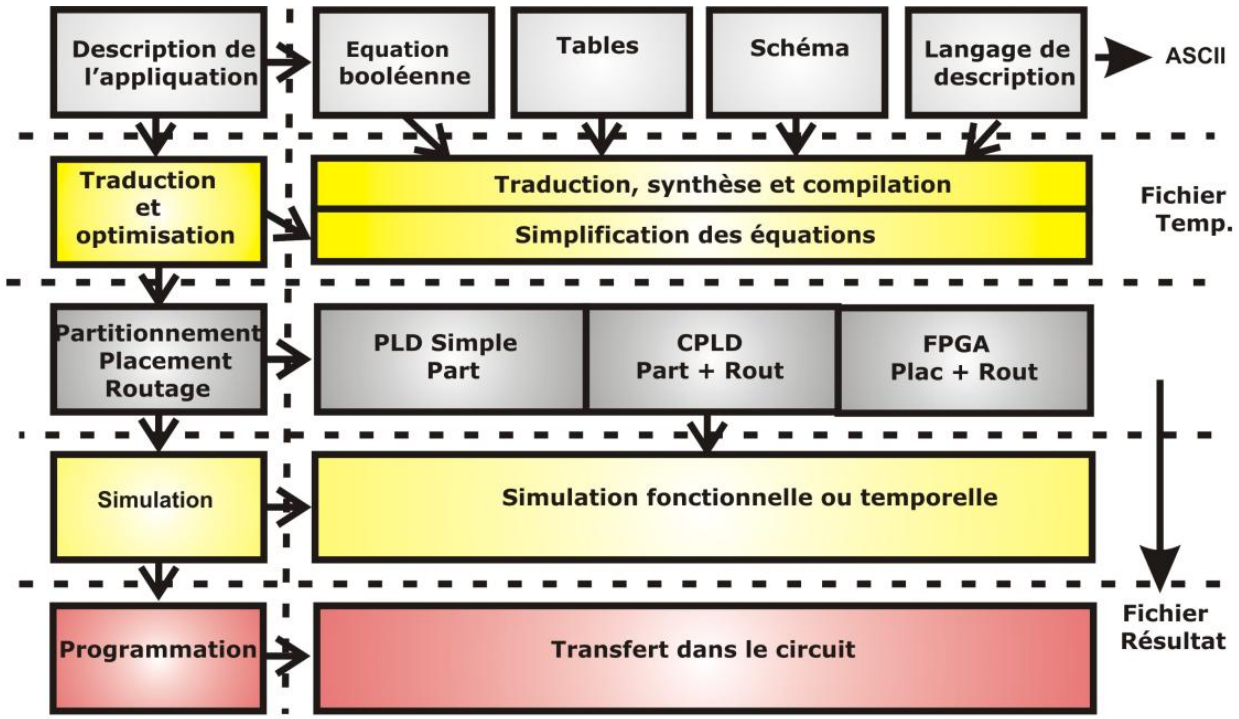


Figure 2.14 : Les étapes du développement pour les PLD

2.6.1 Description de l'application :

Définir manuellement les connections de routage dans un circuit programmable peut être faisable avec les premiers PALs mais il est presque impossible sur les FPGAs moderne considérant la densité d'intégration. La programmation de ces circuits programmables peut être réalisée soit par outil de saisie de schémas, ou par l'utilisation des langages de description matérielle (HDLs), et l'utilisation des compilateurs de langue de haut niveau, sur une platforme de programmation. Ces méthodes sont énumérées dans les niveaux croissants de l'abstraction, pour l'outil de saisie d'un schéma la conception étant le niveau le plus bas. La tendance actuelle pour ces circuits est la possibilité de réaliser des systèmes sur puce (ou SoC) en utilisant des « composants virtuels » et de concevoir ainsi des blocs de propriété intellectuelle (Intellectual Property ou IP) qui sont par exemple des fonctions VHDL/Verilog génériques réutilisables.

2.6.1.1 Outil de saisie d'un schéma :

La conception schématique pratique des bases choisissant des portes logiques standard à partir d'une bibliothèque pour créer une description graphique du circuit à réaliser, et les câblant manuellement. La bibliothèque schématique de conception inclut typiquement des portes logiques booléennes, des multiplexeurs, des buffers d'I/O, et des macros spécifiques

pour des fonctions de circuit, telles que des diviseurs d'horloge. Des composants faits sur commande peuvent être construits par des blocs plus petits pour créer des macros d'utilisateur pour l'usage dans des grandes conceptions.

Prenant comme exemple, la création d'un additionneur incomplet, son fonction est d'additionner deux bits, on construit sa table vérité, comme montré dans Table2.1.

Table 2.1. Table de vérité d'un additionneur incomplet

A	*B*	*S*	***Cout***
0	0	0	**0**
0	1	1	**0**
1	0	1	**0**
1	1	0	**1**

Un additionneur additionne deux entrées binaires A et B, pour produire a la sortie deux bits la somme S et retenue (Carry) Cout. Les équations logiques de cette structure à implémenté peuvent être soustraite a partir de la Table de vérité :

$$S = A\overline{B} + \overline{A}B = A \oplus B$$
$$Cout = AB$$

Une fois que les équations logiques sont déterminées le circuit peut être facilement saisi comme il est représenté sur le schéma 2.15. Un inconvénient, cependant, il est difficile d'extraire une fonction de logique à partir du schéma. En outre, les changements du circuit à travers la modification schématique est lourde.

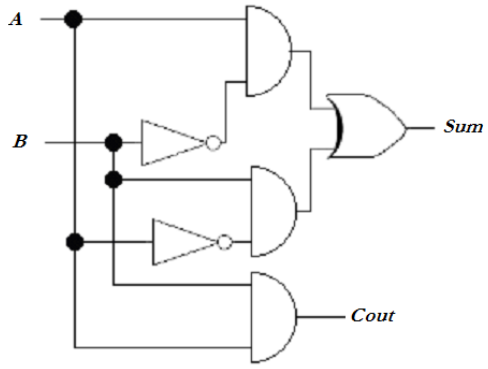


Figure 2.15. Schéma de l'additionneur incomplet

Il convient à noter que quel que soit le niveau d'abstraction, l'outil de synthèse optimisera la conception pour la structure spécifique du circuit et le résultat final peut différer de manière significative de la conception originale.

2.6.1.2 Les langages de description de matériel :

Actuellement la densité de fonctions logiques (portes et bascules) intégrée dans les PLDs est telle (plusieurs milliers de portes voire millions de portes) qu'il n'est plus possible d'utiliser les outils de saisie d'un schéma pour développer les circuits. Les sociétés de développement et les ingénieurs ont voulu s'affranchir des contraintes technologiques des circuits. Ils ont donc créé des langages dits de haut niveau à sa voir aux langages de description matérielle de haut niveau (les HDL, Hardware Description Language). Deux d'entre eux ont émergé et sont couramment utilisés : VHDL (VHSIC Hardware Description Language) et Verilog. Ces deux langages bénéficient du support de la quasi-totalité des logiciels [23].

Toutes les deux sont basés sur des descriptions écrite (texte) de la structure électroniques d'un circuit numérique, Ils permettent au code écrit d'être portable, c'est à dire qu'une description écrite pour un circuit peut être facilement utilisée pour un autre circuit. Il faut avoir à l'esprit que ces langages dits de haut niveau permettent de matérialiser les structures électroniques d'un circuit.

a. Le VERILOG : Historique et description

La société Gateway Design Automation Inc. En 1984 lance le langage Verilog comme langage matériel, Le langage original appelé HiLo, s'apparente beaucoup au langage C [24]. Ce qui suscite un intérêt immédiat notamment de la part de la communauté des informaticiens. Au départ, il était un langage propriétaire et n'était pas standardisé. En 1989, Cadence devient le propriétaire de la société Gateway et donc du langage Verilog et de son mécanisme « d'interface de langage de programmation » (Progamming Language Interface : PLI). En 1990, Verilog est ouvert au domaine public ce qui permet à chacun de développer son simulateur en Verilog et devenir un compétiteur potentiel de Cadence. Ainsi, le Verilog pouvait devenir le langage de description matérielle le plus répandu et le plus utilisé. C'est pourquoi le groupe de travail OVI (Open Verilog International) a été formé sur le langage VERILOG et du mécanisme PLI. La popularité du Verilog augmentant et l'arrivée du langage VHDL, il devenait important pour les personnes de l'OVI de fonder un comité de travail pour sa standardisation IEEE. Ce comité a été formé en 1993 et a donné lieu au standard 1364-1995 qui combine la syntaxe du langage Verilog et son PLI.

Pour ces équipes de conception, les points positifs de ce langage sont : une facilité de prise en main pour les habitués du langage C ainsi qu'une assignation des signaux moins contraignante que le VHDL. Par contre, cette absence peut être piégeuse dans la mesure où le type de signaux (bus, signé …) n'est pas pris en compte. De plus, Verilog ne comporte pas de concept de package (ni de librairies). Ainsi chaque fonction et chaque procédure d'un module doivent être définies dans ce module.

La description d'un système mixte est aussi possible en Verilog et porte le nom de Verilog-AMS HDL. Ce langage est défini sur la base du Verilog 1364-1995 pour la partie numérique, et du Verilog-A pour les systèmes analogiques.

b. VHDL : Historique et description

Le VHDL est un langage de description matériel, Dans les années 80, le département de la défense aux Etats-Unis fait un appel d'offre pour avoir un langage de description matérielle numérique unique. Le langage VHDL (VHSIC Hardware Description Language) est inventé pour répondre à ces critères. Ce langage se base sur le VHSIC (Very High Speed Integrated Circuit) qui est un projet de recherche nationallemené par le groupement IBM/Texas Instruments / Intermetrics. Ce langage est ouvert au domaine public en 1985 et deviendra une norme en 1987 sous la dénomination de IEEE 1076-1987. Des changements minimes se feront pour la seconde normalisation en 1993 et il portera le nom VHDL'93. La dernière évolution de la norme est la norme IEEE 1076-2001.

Les avantages qui se dégagent de ce langage sont la réutilisation de fichiers grâce à la notion de paquet et la généricité individuelle des modèles.

Cependant, la norme IEEE 1076-2001 ne permet pas seule la description mixte (numérique et analogique). Le Verilog et le VHDL ont des capacités techniques équivalentes [24]. Le choix du langage est souvent dicté par la « culture » de l'équipe de développement ou de l'équipe de recherche. De plus, ce choix est parfois imposé par les outils disponibles, les logiciels de simulation et de synthèse, dont les langages associés sont fixés par des aspects économiques. Nous pouvons noter toutefois que les outils actuels permettent de mélanger ces deux langages.

VHDL a continué son évolution et la description analogique vient d'y être intégrée sous la référence IEEE 1076.6 (VHDL-AMS). L'extension vise l'analogique.

Pourquoi un langage de description ? L'électronicien a toujours utilisé des outils de description pour représenter des structures logiques ou analogiques. Le schéma structurel que l'on utilise depuis si longtemps et si souvent n'est en fait qu'un outil de description graphique. Aujourd'hui, l'électronique numérique est de plus en plus présente et tend bien souvent à remplacer les structures analogiques utilisées jusqu'à présent. Ainsi, l'ampleur des fonctions numériques à réaliser nous impose l'utilisation d'un autre outil de description. Il est en effet plus aisé de décrire un compteur ou un additionneur 64 bits en utilisant l'outil de description VHDL plutôt qu'un schéma.

Le deuxième point fort du VHDL est d'être "un langage de description de haut niveau". D'autres types de langage de description, comme l'ABEL par exemple, ne possèdent pas cette appellation. En fait, un langage est dit de haut niveau lorsqu'il fait le plus possible abstraction de l'objet auquel ou pour lequel il est écrit. Dans le cas du langage VHDL, il n'est jamais fait référence au composant ou à la structure pour lesquels on l'utilise. Ainsi, il apparaît deux notions très importantes : portabilité des descriptions VHDL, c'est-à-dire, possibilité de cibler une description VHDL dans le composant ou la structure que l'on souhaite en utilisant l'outil que l'on veut.

c. Structure d'une description VHDL :

En VHDL, une structure logique est décrite sous la forme d'une paire de fonctions, d'une part une entité (ENTITY) et d'autre part, l'architecture (ARCHITECTURE) de la façon suivante :

```
LIBRARY LIBNAME;
USE LIBENAME.PACKAGENAME.ALL;
ENTITY ENTITY_NAME IS
PORT (
     SIGNAL_NAME :  MODE SIGNAL_TYPE;
     .
     .
     .
     SIGNAL_NAME :  MODE SIGNAL_TYPE);
END ENTITY_NAME;

ARCHITECTURE ARCHITECTURE_NAME OF ENTITY_NAME IS
      DECLARATION DE COMPOSANTS
   +  DECLARATION DES SIGANUX INTERNES
   +  AUTRE DECALARATION ...
BEGIN
   INSTRUCTION CONCURENTES;
   PROCESSUS;
   INSTANCES DE COMPOSANT;
END  ARCHITECTURE_NAME;
```

L'entité donne les informations concernant les signaux d'entrées et de sorties de la structure ainsi que leurs noms et leurs types. Par contre l'architecture décrit le comportement d'intérieur de l'entité. Il est possible de créer plusieurs architectures pour une même entité. Chacune de ces architectures peut décrira l'entité de façon différente.

2.6.2 Traduction et optimisation :

La description VHDL définit la fonctionnalité du circuit en termes de blocs définis " haut niveau ". Progressivement, les blocs sont détaillés précisément jusqu'à une description proche des ressources matérielles. En effet, le langage VHDL autorise trois niveaux de description [25].

- le niveau structurel décrit le câblage des composants élémentaires,
- le niveau flot de données décrit les transformations d'un flot de données de l'entrée à la sortie,
- le niveau comportemental décrit le fonctionnement par des blocs programmes appelés Processus qui échangent des données au moyen de signaux comprenant des instructions séquentielles.

Une description VHDL d'un circuit subit toute une suite de transformations avant que ce circuit soit effectivement configuré sur un FPGA.

L'étape de synthèse consiste à lire la description VHDL du circuit pour en extraire toute la structure logique. Lorsque des composants de ce circuit sont décrits de manière comportementale, c'est le rôle du synthétiseur d'inférer la logique nécessaire à la réalisation des comportements spécifiés. Cette tâche est la modélisation au niveau Transfert de registres (Register Transfer Level). Cette modélisation revient à décrire l'implémentation sous forme d'éléments séquentiels (les registres ou bascules) et de combinaisons logiques entre les différentes entrées/sorties des éléments séquentiels et des entrées/sorties primaires du circuit. Cette modélisation est codée à l'aide d'un langage de programmation. Cela se fait soit en VHDL, soit en Verilog. Le propre d'une description RTL est d'être automatiquement synthétisable en portes logiques combinatoires (portes ET, OU, multiplexeur, etc.) et séquentiels (comme les bascules D synchrones) issues d'une bibliothèque de cellules standard (Standard Cell Library). Pour cela, le codage RTL en VHDL ou Verilog doit suivre certaines règles précises afin que le code soit dit « synthétisable ». Il est tout à fait possible de décrire

une fonctionnalité équivalente en VHDL ou Verilog qui ne soit pas synthétisable. On appelle cela une description comportementale de plus haut niveau que le niveau RTL.

Un autre rôle du synthétiseur est d'appliquer diverses optimisations logiques au circuit. Il cherche ainsi à minimiser toutes les expressions logiques utilisées par ce circuit. Enfin, le synthétiseur après compilation produit une netlist c'est-à-dire une description de haut niveau (liste de fils et de porte logiques) à partir de la description RTL du circuit.

2.6.3 Partitionnement placement et routage :

La netlist du circuit est ensuite adaptée aux primitives logiques du FPGA cible. Cette étape de ciblage technologique permet ainsi de passer d'une description structurelle générique à une description spécialisée pour une architecture précise. Les éléments de base ne sont alors plus de simples portes logiques mais des cellules programmables du FPGA.

La phase de mapping ayant identifié les ressources logiques nécessaires à l'implémentation d'un circuit, la tâche de placement/routage consiste alors à disposer et connecter ces ressources sur la surface du FPGA. Il est possible de donner au placeur/routeur des contraintes spatiales ou temporelles, pour fixer par exemple la position des ports d'entrée/sortie du circuit ou bien une période d'horloge maximale.

A l'issue de l'étape de placement/routage, le circuit obtenu est tel qu'il sera programmé sur le FPGA. C'est sur ce modèle que sont calculées les estimations de surface et de latence du circuit.

2.6.4 Simulation :

La simulation logicielle d'un circuit est une étape essentielle de sa conception. Il faut en effet s'assurer que celui-ci fonctionne correctement avant de le configurer sur le FPGA. La simulation permet ainsi de fournir au circuit des vecteurs de test en entrée, puis de vérifier que le circuit se comporte de la manière prévue. Bien entendu, la simulation ne garantit pas la correction du circuit. Mis à part pour de petits opérateurs, il est en effet impossible de réaliser une simulation exhaustive à cause de la lenteur de la simulation logicielle. Il faut donc essayer de choisir des vecteurs de test adaptés à l'opérateur pour couvrir le mieux possible les différents cas de figure.

Le simulateur logiciel utilisé pour nos architectures est l'édition gratuite de ModelSim-Altera version 6.3g, développée par Altera.

Comme mentionné plus haut, la simulation logicielle est bien trop lente pour pouvoir réaliser des tests exhaustifs. Il est toute fois envisageable, pour certains cas, d'effectuer ces tests en utilisant des cartes de développement pour FPGA, munie de ports d'entrée/sortie rapides. Cela a par exemple été le cas de notre application.

2.6.5 Configuration (programmation) :

Enfin, la dernière étape avant la programmation effective du FPGA consiste à générer un bitstream. Ce bitstream est en fait un fichier contenant tous les bits de configuration du FPGA, construit pour correspondre parfaitement au circuit décrit lors du placement/routage.

Une fois le bitstream créé, il ne reste plus qu'à le télécharger sur le FPGA grâce à une interface dédiée. Le FPGA restera ainsi configuré tant qu'il sera sous tension, ou bien lorsqu'un autre circuit y sera programmé suivant le même procédé.

ISE, est l'environnement de développement fourni par Xilinx. Cette suite logicielle inclut notamment le synthétiseur XST, ainsi que tous les outils évoqués précédemment. Par contre Altera fourni l'environnement de développement Quartus.

2.7 Les atouts de la technologies FPGA

2.7.1 Performances :

Comme ils tirent parti du parallélisme matériel, les FPGAs offrent une puissance de calcul supérieure à celle des processeurs de signaux numérique (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opération par cycle d'horloge. Berkeley Design Technology Inc (BDTI), une importante société d'analyse et de « benchmarking », a publié des études montrant que les FPGAs peuvent offrir une puissance de traitement supérieure à celle d'une solution DSP dans certaines applications [26]. Contrôler les entrées sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités, qui répondent mieux aux besoins de l'application.

2.7.2 Temps de mise sur le marché :

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide, Ainsi, vous pouvez tester ou un concept, puis sur des matériels sans savoir à passer le long processus de fabrication d'un ASIC personnalisé [27]. Par la suite, vous pourrez porter les éventuelles modifications nécessaires à votre FPGA, en quelques heurs au lieu de quelques semaines. Le matériel « sur étagère » actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les d'abstraction. Ces outils comprennent souvent des cœurs de Propriété Intellectuelle (fonction précompilées) utiles pour le contrôle et le traitement de signaux.

2.7.3 Coût :

Les couts d'ingénierie non récurrents (NRE) des ASIC personnalisées sont bien supérieures à ceux des solutions matériels basées sur du FPGA. L'important investissement de départ que requièrent les ASICs se justifie pour les OEM. par exemple, qui peuvent livrer des circuits par milliers chaque année. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la re-conception d'un ASIC.

2.7.4 Fiabilité :

Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés des processeurs comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche.

2.7.5 Maintenance :

Comme nous l'avons vu, les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la re-conception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, vous pouvez y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit.

2.8 Les tendances courantes :

Le FPGA offre une souplesse de conception grâce à sa facilité d'utilisation et sa facilité de programmation (et reprogrammation). Contrairement à un circuit ASIC, pour lequel le concepteur maîtrise totalement le placement routage au niveau transistor, le FPGA n'autorise pas cette opération qui s'effectue de manière transparente pour le développeur. Pour une application visant la mise sur le marché d'une forte qualité des produits, le circuit spécifique est la solution faible coût. Par contre, dans le cas d'un développement ponctuel, le FPGA est nettement plus avantageux. C'est pourquoi ce composant est plus adapté pour la mise au point de prototypes et accessible à un plus grand nombre d'utilisateur que l'ASIC. La tendance courante des architectures des FPGAs s'oriente vers des systèmes embarqués complet. Les densités des FPGAs ont augmenté jusqu'au point qu'on peut implémenter un microprocesseur entier de type RISC dans un seule circuit. Identifiant cette tendance, les constructeurs des circuits FPGA intègres également des macros hardwares tels, blocs mémoire (RAM), des multiplicateurs, et des corps de microprocesseur dans plusieurs types FPGAs. Et ils proposent aussi des macros optimiser pour les UAL, microprocesseur RISC, une interface PCI ou d'autre éléments de cette complexités. Ainsi qu'ils s'orientent vers inclure des macros de signal mixed tels que les ADCs et DACs qui augmente l'utilité du FPGA.

Le circuit Excalibur d'Altera contient un corps du processeur ARM922T tandis que le circuit Virtex-II Pro de Xilinx contient jusqu'à quatre microprocesseurs de PC d'IBM. Ceci permet aux développeurs d'optimiser leurs applications pour avoir le maximum de performance.

2.9 Le choix des FPGAs

Pour réussir une application à base d'un FPGA et afin d'obtenir un système plus performant, consommant un minimum de puissance, il est nécessaire de respecter un certain nombre de règles comme. En premier lieu il faut connaitre les caractéristiques qui doivent être dans le FPGAs cible pour assurer son adéquation avec les besoins du projet, puisque les fondeurs des FPGAs définissent ses propres structures des blocs logiques configurables, des cellules logiques ; de la logique fixe, comme des multiplicateurs ; des ressources de mémoire, comme des blocs de RAM intégrés. Et bien que souvent les FPGAs contiennent aussi d'autres composants, comme des micro-processeurs et des circuits spécifiques ainsi que la taille des circuits (nombres de blocs disponibles, nombre de port d'entrée sortie). Ceci est dû au fait que les FPGAs sont utilisée dans des applications variées et il n'y a pas un seul circuit pour toutes les applications. Ceux qui sont utilisés en aérospatial ne sont pas prédestinés à réaliser les mêmes fonctions ni à opérer dans les mêmes conditions que ceux utilisés en automobile ou bien en téléphonie mobile [28].

Le choix du bon FPGA est simplifié par la classification adoptée parles fabricants, Xilinx s'est organiser en familles au moment où Altera s'est organisé en série. Chaque famille ou série est constituée de membres ou sous famille. Les membres d'une même famille se partagent les même caractéristiques de base et chaque membre se distingue des autres par ses propres caractéristiques à savoir le mémoire, les ressources disponibles ou le nombre d'entrée/ sortie

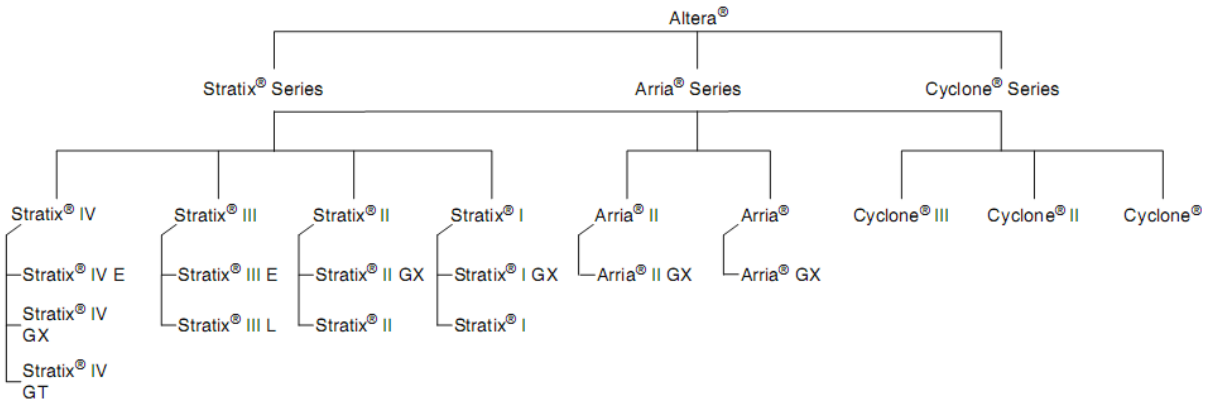


Figure 2.16. L'arbre des séries d'Altera

Maitriser les outils d'implémentation et de choisir des outils de synthèse de qualité. La possession d'un logiciel de programmation et des outils permettant une simulation des programmes sont également souhaitables.

2.10 La carte de développement DE2

2.10.1 Présentation

La carte de développement DE2 a été utilisée comme plate-forme de développement. Elle contient un mono-chip FPGA (Cyclone II EP2C35F672C6). Ce circuit fabriqué par la société Altera englobe 35.000 cellules (Logics Elements) utilisables dans un boîtier de 672 broches (BGA) comme un noyeau. La carte dispose d'un ensemble de mémoire (SRAM (61LV25616) : 256 k*16 (10ns)), SDRAM (IS42S16400-8) : 4 M*16 (100 MHz <2-2-2>), Flash (S29AL032D) : 4 M*8 (10ns) et une Flash de configuration: EPCS16 , cette mémoire permet de stocker de manière non-volatile la configuration du FPGA, elle est dotée d'un convertisseur audio (Codec Audio (WM8731) : 24 bits mono (8 - 96 kHz)), vidéo et TV (Décodeur Vidéo/TV (ADV7181B) : NTSC/PAL 50/60 Hz), ainsi que d'interfaces IrDA, Ethernet et USB ; la carte est munie d'un afficheur LED et écran LCD et d'un grand nombre de Switch et de 4 boutons poussoirs.

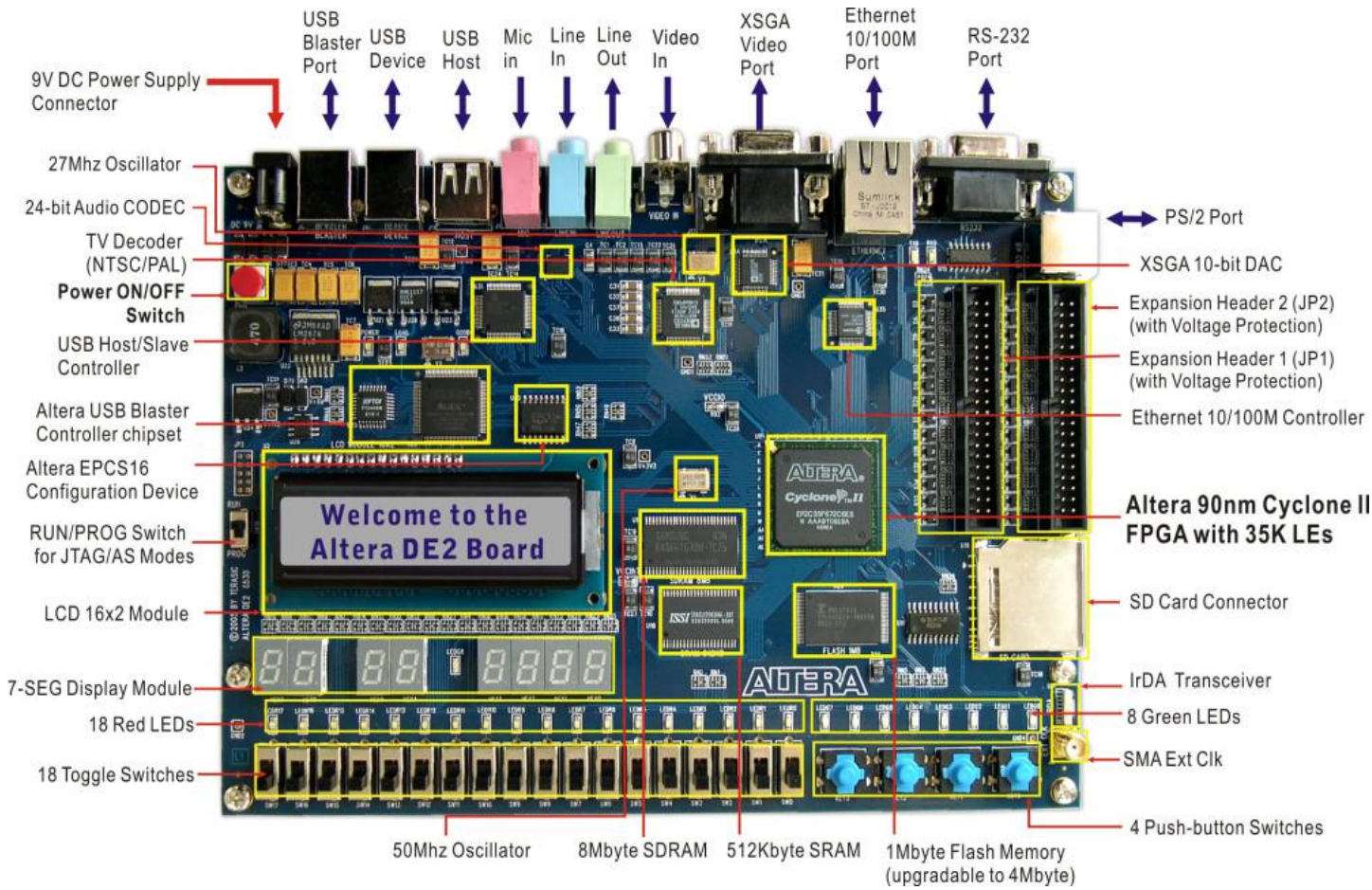
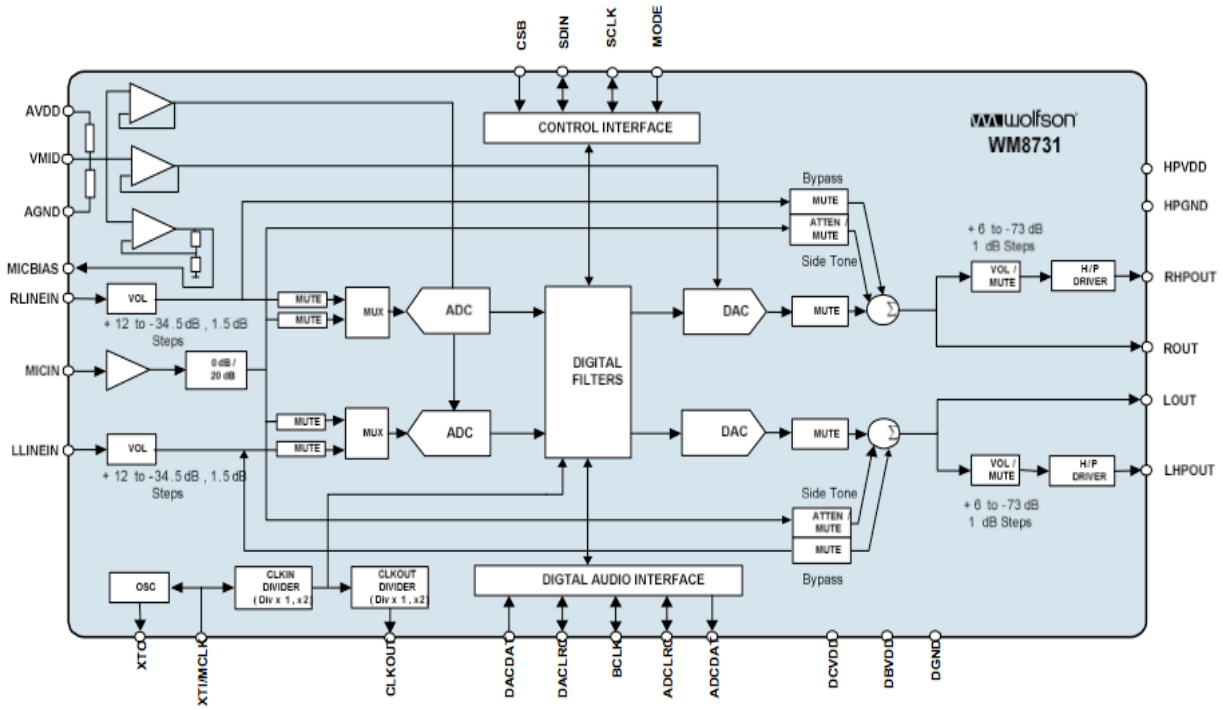


Figure 2.17. La carte DE2

2.10.2 CODEC AUDION

Le WM8731 est un CODEC Audio qui se trouve sur la carte DE2. C'est un composant développé par la société Wolson Microelectronics [29]. Ce composant comporte un ensemble de registres de programmation qui se réalise grâce au bus I²C (Inter Integrated Circuit) qui fait partie des bus série. La propriété intellectuelle (IP) que nous avons développée doit permettre de configurer tout les registres de ce codec audio.

Ce codec audio comporte des drivers haut parleur, un 24-BIT sigma-delta ADCs et un DACs et d'autres fonctions. Ce codec peut être utilisé dans plusieurs applications comme par exemple, les lecteurs MP3, les enregistreurs de voix ou encore les enregistreurs CD et mini-disc. Le WM8731 comporte des entrées audio comme line in et mic-in et des sorties haut



parleur qu'on choisira de les mettre en mode mono ou stéréo. Ci-dessous, une représentation du « Bloc diagram » de ce composant :

Figure 2.18. Le Bloc diagramme du WM8731

Chapitre III

Arithmétique sur FPGA

3 ARITHMETIQUE SUR FPGA

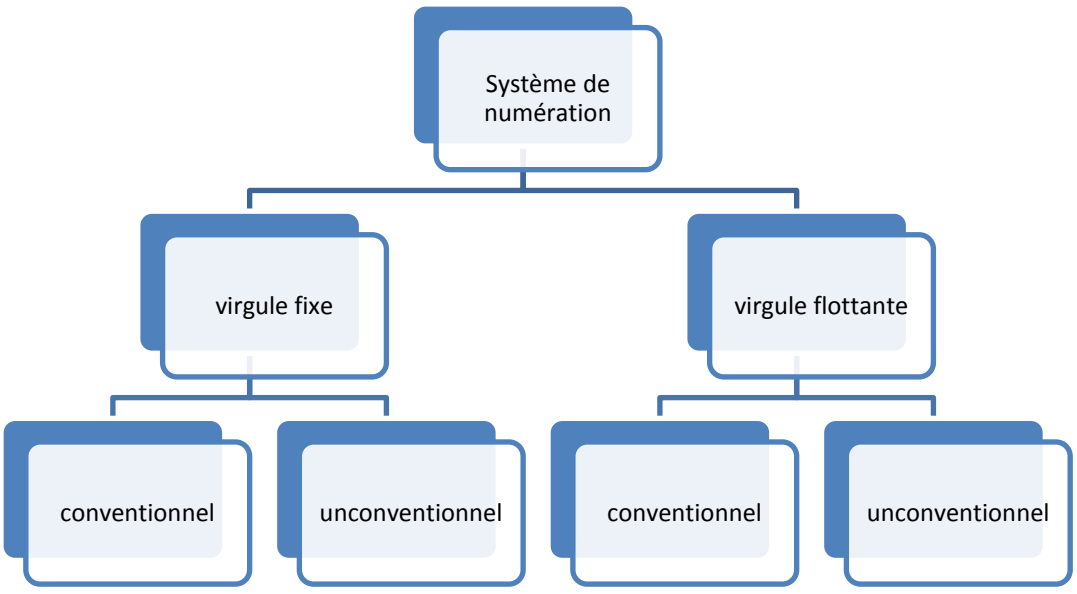
3.1 Introduction :

Dans l'arithmétique des ordinateurs deux principaux concepts fondamentaux qui ont une grande importance : sont la représentation des nombres et la réalisation des opérations algébriques [30.32.33.34]. Nous discuterons d'abord la représentation des nombres possibles, (par exemple, virgule fixe ou virgule flottante), puis les opérations de base comme l'addition et la multiplication.

3.2 Représentation des nombres :

La Décision si on doit utiliser la représentation à virgule fixe ou en virgule flottante doit être faite soigneusement, de préférence à une phase plus tôt dans le projet. En général, la représentation en virgule fixe s'exécute à des vitesses plus élevées et à faible consommation. Tandis que pour la représentation en virgule flottante à une gamme plus élevée, qui peut être attrayante pour des algorithmes les plus compliqués.

Dans cette partie nous présentons quelques méthodes couramment utilisées afin d'exprimer des nombres négatifs et/ou positifs en base deux, les des nombres en virgule fixe et en virgule flottante.



Aperçu des représentations de nombres.

3.2.1 Les nombres en virgule fixe :

Nous devrons en premier temps voir les systèmes de numération en virgule fixe illustré dans la figure 3.1.

3.2.1.1 Représentation des nombres non signés (positifs) :

Cette représentation permet le codage binaire des nombres entiers non signés (positifs):

$$X = \sum_{n=0}^{N-1} x_n 2^n = a_{N-1}2^{N-1} + a_{N-2}2^{N-2} + \cdots + a_1 2^1 + a_0 2^0 \qquad (3.1)$$

$x_n \in [0,1]$, Le chiffre x_0 s'appelle le moins significatif (LSB) et le chiffre plus significatif (MSB) est *2N-1*.

Par exemple, pour un nombre donné en base 2 comprenant 4 bits :

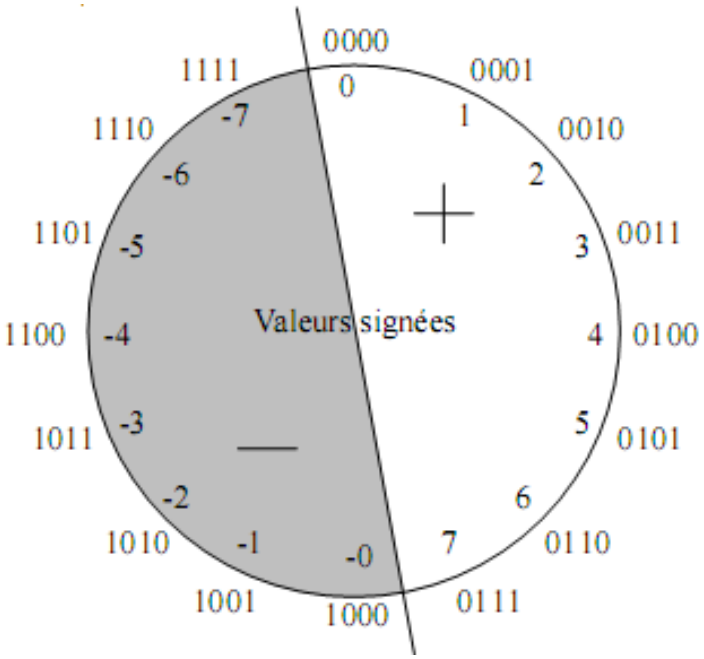
$$A_0 = (1101)_2 = \underbrace{1 \cdot 2^3}_{8} + \underbrace{1 \cdot 2^2}_{4} + 0 \cdot 2^1 + \underbrace{1 \cdot 2^0}_{1} = 13_{10} \qquad (3.2)$$

3.2.1.2 Représentation des nombres signés :

Une méthode de représentation de nombres négatifs consiste à utiliser un bit de poids fort supplémentaire codant le signe figure 3.2. Par convention, les signes + et –sont respectivement dénotés par'0'et'1'.

$$X = \underbrace{(1 - 2_{N-1})}_{Signe} \cdot \underbrace{\sum_{n=0}^{N-2} x_n 2^n}_{valeur\ absolue} = (-1)^{n-1} \cdot \sum_{n=0}^{N-2} x_n 2^n \qquad (3.3)$$

Le nombre X appartient à l'intervalle symétrique $[-(2^{N-1}-1), (2^{N-1}-1)]$ et son opposé s'obtient en inversant le bit de signe. Bien qu'intuitive et très simple, cette approche comporte quelques inconvénients. La double représentation de 0, inévitable dans un codage symétrique en base deux, complique d'une part la comparaison de deux nombres. D'autre part, ce codage implique des algorithmes complexes pour des opérations arithmétiques telles l'addition et la soustraction.



Représentation en signe et valeur absolue de nombres de quatre bits.

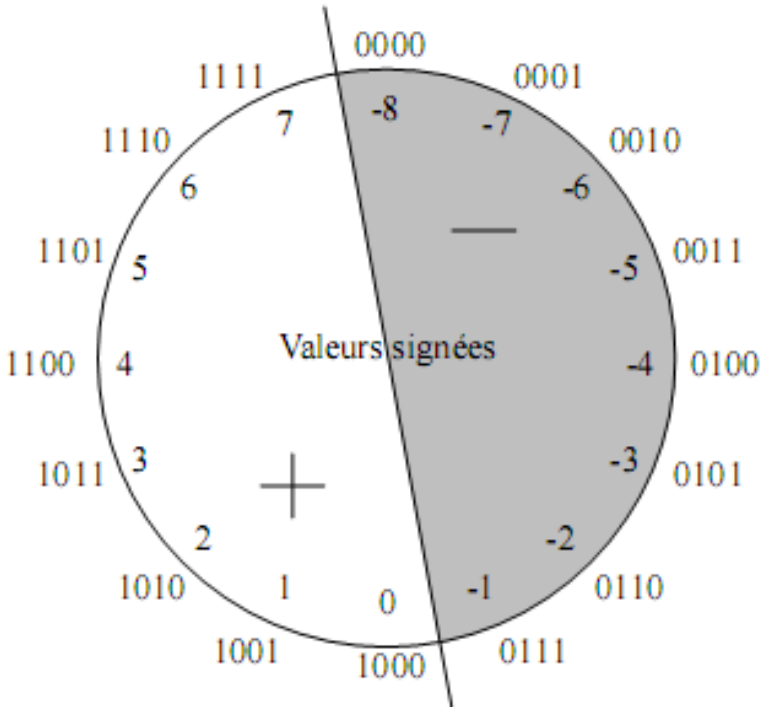
3.2.1.3 Représentations biaisées :

Le principe de cette représentation réside dans l'adjonction d'un biais R à tout nombre X appartenant à Z, de sorte que X + R soit toujours positif. X est ainsi codé par le nombre Y appartenant à N tel que :

$$Y = X + R \qquad (3.4)$$

Afin que le domaine de variation des nombres positifs et négatifs soit approximativement le même, le biais est généralement défini par $R = 2^{n-1}$ ou $R = 2^{n-1}-1$.

Nous obtenons respectivement les domaines de variation $[-2^{n1}. 2^{n1}-1] \cap Z$ et $[2^{n-1}+1, 2^{n-1}] \cap Z$. La figure 3.3 illustre le fonctionnement de ce codage pour des nombres de 4 bits avec $R = 2^3$. le bit de poids fort indique le signe du nombre. Contrairement à la représentation en signe et valeur absolue, '0' et '1' dénotent respectivement des nombres négatifs et positifs.



Représentation biaisée (R = 23) de nombres entre -8 et +7, avec des mots de quatre bits.

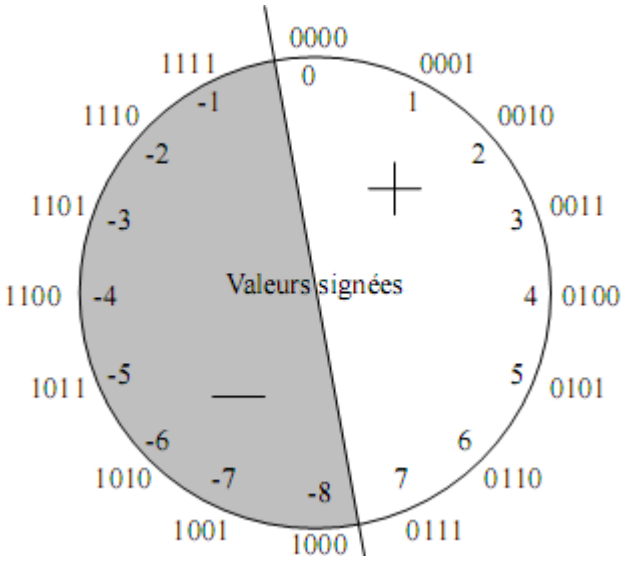
3.2.1.4 Complément a deux (2c) :

La méthode du complément à deux (two's complement 2C) (ou complément vrai) autorise le codage de nombres appartenant à $[-(2^{N-1}-1),(2^{N-1}-1)]$, et son complément sont ainsi définis par :

$$X=\begin{cases}\sum_{n=0}^{N-2} x_n 2^n & X \geq 0 \\ -2^{N-1}+\sum_{n=0}^{N-2} x_n 2^n & X<0\end{cases} \qquad (3.5)$$

Le complément à deux d'un nombre s'obtient donc en inversant tous les bits du mot, puis en ajoutant '1' au résultat.

Le système compléments à deux (2C) est le système de numération signé le plus utilisé dans les DSP actuels. C'est parce qu'il est possible d'additionner plusieurs nombres signés, et la longueur du résultat final est de N-bit, où nous pouvons ignorer le débordement. Le complément à deux peut être employé pour des opérations arithmétiques modulo 2^N sans aucune modification de l'arithmétique.



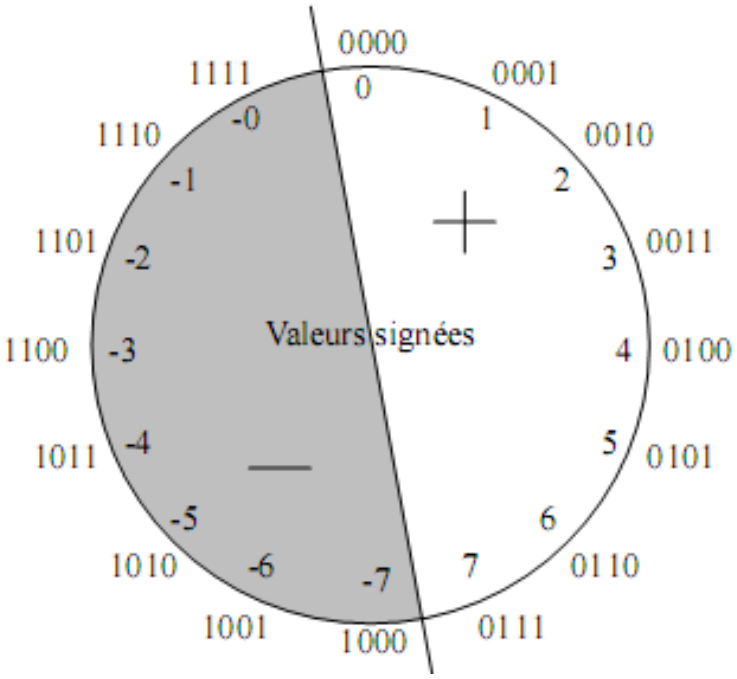
Représentation en complément à deux de nombres entre -8 et +7, avec des mots de quatre bits.

3.2.1.5 Complément a un (1c) :

Cette approche permet le codage de nombres appartenant à l'intervalle symétrique $[-(2^{N-1}+1),(2^{N-1}-1)]$ complément sont respectivement représentés par :

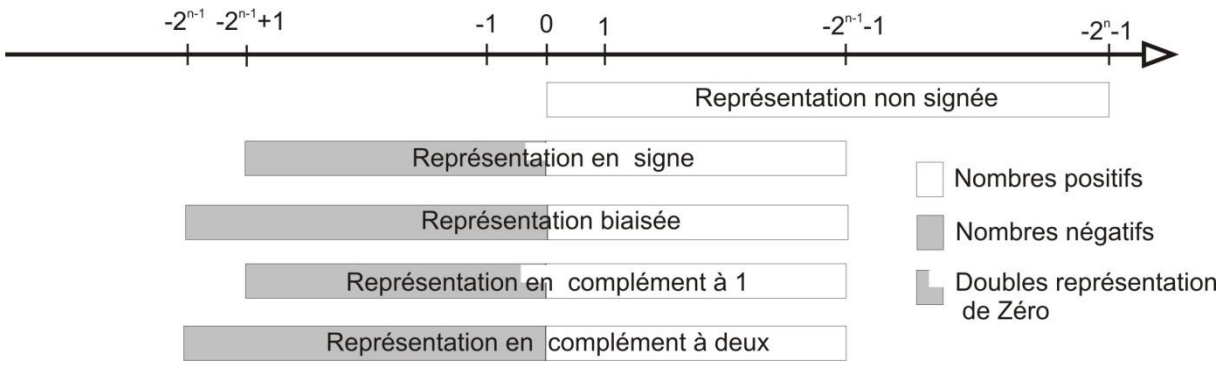
$$X=\begin{cases}\sum_{n=0}^{N-2} x_n 2^n & X\geq 0\\ -2^{N-1}+1+\sum_{n=0}^{N-2} x_n 2^n & X<0\end{cases} \qquad (3.6)$$

Par conséquent, le complément à un s'obtient par inversion de tous les bits du nombre considéré. Le domaine de variation est compris entre $-2^{n-1}+1$ et $2^{n-1}-1$ et le bit de poids fort permet la distinction entre un nombre strictement positif (bit de poids fort égal à 0) et strictement négatif (bit de poids fort égal à 1). Ce pendant, la détection de la valeur zéro se révèle plus problématique. Il existe en effet un "zéro positif" lorsque tous les bits du nombre sont égaux à 0 et un "zéro négatif" lorsque tous les bits sont égaux à 1. La figure 3.5 illustre le fonctionnement de ce codage pour des mots de quatre bits.



Représentation en complément à un de nombres entre -7 et +7, avec des mots de quatre bits.

La figure 3.6 résume les caractéristiques des diverses représentations binaires étudiées dans les paragraphes précédentes.



Résumé des diverses représentations des nombres étudiées.

3.2.1.6 Un-conventionnel en virgule fixe :

Intéressons-nous maintenant à l'examen des systèmes de numération selon Figure 3.1. Les systèmes de numération en virgule fixe peu usuels un-conventionnel discutés, ces systèmes ne sont pas souvent employés, mais ils peuvent rapporter des améliorations significatives pour des applications ou des problèmes particuliers. Par exemple le système 2C.

3.2.1.7 Nombres à chiffre signé :

Le système à chiffre signé (Signed Digit Numbers SD) diffère des systèmes binaires traditionnels présentés dans la section précédente dans le fait qu'il représente le chiffre par les valeurs {0.1, -1}, où -1 parfois peut être noté par $\bar{1}$. Prenant comme exemple le codage du nombre décimal $15=1111_2$ utilisant le codage binaire à 5-bits et le système à chiffre signé :

1) $15_{10} = 16_{10} - 1_{10} = 1000\bar{1}_{SD}$
2) $15_{10} = 16_{10} - 2_{10} + 1_{10} = 100\bar{1}1_{SD}$
3) $15_{10} = 16_{10} - 4_{10} + 1_{10} = 10\bar{1}11_{SD}$ (3.7)

La représentation, à la différence d'un code complément à deux 2C, est non-unique. Nous appelons un système à digit signé canonique (canonic signed digit system (CSD)) le système avec un nombre minimum d'éléments de non-zéro. Une autre propriété est que la représentation résultante à au moins un zéro entre deux chiffres, qui peuvent avoir les valeurs 1, -1, ou 0.

$$27_{10} = 11011_2 = 1110\bar{1}_{SD} = 100\bar{1}0\bar{1}_{CSD} \quad (3.8)$$

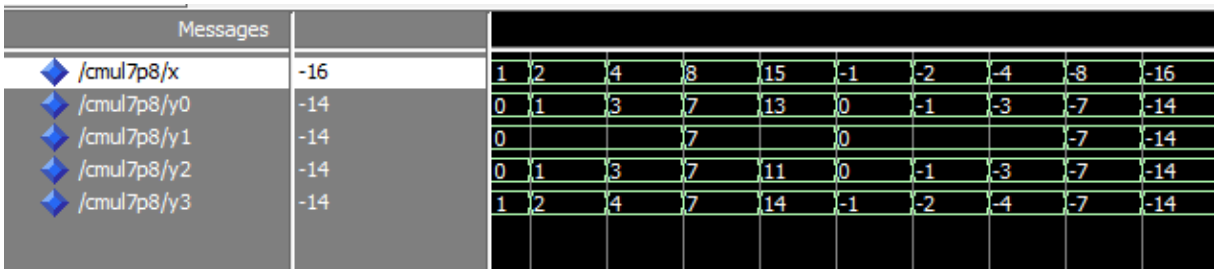
3.2.1.8 Le code (canonical signed digit) fractionnel :

La plus part des algorithmes de DSP exigent l'exécution des nombres fractionnés. Dans VHDL l'analyse d'une expression habituellement est faite de gauche à droite, qui signifie une expression comme $y = 7 \times x/8$ est implémenté en tant que $y = (7 \times x)/8$, et elle peut être aussi écrite avec une autre manière y = x/8×7 est implémenté sous la forme $y = (x/8) \times 7$. Cette dernière produira malheureusement une grande erreur de quantification, en effet $x/8$ est synthétisée par un décalage à droite par trois bits,

Considérer le codage du nombre décimal fractionné $0.875 = 7/8$ en utilisant une représentation binaire à 4-bits. Le 7 peut être mis comme $7 = 8 - 1$ la synthèse est avec différents résultats :

$$\begin{aligned} y0 &= 7 \times x/8 = (7 \times x)/8 \\ y1 &= x/8 \times 7 = (x/8) \times 7 \\ y2 &= x/2 + x/4 + x/8 = (x/2) + (x/4) + (x/8) \\ y3 &= x - x/8 = x - (x/8) \end{aligned} \quad (3.9)$$

```
1
2
3    ENTITY cmul7p8 IS ------>Interface
4    PORT( x : IN INTEGER RANGE-2**4 TO 2**4-1;
5          y0,y1,y2,y3 : OUT INTEGER RANGE -2**4 TO 2**4-1);
6    END;
7     ARCHITECTURE fpga OF cmul7p8 IS
8    BEGIN
9       y0<=7*x/8;
10      y1<=x/8*7;
11      y2<=x/2+x/4+x/8;
12      y3<=x-x/8;
13   END fpga;
```



Résultats de la simulation par ModelSim.

3.2.2 Les nombres en virgule flottante :

La représentation en format fixe présente l'inconvénient majeur de sa taille, à savoir le nombre est limité. Pour passer cette limitation les systèmes à virgule flottante ont été développés pour fournir une haute résolution élevée. La représentation en virgule flottante permet de résoudre de nombreux problèmes de représentation par rapport à la représentation en virgule fixe. Selon la position de la virgule, on peut représenter des nombres très petits ou très grands. C'est donc un système de représentation des nombres dans lequel l'intervalle des nombres exprimables est indépendant du nombre de chiffres significatifs (la précision).

La représentation est basée sur la représentation des nombres en notation scientifique

$N = S\ M.bE$

S = indique le signe si le nombre est négatif ou positif

M = mantisse en 2* de forme 0,xxx

b = base de l'exponentiation (2 ou 16)

E = exposant en binaire décalé

Comme exemple on écrit 435,23 10^4 où 435,23 est la mantisse 4 est l'exposant. Cette représentation en virgule flottante n'est pas unique, on pourrait aussi écrire : 4352,3 10^3, 4352300 ici l'exposant est 0, 0,43523 10^7. Il faut donc les représenter sous forme normalisée afin que la représentation ne varie pas d'une application à une autre.

La forme normalisée est donc un nombre dans lequel le chiffre qui précède la virgule est un 0 et le chiffre qui suit la virgule n'est pas un 0. Si on prend l'exemple précédant 0,043523 10^8 n'est pas normalisé. Par contre 0,43523 10^7 est normalisé et est stocké : 43523 comme mantisse et 7 comme exposant.

Selon la précision exigée, on utilise :

- une représentation 16 bits (précision réduite),
- sur 64 bits (double précision),
- ou sur 32 bits (simple précision)

La plupart des systèmes en virgule flottante implémenté sur des microprocesseurs sont conformes au standard flotter-points d'IEEE [35.36], tandis que sur les FPGAs on utilise souvent des formats personnalisés.

Ils sont arrangés comme suit :

	le signe S	**L'exposant E**	**La mantisse M**
16 bits	±	$a_1a_2 \ldots a_8$	$b_1b_2 \ldots b_{23}$
32 bits	±	$a_1a_2 \ldots a_{11}$	$b_1b_2 \ldots b_{52}$
64 bits	±	$a_1a_2 \ldots a_{15}$	$b_1b_2 \ldots b_{64}$

Le standard IEEE 754 [36] est le standard le plus utilisé aujourd'hui pour représenter les nombres en virgule flottante.

3.3 Operateurs arithmétiques spécifiques pour les FPGAs :

Nous nous intéressons à l'implémentation électronique des algorithmes de traitement du signal. L'implantation de ces algorithmes font souvent appel à des opérations de calcul tels que l'addition, la multiplication, la division .... Dans ce chapitre nous présentons différentes architecctures d'additionneurs et de multiplieurs. Une étude comparative est effectuée au niveau de l'intégration électronique en technologie FPGA et ce en vue de l'optimisation de l'implémentation d'un algorithme de débruitage.

2.3.1 Introduction

Dans plusieurs traitements de signaux discrets, comme les corrélations, les convolutions, le filtrage et l'analyse fréquentielle, l'addition et la multiplication sont des opérateurs indispensables [37]. De ce fait, plusieurs algorithmes d'addition et de multiplication ont été étudiés et implémentés sous plusieurs formes: intégrés dans des DSP, circuits intégrés ou même conçus dans des circuits programmables. L'évolution des technologies de l'intégration des circuits électroniques a fait que les recherches dans le domaine architectural se fait de plus en plus poussée. Ces recherches visent l'amélioration des performances temporelles des éléments à intégrer ainsi que la diminution de la surface de silicium occupée. Ce chapitre concerne l'étude architecturale des additionneurs et des multiplieurs en vue de l'implémentation matérielle de notre l'algorithme de débruitage.

Les critères d'évaluation de l'architecture pour chaque opérateur sont basés sur :

- le retard global entre l'instant d'acquisition de la donnée et l'instant final du traitement de l'opération.

- Occupation en unités élémentaires du FPGA occasionnée par l'implantation électronique de l'architecture de l'opérateur considéré. Ceci représente la taille en portes (voir en surface de Silicium) de l'opérateur.

2.3.2 Addition binaire

L'addition est une opération très courante dans le traitement numérique du signal (DSP). Il est donc important qu'elle soit optimisée pour être rapide. Malgré la simplicité apparente du problème, il existe de multiples façons de construire des additionneurs efficaces en temps et en nombre de portes logiques utilisées.

2.3.2.1 Semi additionneur

Ce premier circuit est la brique de base. Il prend en entrée deux bits A et B et calcule la somme S et la retenue C_{out} (pour Carry en anglais). Les bits C_{out} et S peuvent aussi être vus comme les bits de poids fort et de poids faible de l'écriture sur deux bits de la somme A + B.

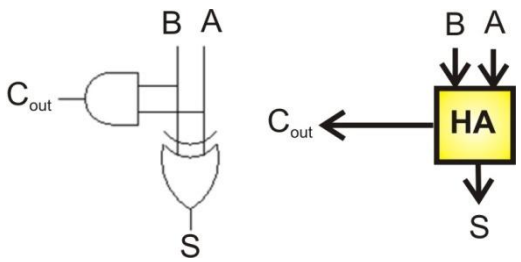


Schéma et symbole d'un Semi Additionneur (half-adder)

2.3.2.2 Additionneur complet

Pour construire un additionneur sur plusieurs bits, plusieurs additionneurs 1 bit sont mis en cascade. Chacun de ces additionneurs prend en entrée deux bits A et B ainsi que la retenue précédente C_{in}. Il calcule la somme S de ces trois valeurs binaires ainsi que la retenue C1. Comme pour le semi-additionneur, ces bits C_{out} et S peuvent aussi être vus comme les bits de poids fort et de poids faible de l'écriture sur deux bits de la somme A + B + C_{in}.

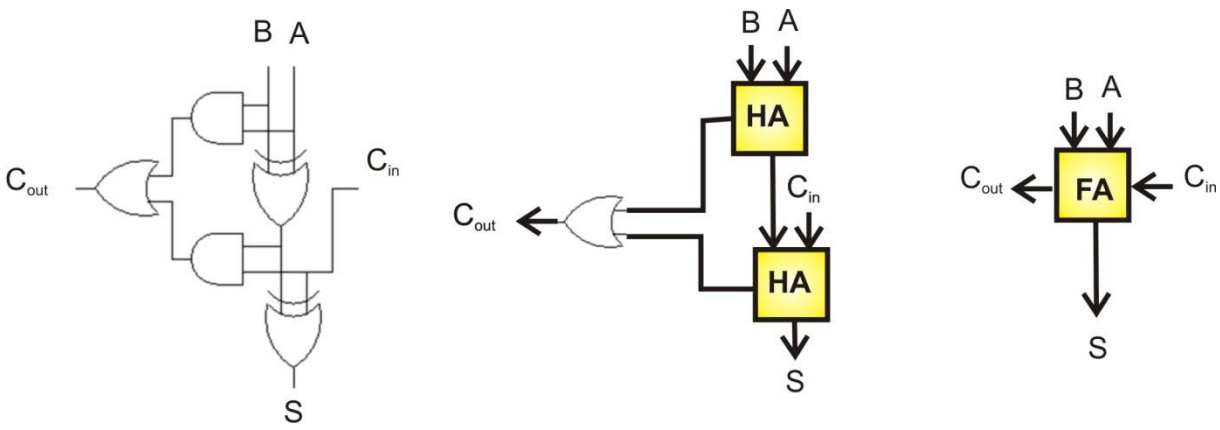


Schéma et symbole d'un Additionneur Complet (Full-adder)

2.3.2.3 Additionneur par propagation de retenue

Etudions l'exemple de l'addition binaire de deux nombres positifs X et Y de huit bits. Lorsque nous effectuons ce calcul à la main, nous additionnons tout d'abord les deux bits de poids faible et déterminons le bit S_0 du résultat ainsi qu'un bit de retenue C_1.

$$S_i = X_i \oplus Y_i \qquad (3.10)$$

$$C_{i+1} = X_i Y_i \qquad (3.11)$$

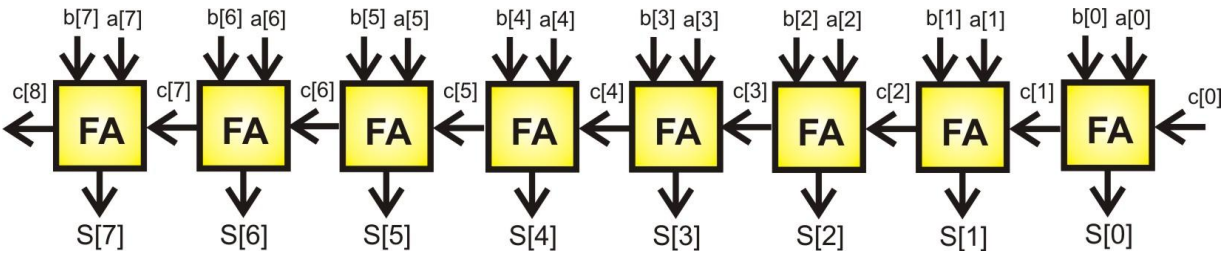
Nous traitons séquentiellement les colonnes restantes, de droite à gauche, en additionnant X_i, Y_i et la retenue entrante C_i afin de générer un bit de somme S_i ainsi qu'une retenue sortante C_{i+1} définis par l'équation suivante:

$$S_i = X_i \oplus Y_i \oplus C_i \qquad (3.12)$$

$$C_{i+1} = X_i Y_i + X_i C_i + Y_i C_i \qquad (3.13)$$

La figure 3.10 illustre la structure d'un additionneur à retenue propagé (Carry-Propagate Adder ou CPA) de deux nombres sur huit bits. Il est constitué d'une cellule half-adder et de 7 (n-1) cellules full-adder connectées en cascade. La retenue sortante C_n indique

un éventuel dépassement de capacité. Sa gestion dépend du système dans le quel intervient l'additionneur.



La structure d'un additionneur à retenue propagé

La propagation de retenue implique théoriquement un temps de calcul proportionnel à la taille des opérandes. Ainsi, en doublant la précision de l'additionneur, nous diviserions sa fréquence de fonctionnement par deux.

2.3.2.4 Additionneur par anticipation de retenue

La lenteur de l'additionneur par propagation de retenue impose d'utiliser d'autres techniques pour des additionneurs ayant un nombre important de bits. Comme cette lenteur est due au temps nécessaire à la propagation de la retenue. Remarquons toute fois que les entrées x_i et y_i d'une cellule full-adder sont disponibles bien avant la retenue entrante c_i que, toutes les techniques ont pour but d'accélérer le calcul des retenues.

La première technique appelée *anticipation de retenue* ou consiste à faire calculer les retenues par un circuit extérieur.

Afin de faciliter le calcul des retenues, on introduit deux quantités appelées G (*pour Generate en anglais*) et P (*pour Propagate en anglais*). Pour deux quantités binaires A et B, les quantités G et P sont définies de la façon suivante.

$$P = A + B \quad et \quad G = A \cdot B \qquad (3.15)$$

Soient $A = A_{n-1}\ldots A_0$ et $B = B_{n-1}\ldots B_0$ deux entrées de n bits. On note C_i la retenue de l'addition des i bits de poids faible de A et B. Pour accélérer le calcul des C_i, on introduit les deux quantités G_i et P_i associées aux entrées A_i et B_i par les formules suivantes.

$$P_i = A_i + B_i \quad et \quad G_i = A_i \cdot B_i \qquad (3.16)$$

La valeur G_i est la retenue engendrée par l'addition des deux bits A_i et B_i et la valeur de Pi détermine si la retenue de C_i se propage. On a donc la formule suivante qui exprime

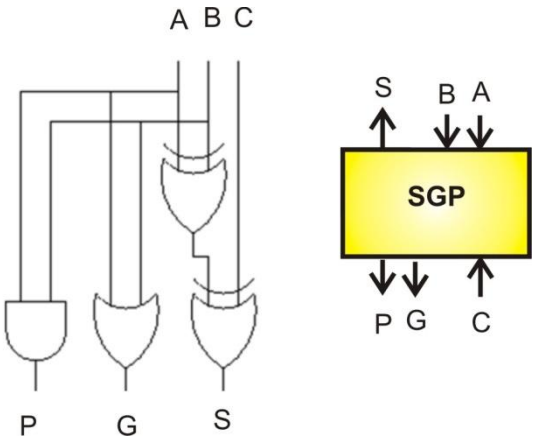
simplement que la retenue C_{i+1} provient soit directement de l'addition des bits A_i et B_i soit de la propagation de la retenue C_i.

$$G_{i+1} = G_i + P_i C_i \qquad (3.17)$$

En utilisant plusieurs fois cette formule, on peut obtenir les formules suivantes qui expriment C_{i+1} en fonction d'une retenue précédente et des valeurs G_j et P_j intermédiaires.

$$C_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} C_{i-1} C_{i+1} \qquad (3.18)$$

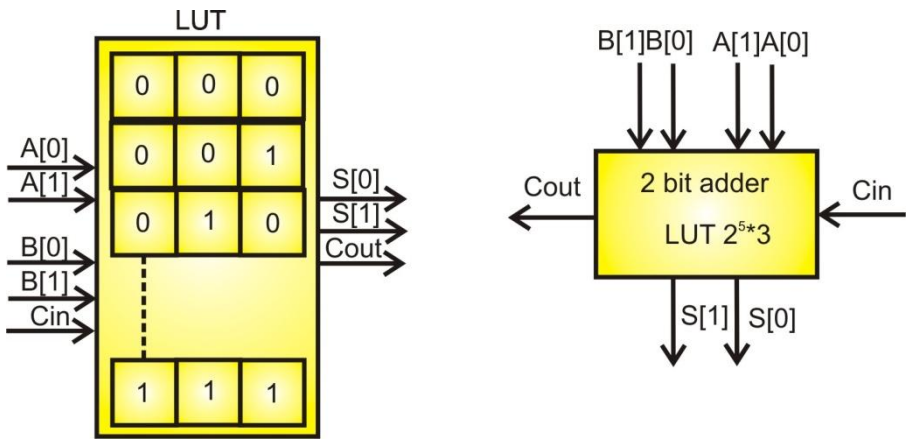
Le circuit de la figure 3.11 permet de calculer la somme ainsi que les deux quantités G_i et P_i.



La structure d'une cellule S, G et P.

2.3.2.5 Additionneur par l'utilisation d'une LUT :

L'idée générale pour remédier le problème du temps de calcule de retenue et de la complexité de la réalisation. On s'oriente à utilisé une LUT qui regrouper l'ensemble de bits dans une LUT comme il est illustré dans la figure 3.11. La taille de la LUT dans ce cas est de 2^5, le vecteur de sortie et de 2^3.



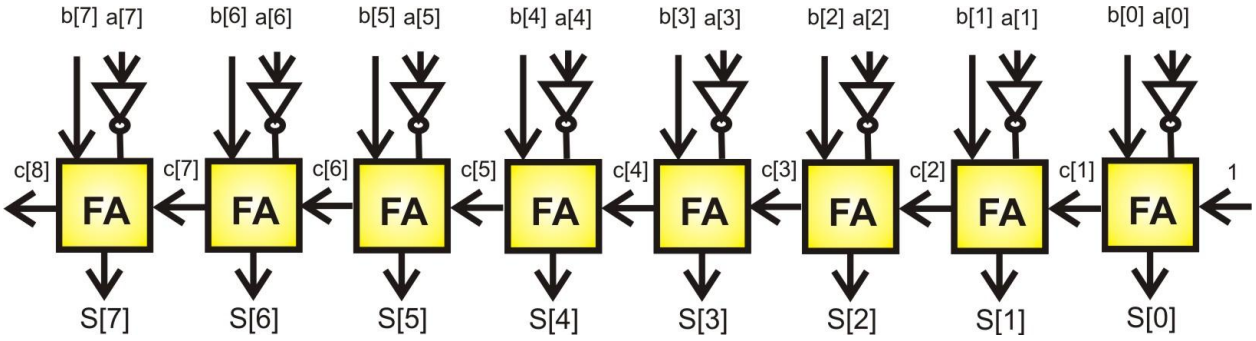
La structure d'un Additionneur avec une LUT

2.3.2.6 L'addition et la soustraction en complément à deux :

Lorsque deux opérandes en complément à deux comportent un nombre de chiffres identique, leur addition s'effectue à l'aide du circuit à retenue propagée étudié précédemment. Considérons maintenant deux nombres X etY possédant respectivement n et $n+k$ bits ($k \geq 1$) . Il suffit de représenter X avec $n+k$ chiffres afin que les opérandes soient codés à l'aide du même biais $R=2^{n+k}$. Cette opération consiste simplement à étendre le bit de signe de k positions vers la gauche. Si X est négatif. Dans le cas où X est positive, aucune correction n'est nécessaire car le biais n'est appliqué qu'aux nombres négatifs. L'extension du bit de signe n'a aucune incidence sur la valeur codée par X. Une fois cette opération réalisée, un additionneur à retenue propagée détermine la somme X+Y. La figure 3.12 présente un soustracteur, obtenu en modifiant l'additionneur à retenue propagée, destiné à des opérandes en complément à deux. Comme

$$X - Y = X - \bar{Y} + 1 \qquad (3.19)$$

Il suffit donc à inverser l'ensemble des bits de Y et de substituer à la cellule half-adder une cellule full-adder dont le retenue est fixée à 1.



La soustraction en complément à deux réalisée avec un additionneur à retenue propagée.

2.3.3 La multiplication :

La multiplication est une opération très simple car elle se réalise par l'addition répétée. Considérons X et Y, deux nombres entiers non signés de n chiffres. Leur produit Z, un nombre comportant 2n chiffres, s'obtient par exemple en calculant

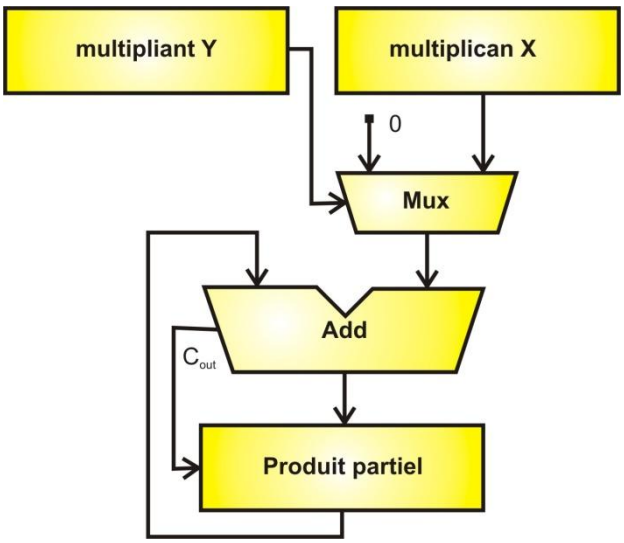
$$Z = XY = \sum_{i=0}^{n-1} y_i X 2^i \qquad (3.20)$$

Remarquons que ce procédé de multiplication, illustré par la figure3.13, correspond à la méthode que nous utilisons pour calculer à la main le produit de deux nombres.

```
    42
  X 27
   294
  84
 1134
```

La multiplication traditionnelle de deux nombres.

La multiplication binaire non signé (ou complément à deux) elle s'exécute exactement de la même manière, mais plus facile parce que les chiffres du multiplicateur sont un ou zéro. Cela signifie que les produits partiels sont des zéros ou une copie du multiplicateur, décalé à gauche convenablement. C.a.d que la multiplication utilise simplement le décalage et l'addition (shifters and adders). La figure 3.14 illustre une réalisation matérielle possible de ce multiplicateur.

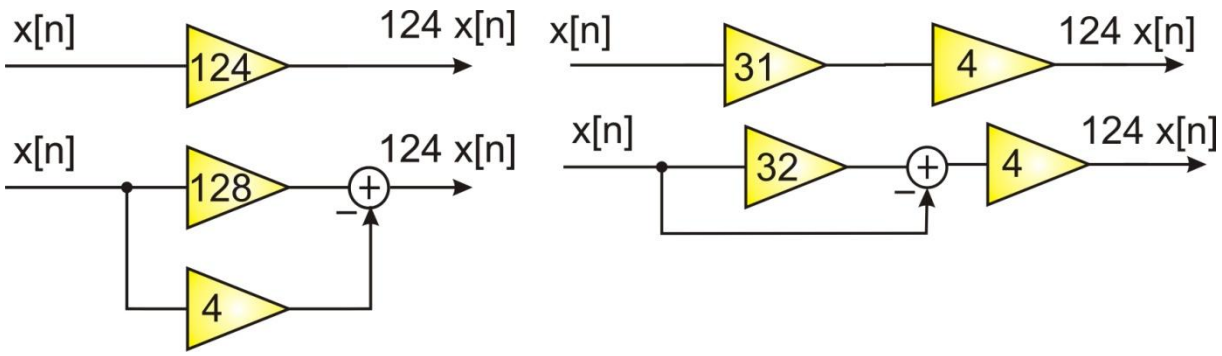


Le multiplicateur basé sur l'algorithme de décalage et d'addition

2.3.3.1 La multiplication graphique :

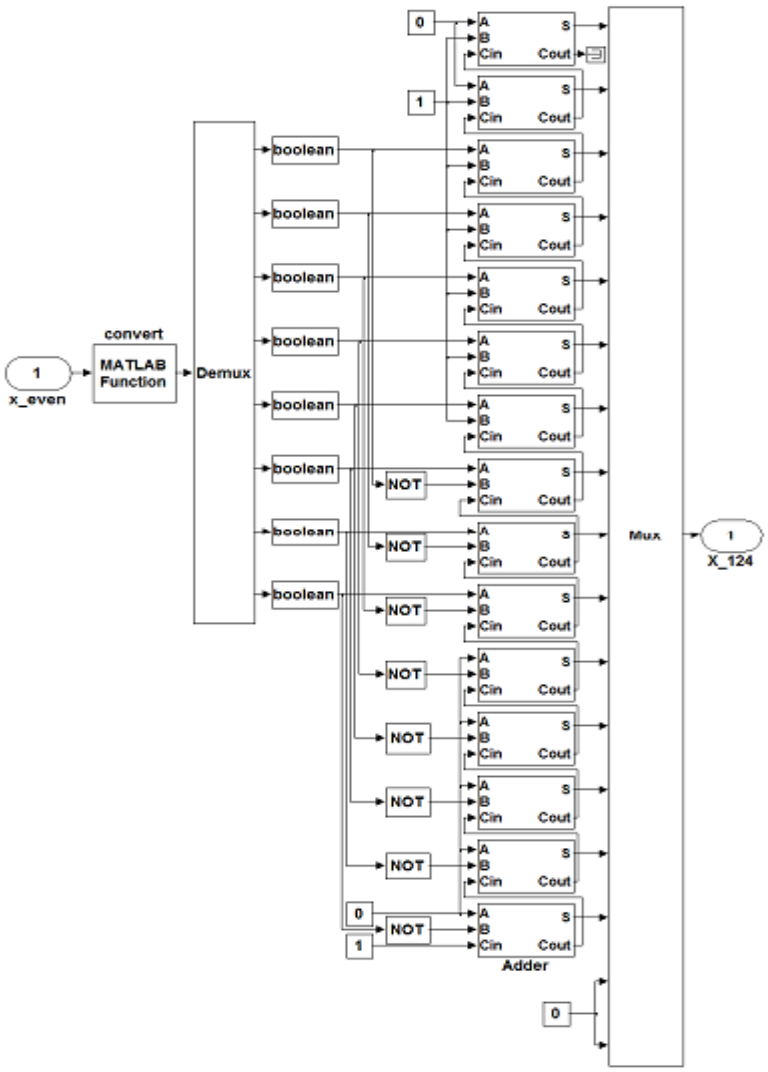
Nous avons constaté que le coût de la multiplication est une fonction directe du nombre d'éléments non-nuls a_k Dans A. Le système CSD minimise ce coût. Le CSD est également la base pour le multiplicateur de Booth [38] et de réaliser les facteurs individuels

dans un sens optimal CSD [39,40,41,42]. La figure 2.3 illustre cette option pour le coefficient de 124 [43]. Le code binaire le code CSD sont donnés par $124_{10} = 01111100_2 = 10000100_{CSD}$



Les deux réalisations pour la constante 124.

L'opération de la multiplication par le coefficient 124 nécessite le décalage à gauche de cinq bits vers le bit le plus significatif (MSB) de la donnée. Suivie par la soustraction de la donnée du résultat obtenu du décalage et le bit de signé est propagés sur le reste des bits. Le résultat obtenu doit être décalé vers la gauche par deux bits pour assurer la multiplication par 2. La figure 3.17 illustre une telle structure, notons ici que l'un des opérandes doit être une constante.



Le multiplicateur avec un coefficient 124 [43]

2.3.3.2 Multiplication d'accumulateur (MAC) et somme de produit (SOP) :

Les algorithmes de traitement numérique de signal (DSP) sont reconnus comme des multiplications à accumulation (MAC) intensive. Pour illustrer, considérons la somme de convolution linéaire donnée par :

$$y[n] = f[n] * x[n] = \sum_{K=0}^{L-1} f[k]x[n-k] \quad (3.21)$$

Cette opération exige L multiplications consécutives et L-1 addition par échantillon de y[n] pour calculer la somme des produits (SOP). Ceci suggère qu'une *N x N* bits multiplicateur doivent être fusionnés avec un accumulateur, voir Figure 3.18. Une précision de $N \times N$ bits est de taille de *2N* bits. Si les deux opérandes sont des nombres signés de même taille, le

produit n'aura que *2N-1* bits significatifs, à savoir les deux bits de signe. L'accumulateur, et afin de maintenir suffisamment de gamme dynamique, est souvent conçu pour avec *K* bits supplémentaires.

Une approche alternative à la MAC pour calculer une somme de produit sera discutée dans la section suivante :

2.3.3.3 La multiplication distribue

L'Arithmétique distribuée pour (Distributed Arithmetic DA) est une technique importante pour les FPGA. Elle est énormément utilisée dans le calcul de la somme des produits :

$$y = \langle c, x \rangle = \sum_{n=0}^{N-1} c[n] \times x(n) \qquad (3.22)$$

En plus de la convolution, la corrélation, TFD et la plupart de calcul numérique peut être représenté en tant que somme de produit (Sum Of Produits). Pour exécuter un cycle de filtrage, lorsqu'on utilise une unité arithmétique classique, on prend d'environ un *N MAC* cycles. Ce temps peut être réduit avec le pipelining, mais peut néanmoins être extrêmement longtemps. Il s'agit d'un problème fondamental lorsque les multiplicateurs d'usage général sont utilisés.

Pour des nombreuses DSP applications, la multiplication n'est pas nécessaire techniquement. Si les coefficients de filtre c [n] sont connus a priori, alors techniquement, le terme de produits partiels c [n] x [n] devient une multiplication par une constante. C'est une différence importante et est une condition préalable pour une conception d'une arithmétique distribué DA.

La première discussion de l'arithmétique distribuée remonte à un article par Croisier en 1973 [44] et DA a été popularisée par Peledand Liu [45]. Yiu [46] a étendu DA pour les nombres signés, et Kammeyer [47] et Taylor [48] ont étudié les effets de quantification dans les systèmes de l'arithmétique distribué. Tutoriaux des arithmétiques distribuées sont disponibles à partir de White [49] et Kam-meyer [50]. L'arithmétique distribuée est également abordé dans les manuels [51.52]. Pour comprendre le paradigme de la conception de l'arithmétique distribué, considère la somme des produits produit scalaire ci-dessous:

$$y = \langle c, x \rangle = \sum_{n=0}^{N-1} c[n] \times x(n)$$

$$= c[0]x[0] + c[1]x[1] + \cdots + c[N-1]x[N-1] \qquad (3.23)$$

Supposons en outre que les coefficients *c [n]* sont des constantes connues et *x [n]* est une variable. Un système d'arithmétique distribuée non signé suppose que la variable *x [n]* est représenté par:

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \times 2^n \qquad (3.24)$$

$$\text{avec } x_b[n] \in [0,1],$$

où $x_b[n]$ désigne le bit *b* de *x [n]*, soit le n$^{\text{iém}}$ élément de x. Le produit intérieur y peut, par conséquent, être représenté comme suit:

$$y = \sum_{n=0}^{N\_1} c[n] \times \sum_{b=0}^{B-1} x_b[n] 2^b \qquad (3.25)$$

Redistribution de l'ordre de sommation (d'où le nom arithmétiques distribués) résultats dans:

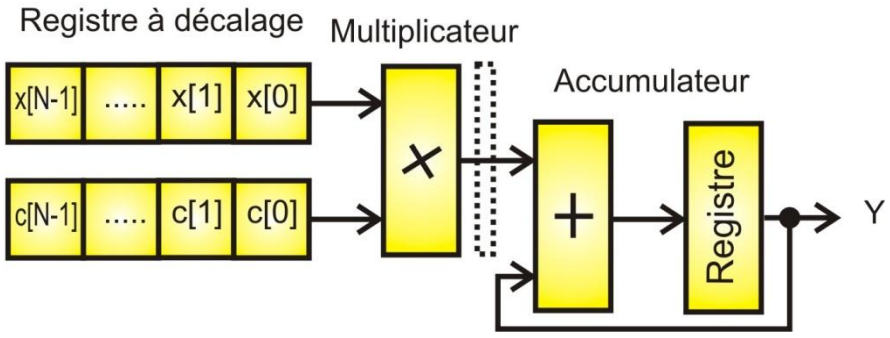
$$\begin{aligned} y = {} & c[0](x_{B-1}[0]2^{B-1} + x_{B-2}[0]2^{B-2} + \cdots x_0[0]2^0) + \\ & + c[1](x_{B-1}[1]2^{B-1} + x_{B-2}[1]2^{B-2} + \cdots x_0[1]2^0) \\ & \vdots \\ & + c[N-1](x_{B-1}[N-1]2^{B-1} + x_{B-2}[N-1]2^{B-2} + \cdots x_0[N-1]2^0) \\ & = (c[0]x_{B-1}[0] + c[1]x_{B-1}[1] + \cdots + c[N_1]x_{B-1}[N-1])2^{B-1} \\ & + (c[0]x_{B-2}[0] + c[1]x_{B-2}[1] + \cdots c[N_1]x_{B-2}[N-1])2^{B-2} \\ & \vdots \\ & + (c[0]x_0[0] + c[1]x_0[1] + \cdots + c[N\_1]x_0[N-1])2^0 \qquad (3.26) \end{aligned}$$

Ou bien sous une forme plus compacte :

$$y = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} \underbrace{c[n] \times x_b[n]}$$

$$y = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n] \times x_b[n]) \qquad (3.27)$$

La figure illustre une telle structure.



La structure d'une multiplication distribue

Exemple :

Un produit scalaire de troisième ordre est défini par l'équation de produits scalaires suivante :

$$y = \langle c, x \rangle = \sum_{n=0}^{2} c[n] \times x[n] \quad (3.28)$$

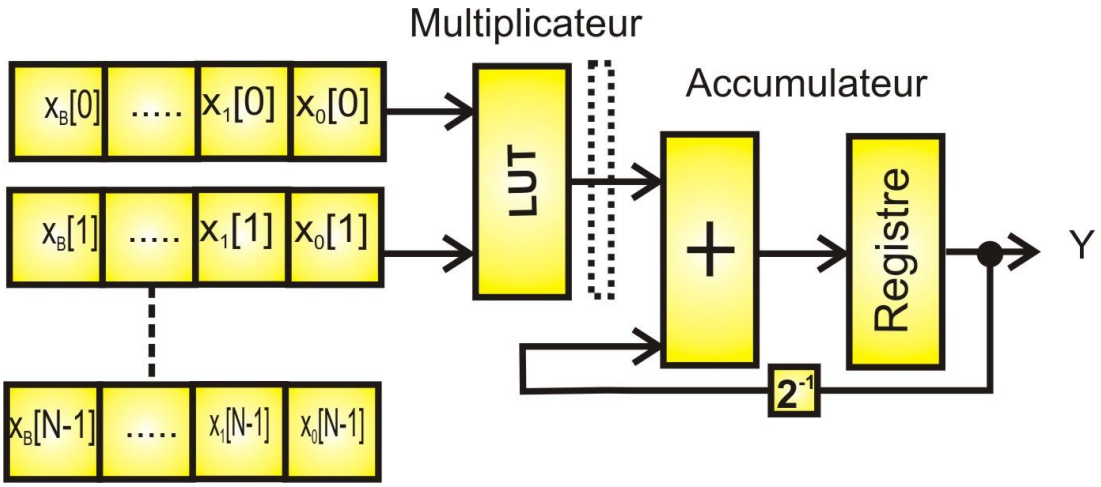
Supposons que *c[0]=2, c[1]=3, c[2]=1*. Le produit scalaires avec *x[n]={ x[0]=1, x[1]=3, x[2]=7}*, est obtenu par le calcul suivant :

$$y = \langle c, x \rangle = c[0]x[0] + c[1]x[1] + c[2]x[2]$$

$$= 2 \times 1 + 3 \times 3 + 1 \times 7 = 18$$

Pour remédier à toute utilisation des multiplicateurs qui sont gourmand en élément logique et en temps d'exécution ainsi que l'un des multipliant est une constante. L'implémentation sur FPGA de la fonction *f (c[n], x[n])* nécessite une réflexion particulière. La méthode de mise en œuvre préférée est de réaliser la cartographie *f (c[n], x[n])* en utilisant une look-up tables (LUT). La LUT contient 2^N mots est préprogrammée pour accepter un vecteur d'entrée de N-bits $x_b=[x_b[0], x_b[1],\ldots,x_b[N-1]]$, et une sortie f (c [n], x_b[n]). Les mappages individuels *f (c[n], x[n])* sont pondérés par la puissance appropriée de deux

facteurs et accumulé. L'accumulation peut être implémentée efficacement comme il est indiqué dans figure 3.18. Après N cycles de calcule le résultat du produit scalaire est obtenu.



La structure d'une multiplication distribue à base d'une LUT

2.3.3.4 Look-Up Table (LUT) Multiplication

La Multiplication on utilisant une table de correspondance (LUT) est alternatif utilisée pour implémenté des opérations mathématiques sur FPGA, elle est tout simplement un bloc de mémoire contenant le résultat de multiplication complète des deux opérandes il contient donc toutes les combinaisons possibles de cette multiplication. Les dimensions de la table requis est considérables, et délicats, ce qui rendre le technique impraticables pour les FPGA de cette manières.

REPRESENTE LE CONTENU D'UNE LUT A 6 ENTREES POUR 3 X 3 BITS

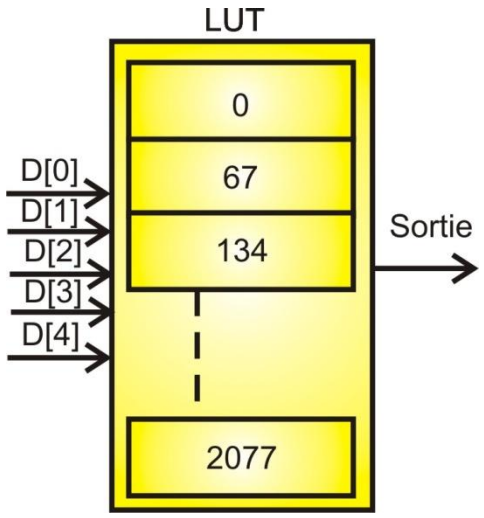
	000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	000000	000000	000000	000000	000000	000000	000000	000000
001	000000	000001	000010	000011	000100	000101	000110	000111
010	000000	000010	000100	000110	001000	001010	001100	001110
011	000000	000011	000110	001001	001100	001111	010010	010101
100	000000	000100	001000	001100	010000	010100	011000	011100
101	000000	000101	001010	001111	010100	011001	011110	100011
110	000000	000110	001100	010010	011000	011110	100100	101010
111	000000	000111	001110	010101	011100	100011	101010	110001

Pour une multiplication où un des multipliant est une constante, cette technique est beaucoup plus efficace sur FPGA, il se fait de construire une table de multiplication qui ne dispose que d'une seul colonne correspondant à la valeur du résultat de la multiplication pré-calculé. Il est connu comme une multiplication par un coefficient constant (KCM). L'exemple ci-dessous multiplie une entrée à 5-bits (valeurs de 0 à 31) par une constante 67. Noter

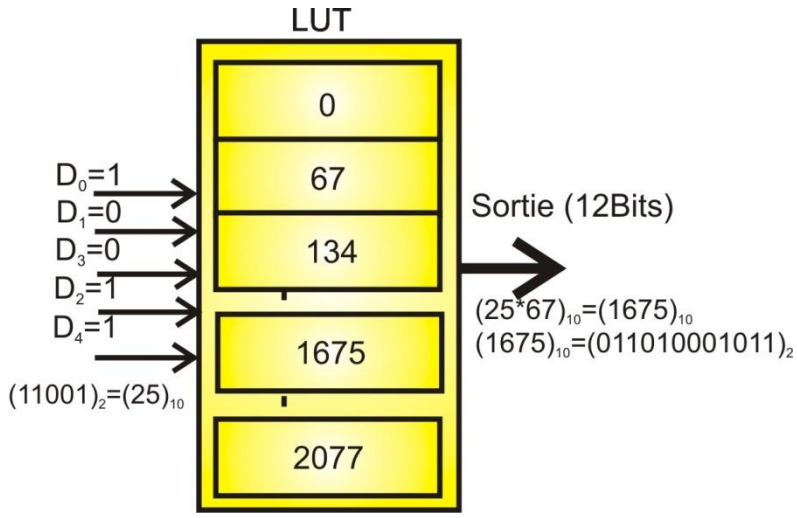
qu'avec un multipliant est une constante, toutes les entrées sont disponibles sur LUT pour le multiplicande variable. Cela rend la KCM plus efficace qu'une multiplication complète (moins de produits partiels pour une largeur donnée). C'est une technique qui est recommandée sur FPGA car elle réduit le nombre des éléments logiques, elle réduit aussi le temps d'exécution qui le rend plus efficace en traitement du signal.

REPRESENTE DE LA LUT (ENTREE (5 BIT) * 67)

	Entrée (5 bits) * 67			
Entrée	**00**	**01**	**10**	**11**
000	0	536	1072	1608
001	67	603	1139	1675
010	134	670	1206	1742
011	201	737	1273	1809
100	268	804	1340	1876
101	335	871	1407	1943
110	402	938	1474	2010
111	469	1005	1541	2077



Le schéma bloc d'une LUT (Entrée (5 bits) * 67, sortie(15 bits))



Le schéma bloc d'une LUT (Entrée = 25, sortie = 1675)

Chapitre IV

Théories de la transformée en Ondelettes et la notion de débruitage

4. THEORIES DE LA TRANSFORMEE EN ONDELETTE

4.1. Introduction :

Le traitement du signal a pour but principal objet la description des signaux liés au monde réel dans but de traitement, d'identification, de compression, de compréhension ou de transmission. Dans ce contexte, les transformations linéaires ont toujours joué un très grand rôle, et parmi ces dernières la transformation de Fourier (1822). Cette transformation permet d'explorer la composition fréquentielle du signal et par ses propriétés de lui appliquer facilement des opérations de filtrage. Mais, très tôt dans l'histoire du traitement du signal, il est apparu que la décomposition obtenue n'était pas toujours la plus satisfaisante et la première transformation en ondelettes est proposée par Haar en 1910 ; cette transformation découpe le signal en composantes fréquentielles suivant une résolution adaptée à l'échelle. Le premier à avoir utilisé cette technique et le premier à avoir proposé le nom d'ondelettes fut Jean Morlet (1983). Dans l'analyse de données issues de soudages sismiques effectués pour des recherches géologiques. Apres des bases mathématiques solides ont été mises en place faisant apparaître la notion de base orthogonale (Yves Meyer 1985), d'analyse multirésolution (S. Mallat 1989 [53,54,55]) et d'ondelettes à support compact (I. Daubechies 1988 [56]).

Bien que l'analyse par ondelettes ait été introduite dans les années 1970 par Yves Meyer [57, 58] dans un contexte d'analyse du signal et d'exploration pétrolière, et par la suite reprise par des théoriciens et ingénieures comme Morlet dans le début des années 80 qui on fait une véritables théorie et outil mathématique [59], elles sont néanmoins un outil récent du point de vue de leur applicabilité dans divers domaines. Leur propriété essentielle réside dans leur capacité à analyser à plusieurs échelles de temps, et définir des propriétés locales de signaux complexes pouvant présenter de zones non stationnaires. Leur champ d'applications s'élargit de plus en plus. Les domaines tels que la géophysique l'astrophysique, les télécommunications, l'imagerie et le codage vidéo utilisent les ondelettes comme nouvelle technique d'analyse et de synthèse du signal. Toute fois, il faut aussi souligner que l'utilisation des ondelettes a connu beaucoup de succès dans deux domaines en particulier, celui du débruitage et celui de la compression. Deux meilleurs exemples d'applications des ondelettes sont le stockage numérique d'empreintes digitales effectué par le FBI qui ont utilisé les ondelettes comme un outil de compression et de débruitage [60], et la norme de compression JPEG 2000 [61]

Figure 4. 10 :

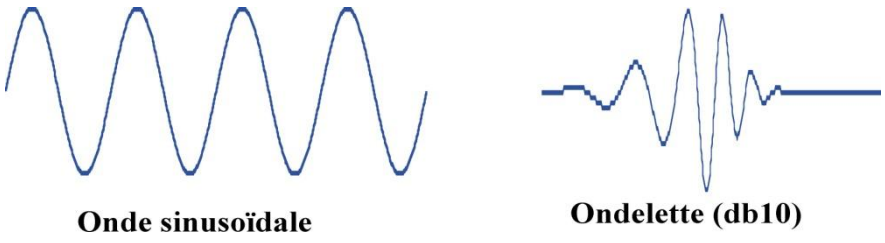
Analyse temps-échelle par la transformée en ondelette.

4.2. Transformée d'une ondelette :

Dans la plupart des applications de traitement de signal, la transformation fréquentielle du signal est très importante. La transformée de Fourier est la plus utilisée pour obtenir le spectre de fréquence d'un signal. Les signaux de la parole, d'image et la majorité des signaux biologiques ont différentes caractéristiques temporelles et fréquentielles, par exemple, ils ne sont pas stationnaires, et c'est justement dans leurs caractéristiques (statistiques, fréquentielles, temporelles, spatiales) que réside l'essentiel de l'information qu'ils contiennent. Une transformation qui renseigne sur le contenu fréquentiel tout en préservant la localisation afin d'avoir une représentation temps-fréquence est nécessaire et indispensable pour les analyser. .

La transformée en ondelettes est conçue pour être adaptative : elle consiste à analyser le signal à l'aide d'une fonction bien localisée, de moyenne nulle, qu'on appelle ondelette, que l'on translate sur tout le signal et que l'on peut dilater. Ils existent autant de livres de référence ou le lecteur pourra se documenter sur la transformée en ondelettes.

Une ondelette [62] est une fonction qui oscille comme une onde mais qui est rapidement atténuée d'où son nom ondelette qui veut dire petite onde.



L'onde sinusoïdale est symétrique et régulière et l'ondelette est une vibration asymétrique et irrégulière

Elle est localisée à la fois en temps et en fréquence et permet de définir par translation en temps et dilatation en échelle, une famille de fonction analysantes. Les ondelettes sont considérées comme un « zoom » mathématique permettant de décrire les propriétés d'un signal à plusieurs échelles de temps simultanément. D'une façon plus formelle, une fonction ψ est appelée ondelette si elle vérifie dans le domaine fréquentiel, la condition d'admissibilité donnée par :

$$\int_{\Re^+} \frac{\left|\hat{\psi}(\omega)\right|^2}{|\omega|} d\omega = \int_{\Re^-} \frac{\left|\hat{\psi}(\omega)\right|^2}{|\omega|} d\omega < +\infty \qquad (4.1)$$

Où $\hat{\psi}$ désigne la transformé de Fourier de ψ.

Une condition suffisante d'admissibilité et plus simple à vérifier, est donnée par :

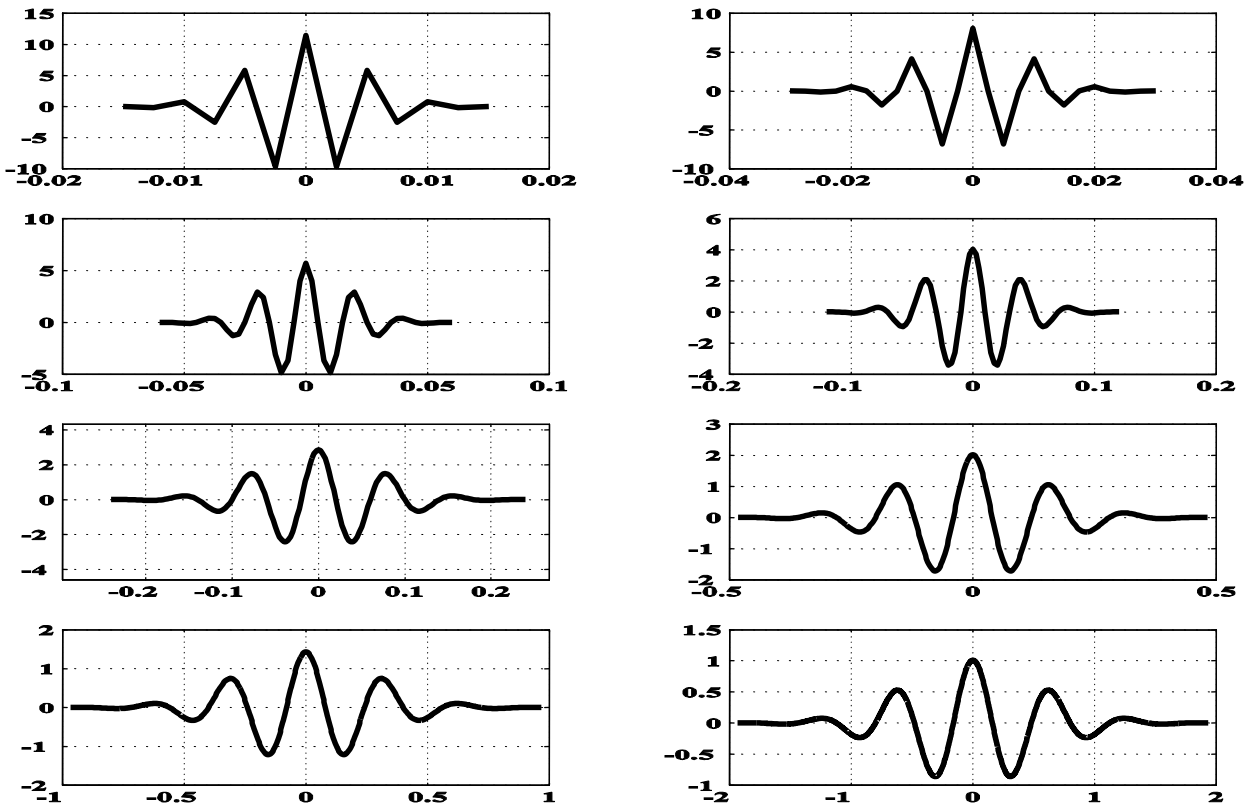
$$\int_{\Re} \psi\,(t) dt = 0 \qquad (4.2)$$

Cette condition considérée comme minimale n'est pas la seule propriété des ondelettes dont l'importance dépend souvent de l'application utilisée.

L'analyse continue par ondelettes associe une famille d'ondelettes translatées et dilatées en varient les deux paramètres a, b. cette famille se caractérise par :

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \qquad (4.3)$$

Où $(a, b) \in \Re^2$et $a>0$. Le paramètre a sert à dilater (comprimer ou étendre) la fonction ψ qui est appelée l'ondelette mère, et b sert à la translater (la déplacer selon l'axe des temps). L'idée est donc de réaliser une projection de la fonction signal sur une famille de fonctions translatées et dilatées (changement d'échelle) $\psi_{a,b}$ à partir d'une fonction unique appelée ondelette, de façon à pouvoir étudier ce qui se passe localement. Par translation et dilatation de l'ondelette ψ , on définit les atomes de la transformée par ondelette en fixant les valeurs de a et b dans l'équation (4.3). En d'autres terme, un atome de la transformée par ondelette est une fonction d'ondelette $\psi_{a,b}(t)$ unique obtenu en remplaçant chacun des paramètres a et b par une valeur fixe ($\psi_{1,20}(t)$) la figure est exemple des atomes de la transformée par ondelette.



L'évolution d'une ondelette.

4.3. Définition d'une ondelette :

La transformée continue en ondelette d'un signal *f(t)* est définit par l'équation (4.4) [63, 64] :

$$TOC_{a,b}(f) = \int_{-\infty}^{+\infty} f(t)\psi_{a,b}^{*}(t)dt \qquad (4.4)$$

Où $\psi_{a,b}^{*}(t)$est la fonction conjuguée de $\psi_{a,b}$. Les coefficients $TOC_{a,b}(f)$ sont nommés les coefficients d'ondelettes de la fonction *f*. out signal c'écrit alors sous forme d'une superposition des ondelettes translatées et dilatées :

$$f(t) = C_{\psi} \int_{-\infty}^{+\infty} \left[\int_{\infty}^{\infty} TOC_{a,b}(f)\psi_{a,b}(t) \frac{da}{a^2} \right] db \qquad (4.5)$$

Où C_{ψ} est un nombre constant qui ne dépend que de l'ondelette choisie ψ qui est définit par l'équation :

$$C_{\psi} = \int_{-\infty}^{+\infty} \frac{\left|\hat{\psi}(\omega)\right|^2}{|\omega|} d\omega < \infty \qquad (4.6)$$

La transformée continue d'ondelette est inversible si l'ondelette mère ψ est vérifie les conditions d'admissibilité [62]

4.4. La transformée discrète en ondelettes (*DWT*) :

Les fondations de la DWT se progresse en 1976 où des techniques de décomposition des signaux discrets ont été conçues [64]. Un travail semblable a été effectué dans le codage du signale de la parole qui a été appelé codage de sous-bande. En 1983, on a développé une technique semblable au codage de sous-bande appelée codage pyramidal. Plus tard beaucoup d'améliorations ont été apportées à ces codes qui ont eu comme conséquence des arrangements efficaces d'analyse multi-résolution (MRA).

La transformée discrète en ondelette consiste en la projection du signal sur une base discrète d'ondelettes réalisant un échantillonnage critique du plan temps-échelle (*t,a*). C'est-à-dire, tel qu'un échantillonnage plus serré conduirait à une redondance d'information, alors qu'un échantillonnage plus vague ne permettrait plus la reconstruction du signal. Une première approche consiste à rechercher, à partir de l'ensemble continu des fonctions de base utilisées dans la *CWT* (transformée redondante dans la mesure où deux coefficients voisins partagent de l'information), une famille discrètes possédant ces propriétés. Donc il est toutefois possible de réduire cette redondance en remplaçant la famille continue d'ondelettes par une famille indexée par des variables temps et d'échelle discrètes, et les intégrales par des sommes discrètes. Il est préférable de réduire au maximum cette redondance en fixant $= 2^{-j}\ et\ b = k\ 2^{-j}$, la famille d'ondelettes correspondantes devient :

$$\psi_{j,k}(t) = 2^{j/2}\psi\left(2^j t - k\right) \qquad (4.6)$$

Cette transformée est appelée transformée dyadique [42.48,45]. En choisissant adéquatementψ, la famille $\psi_{j,k}$ constitue une base orthonormé, on pourra dès lors récupérer le signal originale par la transformée inverse qui s'écrit alors :

$$f(t) = \sum_j \sum_k c_{j,k}\, \psi_{j,k} \qquad (4.7)$$

Avec

$$c_{j,k=} \int_{-\infty}^{+\infty} f(t)\, \psi_{j,k}(t) dt \qquad (4.8)$$

Qui représente les coefficients d'ondelettes qui fournissent donc une représentation alternative de ces fonctions, sans perte d'information ni redondance.

4.5. L'analyse multirésolution (MRA) :

L'analyse multirésolution [59,55,56,65] (MRA) par les ondelettes est devenue fondamentale en théorie du signal. Une analyse multirésolution est définit des opérateur linéaires permettant d'analyser un signal à différents échelles. Mise au point vers la fin de l'année 1986 par Mayer [57] et Mallat [65], elle constitue un outil permettant de regarder un signal de « très près » ou de « très loin ». ce zoom est effectue à l'aide d'une fonction d'échelle ϕ, qui se dilate à travers les échelles. Ce qui consiste à projeter le signal x dans une série de sous-espaces d'approximations *Vj* et de sous-espaces de détails *Wi* . Les projections successives du signal sur les espaces *Vj* en constituent des approximations de plus en plus grossières. L'information perdue entre deux approximations est collectée dans les signaux de détails, qui rendent alors compte du comportement du signal à des résolutions différentes.

En fait, une analyse multi-résolution de L2($\Re$) est une suite $\{Vj\}j \in Z$ de sous-espaces fermés de L2($\Re$) ayant les propriétés suivantes :

1) $\forall (j,k) \in Z^2, f(t) \in Vj \quad \Leftrightarrow f(t-2^j k) \in Vj$
2) $\forall j \in Z \quad , V_{j+1} \subset V_j$
3) $\forall j \in Z \quad , f(t) \in V_j \Leftrightarrow f(\frac{t}{2}) \in V_{j+1}$
4) $\cup_{j=\infty}^{\infty} V_j = L^2(\Re)$
5) $\cap_{j=-\infty}^{\infty} V_j = \{0\}$

Il existe $\phi = V_0$telle que $\{\phi(t-n)\}_{n\in Z}$soit base orthonormée de V0.

$L^2(\Re)$ Étant l'espace vectoriel des signaux *f(t)* continus à une énergie finie où *t* appartiens à l'ensemble des réels $\Re$. Z correspond à l'ensemble des entiers.

Il s'agit d'effectuer des projections successives *f(t)* à étudier sur des espaces V_j correspondant à une approximation d'autant plus grossière que *j* est grand comme le montre l'équation suivante :

$$approx_j = proj_{Vj}\{f(t)\} = \sum_k a_{j,k}\, \phi_{j,k}(t) \qquad (4.9)$$

D'une approximation à l'autre, notamment de V_{j-1} à V_j une partie de l'information est perdue mais récupérée par les détails :

$$detail_j = proj_{j-1} - proj_j \qquad (4.10)$$

Les signaux de détails s'obtiennent également par projection du signal f(t) sue sous-espaces tels présentés par l'équation (4.10) :

$$\begin{cases} V_j + W_j = V_{j-1} \\ detail_j(t) = proj_{wj}\{f(t)\} \end{cases} \qquad (4.11)$$

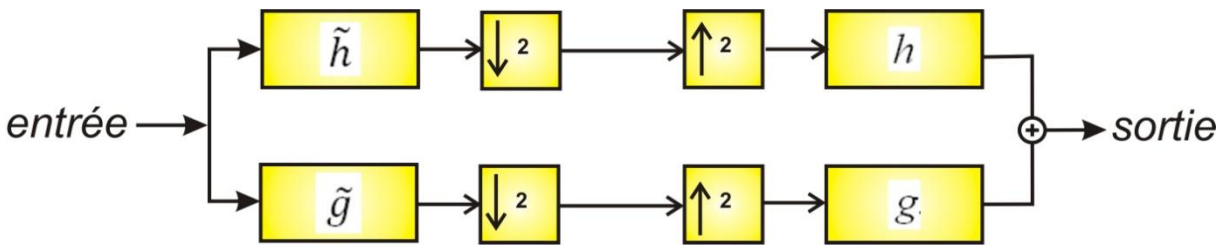
L'analyse multirésolution reformule ainsi un signal *f(t)* sous la forme d'une somme de détails et d'une analyse approximation :

$$f(t) = approx_j(t) + \sum_{j+1}^{j} detail_j(t)$$

$$= \sum_k a(J,k)\phi_{j,k}(t) + \sum_{j=1}^{j}\sum_k d(J,k)\psi_{j,k}(t) \qquad (4.12)$$

4.6. L'algorithme pyramide de Mallat :

L'algorithme pyramidal [55, 53, 65] est une méthode récursive basée sur une succession de convolutions. En effet, Mallat a montré que les coefficients d'ondelettes peuvent être calculés à partir d'une transformées pyramidale mise en ouvre à l'aide de filtres numériques. Récursifs ou non. Le principe de la transformée pyramidale consiste dans la décomposition en banc de filtre (figure) du signal à analyser à l'aide d'une paire de filtre miroirs en quadratiques. L'un de ces filtres fournira les coefficients d'ondelettes (ou détails), le second les coefficients d'approximation. L'approximation est elle-même à son tour décomposée par une seconde paire de filtres, la reconstitution s'obtient simplement par inversion des filtres dans le cas de base orthogonales (figure 4.4)



Décomposition en banc de filtre d'une analyse multirésolution

Dans la figure $\tilde{h}$ est le filtre conjugués en quadrature de h et $\tilde{g}$ est le filtre conjugués en quadratique de g.

Mallat [65] a déduit les formules de décomposition et de reconstitution suivantes : Formules de décomposition (4.13) :

$$a_n^j = \sum_l \tilde{h}[2n-l]a_l^{j-1}$$

$$d_n^j = \sum_l \tilde{h}[2n-l]a_l^{j-1} \qquad (4.13)$$

Formule de reconstitution (4.14) :

$$\tilde{h}[n] = h[-n]$$

$$\tilde{g}[n] = g[-n] \qquad (4.14)$$

De même, les filtres h et g peuvent se déduire à partir des fonctions d'ondelettes et d'échelle par produit scalaire comme suit :

$$h[n] = \langle \phi, \phi_{-1,n} \rangle$$

$$g[n] = \langle \psi, \phi_{-1,n} \rangle \qquad (4.15)$$

$$evec\ \phi_{-1,n}(t) = 2^{\frac{-1}{2}}\phi(2x-n)$$

4.7. Propriétés des ondelettes :

Les propriétés les plus importantes des ondelettes [48] sont :

- Support compact
- Symétrie
- Nombre de moment nuls
- Régularité

4.7.1. Support compact :

La plupart des ondelettes sont à support compact dans le domaine temporel, ce qui veut dire qu'elles sont finie (filtres FIR) est se distinguent par leur atténuation rapide. Un support compact permet une réduction de la complexité de calcul, une meilleure résolution dans le domaine du temps mais donne une résolution pauvre en fréquence.

4.7.2. Symétrie :

Les ondelettes symétriques [65] donnent naissance à des filtres à phase linéaire. Daubchies [49,50] a montré que, pour qu'uen ondelette soit symétrie, le filtre *h* doit être à phase complexe linéaire, et que le seul filtre miroir conjugué symétrique à support fini est le filtre de Haar [66] qui correspond a une ondelette discontinue à un seul moment nul. Mis à part l'ondelette de Haar, il y'a donc pas d'ondelette réelle orthogonale symétrie à support compact.

4.7.3. Nombre de moment nuls :

Par définition, une fonction ψ a p moment nuls si et seulement si la formule (4.16) est vérifiée [65] :

$$\int_{-\infty}^{\infty} t^k \psi(t) dt = 0 \; pour \; 0 \leq k \leq p \qquad (4.16)$$

L'intérêt de cette propriété est que si une ondelette e un nombre de moments nuls suffisant, on obtiendra alors plus de coefficients à petites valeurs. Elle permet de caractériser aussi l'ordre des singularités d'un signal. Si nous désignons par *p* le nombre de moment nul, alors la taille du support est d'au moins de *2 p-1*.

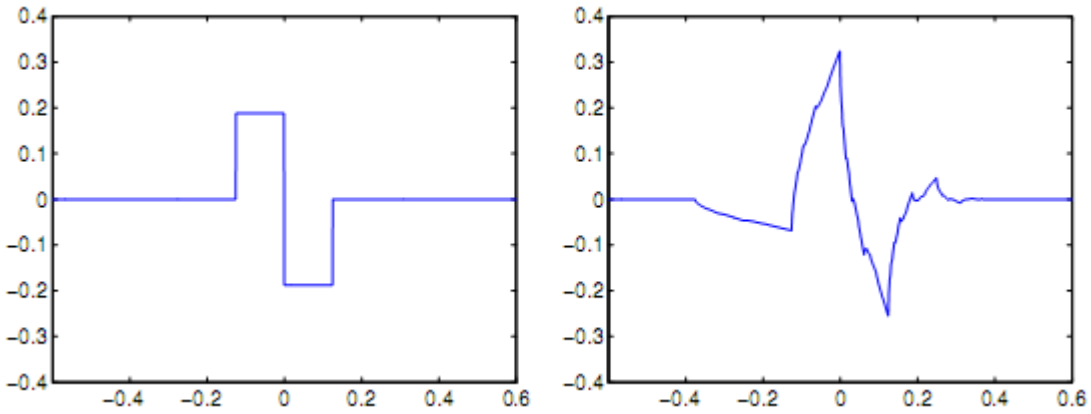
4.7.4. Régularité :

La régularité d'une ondelette [65] est la propriété permettant de localiser les singularités dans un signal. Cette propriété se traduit sur les coefficients d'ondelettes par une amplitude importante caractérisant une singularité dans le signal par la décroissance des coefficients avec l'échelle de résolution. La régularité est une propriété importante pour obtenir des coefficients d'ondelettes les plus petits possibles afin de les annuler.

4.8. Panorama d'ondelettes :

Il existe de nombreuses ondelettes dyadiques décrites dans la littérature [46] (Spline, hannon-Nyquist, Daubechies, etc...) utilisées en codage, débruitage ou analyse de signaux. Nous présentons ici quelques ondelettes couramment utilisées.

L'ondelette de Haar est une ondelette orthogonale symétrique elle la plus ancienne, De part sa simplicité, cette ondelette illustrée en Figure 4.5 est assez utilisée en codage d'image.



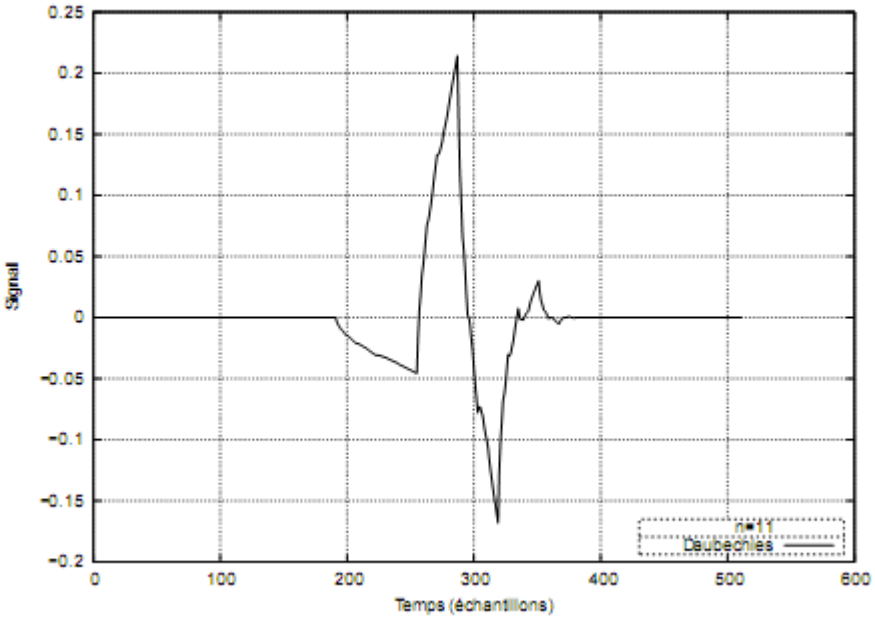
Type d'ondelettes

Par conséquent, n'importe quelle discussion des ondelettes commence par les ondelettes de Haar. Les ondelettes de Daubechies sont les ondelettes les plus populaires. De la famille des ondelettes orthogonales. Ils représentent les bases du traitement de signal et ils sont employés dans de nombreuses applications. Ceux-ci s'appellent également les ondelettes de Maxflat.

4.8.1. Base d'ondelettes de Daubechies :

Les ondelettes les plus connues et les plus utilisées dans le cadre du traitement du signal mono-dimensionnel discret sont ceux de Daubechies [69]. Pour leurs caractéristiques remarquables qui les distinguent des autres types d'ondelettes. Pour la transformé rapide en ondelettes (DWT), les fonctions sont définies par un jeu d'indices que l'on désigne sous l'appellation "coefficients des filtres en ondelettes" [70].

Les ondelettes de Daubechies à support compact sont décrites dans [71]. Ce sont des fonctions à p moments nuls, leur régularité augmente avec p. Le nombre de coefficients est de 4 pour p = 2, de 12 pour p = 6 et de 20 pour p = 10. La forme de chaque ondelette pour 4,12 et 20 coefficients est visualisée sur les figures 4.6.



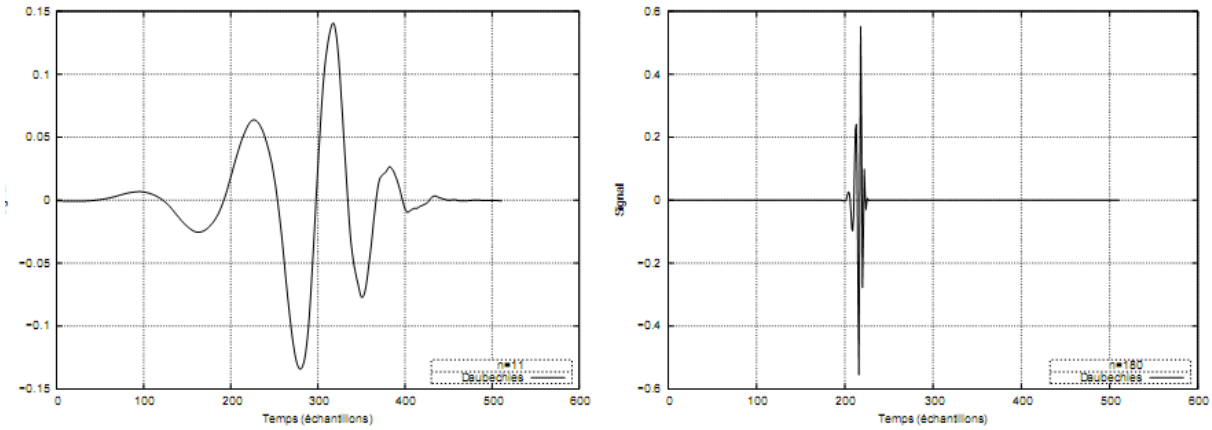
Ondelette de Daubechies à 2 moments nuls (4 coefficients).

Une transformée en ondelettes contient non seulement la répartition spectrale du signal analysé mais aussi une information temporelle. En effet, la transformée de Fourier (FFT) sur une fenêtre glissante en temps offre une résolution uniforme en temps et en fréquence, ce qui se traduit par un pavage régulier de l'espace temps-fréquence. La résolution temporelle est constante, c'est la largeur T de la fenêtre de calcul ($T = 2^N \delta t$, où 2^N désigne le nombre de points dans la fenêtre et δt le pas d'échantillonnage), la résolution fréquentielle est elle aussi constante, c'est l'inverse de la largeur de la fenêtre d'analyse

$$\sigma_f = \frac{1}{T} = \frac{1}{2^N \delta t} = \frac{f_e}{2^N} \qquad (4.35)$$

Nous pouvons exprimer cela en disant que l'on ne connaît que le spectre moyen du signal sur la fenêtre d'analyse. La résolution spectrale peut être rendue la plus faible possible en augmentant suffisamment la largeur de la fenêtre d'analyse, c'est à dire en augmentant autant que nécessaire le nombre d'échantillons dans la fenêtre : lorsque ce nombre augmente indéfiniment, le spectre tend à être continu. La valeur maximale de la fréquence du spectre ne dépend que de la fréquence d'échantillonnage, elle est égale à $\frac{f_e}{2}$ d'après le théorème d'échantillonnage ou de Shanon. Par contre, la transformée en ondelettes utilise un pavage très différent qui traduit le fait que le produit de la résolution temporelle par la résolution fréquentielle est constant pour tous les facteurs d'échelle (voir figure 4.7) [72].

Ceci se traduit dune part par une meilleure résolution en temps pour les hautes fréquences qui sont significatives de variations rapides que pour les basses fréquences, et d'autre part, une résolution temporelle inférieure pour les basses fréquences qui correspondent à des variations lentes.



Ondelette de Daubechies à 6 moments nuls (12 coefficients) pour 2 facteurs d'échelle différents.

Le coefficient d'ondelettes le plus élevé correspond à l'ondelette dont le spectre est le plus haut en fréquence, mais il ne correspond pas à une fréquence unique comme dans le cas de la FFT, dans Tableau 4.I, ils sont illustrés les différents coefficients (D2-D8).

3 LES DIFFERENTS COEFFICIENTS DU FILTRE DE DAUBECHIES

D2 P=1	D4 P=2	D6 P=3	D8 P=4
0.70710	**0.48296**	*0.33267*	0.23037
0.70710	**0.83651**	***0.80689***	***0.71484***
	0.22414	*0.45987*	*0.63088*
	-0.12940	***-0.13501***	***-0.02798***
		-0.85441	-0.18703
		0.03522	0.03084
			0.03288
			-0.01059

4.9. Applications

Le fait que la transformée utilise des fonctions bien localisées dans le plan temps-fréquence lui donne beaucoup d'avantages. La transformée en ondelettes est appliquée dans différents domaines s'étendant du traitement des signaux à la biométrie, et la liste se développe toujours. L'une des premières applications est dans la compression des empreintes digitale par FBI. La transformé en ondelettes est employés pour la compression des images d'empreintes afin de les stocker.

Dans DWT, l'information la plus en avant dans le signal apparaît dans des amplitudes élevées et l'information moins en avant apparaît dans des amplitudes très basses. La compression de données peut être réalisée en jetant ces basses amplitudes. La transformée en ondelette permet des taux de compression élevés avec une bonne qualité de reconstruction. Actuellement, l'application des ondelettes dans la compression d'image est un domaine de recherche les plus chauds. Récemment, La transformée en ondelettes été choisis pour la norme de compression de JPEG2000.

L'application de la transformée en ondelettes dans le traitement du signal.

La figure 4.8 représente les étapes générales suivies dans une application de traitement de signal. Le traitement (Processing) peut comporter la compression, le codage, débruitage etc…Le signal traité est stocké ou transmis. Pour la plupart des applications de compression, le traitement comporte la quantification et le codage d'entropie pour rapporter une image comprimée. Pendant ce processus, tous les coefficients d'ondelette qui sont au-dessous d'un seuil bien choisi sont jetés. Ces coefficients jetés sont remplacés avec des zéros pendant la reconstruction à l'autre extrémité. Pour reconstruire le signal, le codage d'entropie la transformée en ondelettes est décodé, puis quantized et puis finalement la transformé inverse d'ondelette.

Les transformés en ondelettes sont appliquées dans la plupart des technique de compression de la parole, qui réduit le temps de transmission dans des applications mobiles. Ils sont employés dans débruitage, détection de contours bord, extraction de dispositif, reconnaissance de la parole, annulation d'écho et etc... Ils sont très prometteurs pour des applications audio et vidéo en compression temps réel. Les ondelettes ont également de nombreuses applications dans les communications numériques. Parmi eux la Division de fréquence orthogonale multiplexé (OFDM). Des ondelettes sont employées dans la formation d'image biomédicale. Par exemple, les signaux d'ECG, mesurés à partir du coeur, sont analysés en utilisant la transformé en ondelette ou compressé pour qu'ils puissent être stockés. La popularité de la transformé en ondelette se développe en raison de sa capacité de réduire la déformation dans le signal reconstruit tout en maintenant tous les composantes significatifs présentent dans le signal.

4.10. Banc de filtre

4.10.1. Introduction :

La Multirésolution en traitement numérique du signal (DSP) a attiré beaucoup d'attention au cours des deux dernières années en raison des applications dans le codage sous-bande de la parole, audio et vidéo, support de transmission des données multiples, etc. Une caractéristique essentielle des algorithmes Multirésolution est leur haute efficacité de calcul. Un système multi-cadence peut augmenter ou diminuer la fréquence d'échantillonnage de signaux avant ou pendant leur traitement. Ces signaux peuvent être traités simultanément dans les différentes parties du système multi-cadence avec des différentes fréquences d'échantillonnage.

Les bancs de filtres Numériques sont les plus importantes applications des systèmes Multirésolution. La plus part des différentes approches des bancs de filtres ont été développés au cours des quinze dernières années. Parmi ces bancs de filtres, bancs de filtres modulés en cosinus [73,74,75] qui sont très utilisés parce qu'ils sont faciles à implémenté et peuvent fournir une reconstruction parfaite (PR). La transformée de Fourier discrète (DFT) de banc de filtre polyphase [76] est une autre approche qui fournit une grande efficacité de calcul. La DFT modifié (MDFT) du banc de filtre [77,78,79] peut également fournir une reconstruction parfaite.

4.10.2. Conversion de la fréquence d'échantillonnage

Pour comprendre les systèmes Multirésolution, il est essentiel de comprendre comment la fréquence d'échantillonnage on changer. Il ya deux opérations de base de changement des taux d'échantillonnage: sous-échantillonnage (diminuer la fréquence d'échantillonnage) et sur-échantillonnage (augmenter la fréquence d'échantillonnage).

4.10.2.1. Sous-échantillonnage :

Le sous-échantillonnage est le processus de réduction de la fréquence d'échantillonnage. Sous-échantillonnage d'un signal peut être utile si le taux d'échantillonnage est supérieur à la largeur de bande du signal. Il peut réduire le calcul et / ou ainsi que la mémoire nécessaire pour l'implémentation d'un système de traitement numérique du signal. L'opération de sous-échantillonnage par un facteur M est illustré dans la figure 4.9, où le système sous-échantillonnage prend un signal d'entrée x [n] avec une fréquence d'échantillonnage élevée F et il produit à la sortie une séquence d'échantillonnage faible y [m]

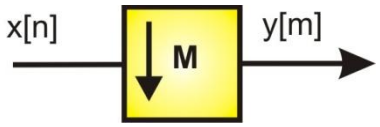
en gardant tous les échantillons M et en rejetant le reste. Ce processus de sous-échantillonnage peut être écrite comme suit :

$$y[m] = x[n]_{n=mM} \qquad \text{avec } n = 0,1,2 \dots N; m = 0,1,2, \dots \frac{N}{M} \qquad (4.36)$$

La relation entre la fréquence de sortie et la fréquence d'entré est :

$$F_L = F_H/M \qquad (4.37)$$

Où M est appelé le facteur de sous-échantillonnage, et il est simplement le rapport entre le débit d'entrée et le débit de sortie.



Le sous-échantillonnage

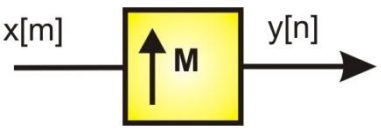
4.10.2.2. Sur-échantillonnage :

Le sur-échantillonnage augmente la fréquence d'échantillonnage du signal par l'insertion des échantillons de valeur zéro entre les échantillons originaux.

La Figure 4.10 illustre le sur-échantillonnage d'un signal par un facteur L, il a un signal d'entrée x [m] avec une faible fréquence d'échantillonnage FL et il produit à la sortie une séquence y [n] de fréquence d'échantillonnage elevée FH par l'insertion des zéros entre chaque échantillon du signal d'entrée. Ce processus de sur-échantillonnage peut être écrit comme suit:

$$y[n] = x[m]_{m=n/L} \qquad \text{avec } n = 0,1,2 \dots N; m = 0,1,2, \dots \frac{N}{L} \qquad (4.38)$$

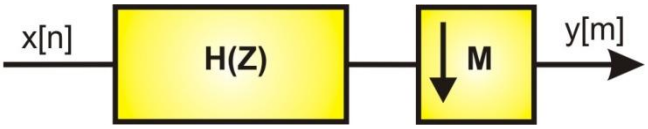
Dans ce cas le taux d'échantillonnage F_L et F_H sont reliés par un facteur L de sur-échantillonnage, qui est la relation du débit de sortie et le débit d'entrée.



Le sur-échantillonnage

4.10.3. La décimation :

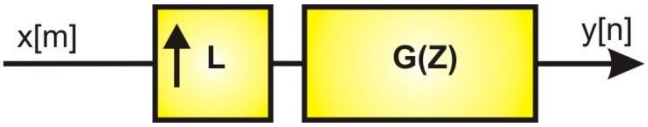
La décimation est le processus de filtrage et du sous-échantillonnage d'un signal pour diminuer sa fréquence d'échantillonnage. Un filtre passe-bas est utilisé avant le sous-échantillonnage est illustré sur figure 4.11.



La décimation.

4.10.4. L'interpolation :

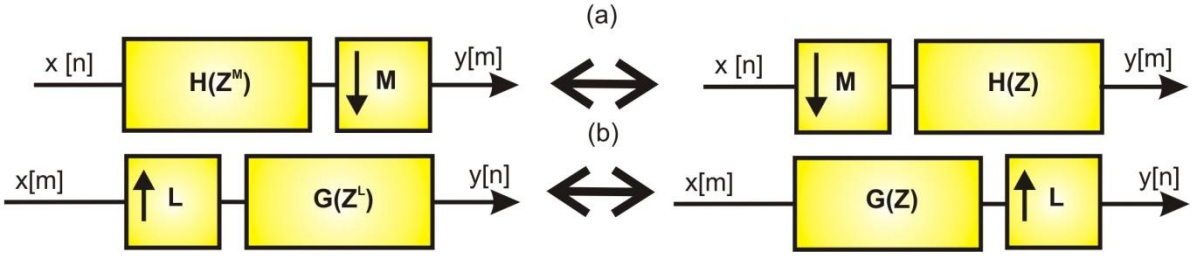
L'interpolation est le processus de sur-échantillonnage et de filtrage d'un signal afin d'augmenter son fréquence d'échantillonnage comme le montre la figure 4.12.



l'interpolation.

4.10.5. Les identités nobles :

Les identités nobles décrire la propriété de l'inversion de l'ordre du filtre et du sous-échantillonnage/sur-échantillonnage. Figure 4.13 (a) et (b) montrent une paire de schémas synoptiques équivalents, qui décrivent les identités nobles pour la décimation et l'interpolation.

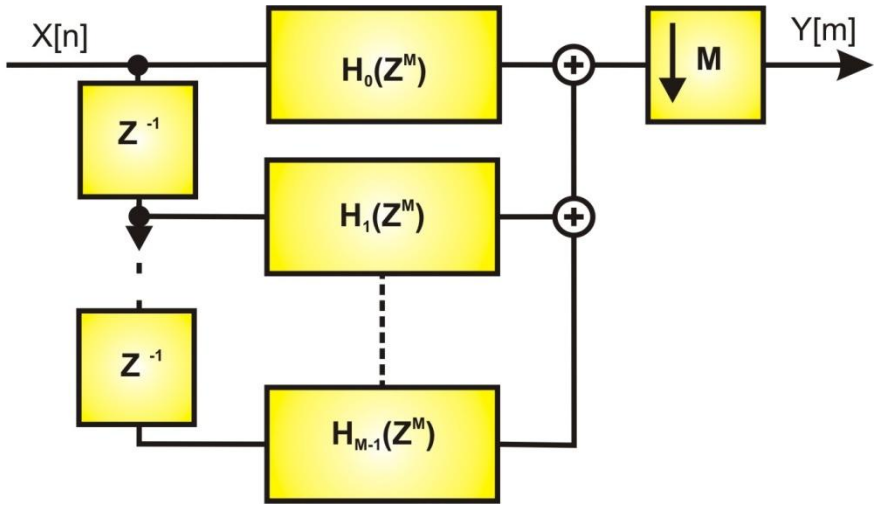


Les identités nobles

La méthode de décimation standard illustré dans la Figure 4.11, est inefficace arithmétiquement parce qu'elle jette la majorité des échantillons calculées de la sortie du filtre. En utilisant l'identité Noble nous pouvons réorganiser la structure de la figure 4.11. Afin d'appliquer l'identité Noble pour la décimation, nous devons d'abord décomposer le filtre H(z) en ses composantes polyphasées :

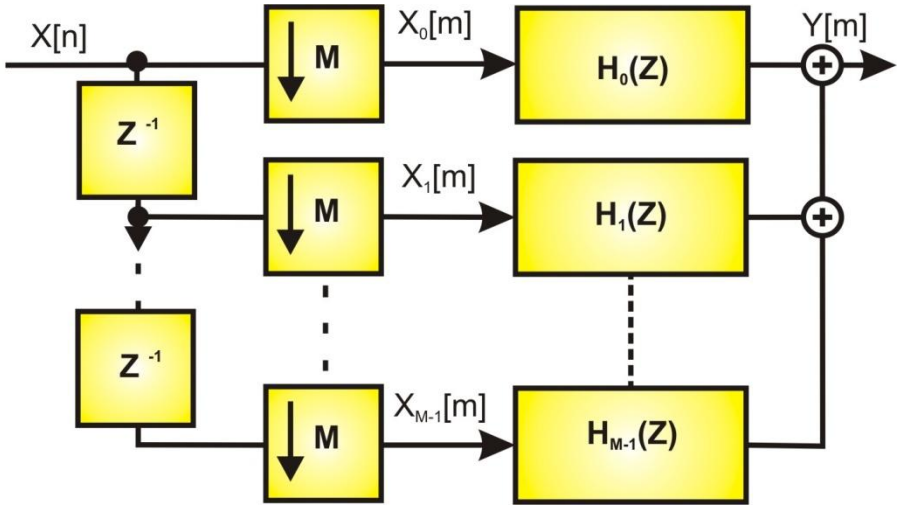
$$H(z) = \sum_{I=0}^{M-1} Z^{-1} H_i(Z^M) \qquad (4.39)$$

Maintenant nous pouvons transformer la figure à la structure polyphase de la figure



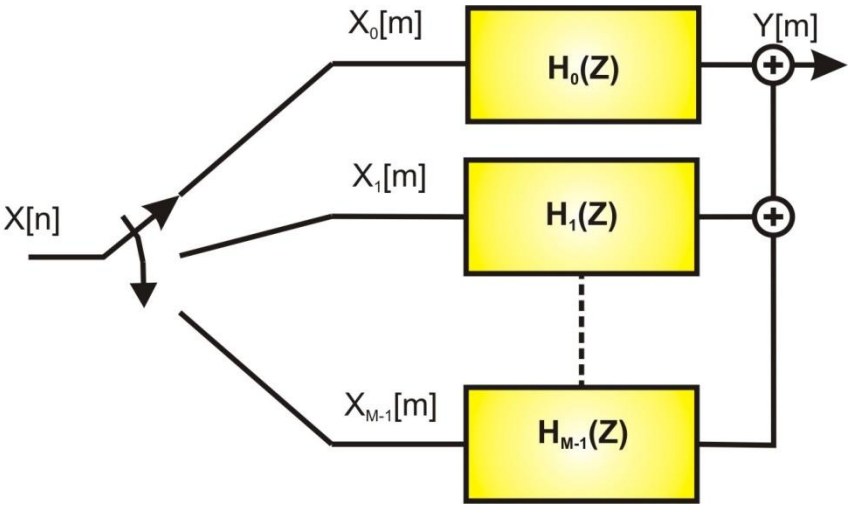
La structure polyphasé

Puis on peut appliquer l'identité noble pour la décimation sur la structure de la figure 4.14 il se résulte la structure efficace de la figure 4.15. Cette structure se semble à une sorte de conversion série-parallèle.



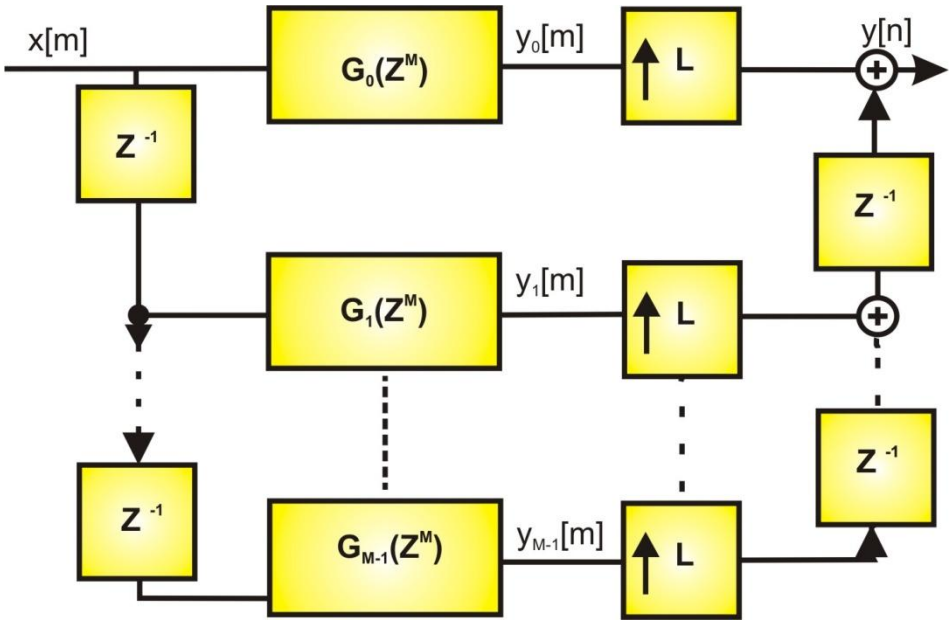
La structure polyphase efficace pour la décimation

La figure 4.16 représente une structure équivalente de la décimation polyphasée en utilisant un interrupteur à l'entrée pour représenter le fractionnement de signal d'entrée x0[m], x1[m],… xm-1[m], [76]



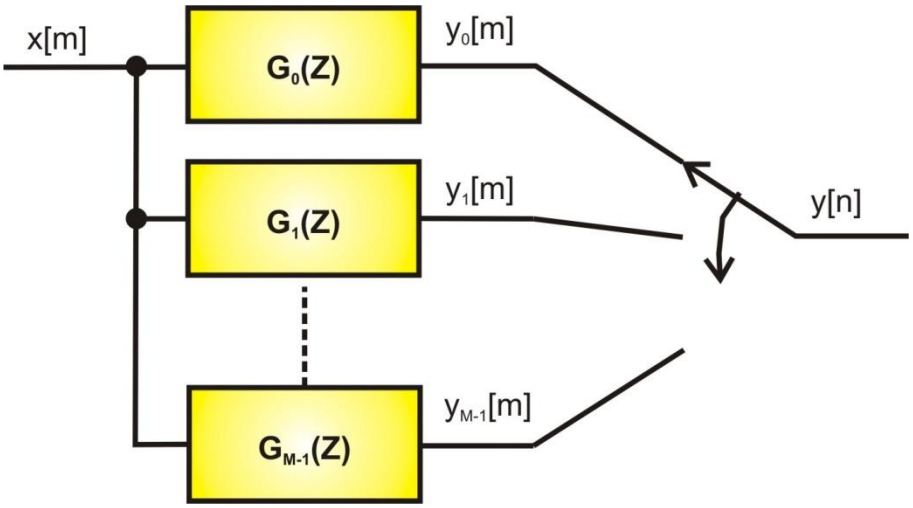
Le décimateur polyphase avec un interrupteur à l'entrée.

La procédure d'interpolation standard illustrée dans la figure 4.12 est également inefficace arithmétiquement car le filtre fonctionne sur une séquence qui est surtout composée de zéros. Nous pouvons utiliser les mêmes procédures que la décimation polyphase pour transformer la figure 4.12 en une structure plus efficace comme le montre la Figure 4.17 en utilisant l'identité Noble pour l'interpolation.



La structure polyphase efficace pour l'interpolation

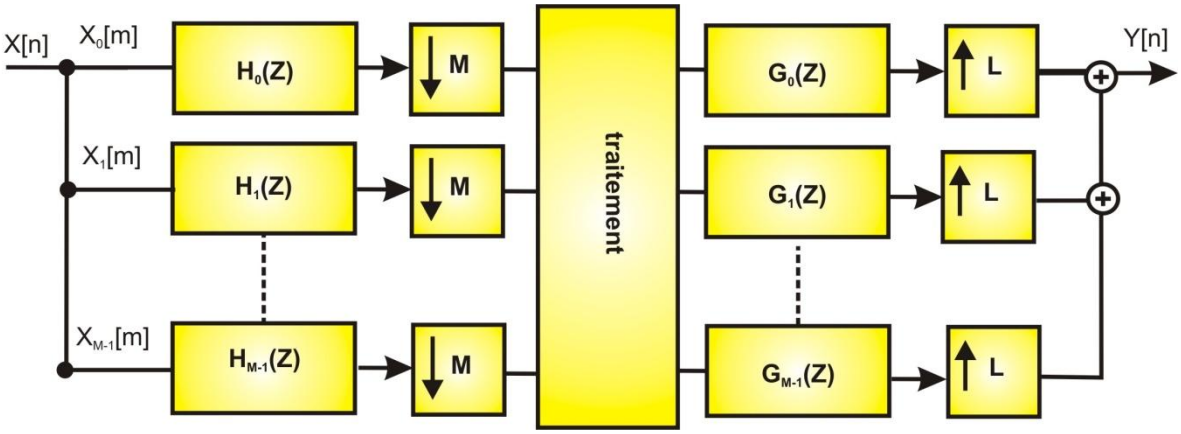
Ce processus peut être également représenté par un interrupteur qui combine séquence dans le signal de sortie comme illustré à la figure 4.18. L'interrupteur combine le signal de sortie par le prélèvement d'un échantillon par échantillon de sous-séquence.



L'interpolateur polyphase avec un interrupteur à la sortie.

4.10.6. Utilisation du Bancs de filtre :

Pour de nombreuses applications en traitement du signal, il est utile de séparer un signal dans différentes bandes de fréquences appelées sous-bandes. Le spectre peut être partitionné en 2. Les sous-bandes peuvent également être non uniformes. Pour notre discussion, nous allons concentrer uniquement sur les espacées de sous-bandes uniforme. Le but de la séparation en sous-bande est de faire un traitement ultérieur plus commode. Certaines de ces applications les plus concorde pour la décomposition en sous-bande sont le codage des signaux audio et vidéo dans le but de les stocker efficacement et / ou de les transmettre.



La structure d'un banc de filtre standard

4.11 Débruitage de signaux

4.11.1 Introduction :

Dans les conditions idéales, le niveau de bruit est réduit à un niveau négligeable par rapport au niveau du signal pour que tout débruitage soit inutile. Malheureusement, dans les

données réelles le bruit est toujours présent. Des techniques de séparation de sources permettent d'estimer les signaux orignaux issus de mélanges.

Dans la présente partie, nous allons donc tout d'abord présenter et clarifier la notion de débruitage, puis nous présenterons quelques techniques utilisées pour le débruitage de signaux.

4.11.2POSITION DU PROBLEME

Le débruitage de signaux est une notion relativement vague si l'on ne précise pas ce que cela signifie pour nous. Dans ce paragraphe nous allons donner une interprétation plus formelle de ce qu'est le débruitage de signaux.

Supposons que l'on ait des observations bruitées *yi* d'un signal x_i

$$y_i = x_i + n_i \qquad i = 0,1,\ldots\ldots,N-1 \qquad (4.40)$$

où n_i est un bruit blanc gaussien centré indépendant et identiquement distribué (iid) de variance $\sigma^2 : n_i \sim \mathcal{N}(0,\sigma^2)$. Notre interprétation de la notion de débruitage est d'inférer sur la valeur des échantillons x_i à partir des observations corrompues *yi* en utilisant divers critères comme la minimisation de l'erreur quadratique moyenne définie par

$$EQM = E\left\{\left\|\hat{X} - X\right\|^2\right\} = \sum_{i=0}^{N-1} E\{(\hat{x_i} - x_i)^2\} \qquad (4.41)$$

4.11.3DEBRUITAGE PAR FILTRES LINEAIRES

Une première méthode pour effectuer un débruitage est d'utiliser des filtres spectraux: ils traitent les coefficients issus de la transformée de Fourrier en fonction de leur fréquence. Leur principale idée est d'ajuster la bande effective au signal recherché de façon à réduire la puissance du bruit d'où une augmentation du rapport signal sur bruit défini par $RSB = P_{signal}/P_{bruit}$ où P_{signal} est la puissance du signal et P_{bruit} celle du bruit. L'emploi du filtrage de Wiener permet de minimiser l'erreur quadratique moyenne, les coefficients du filtre sont alors donnés par

$$H(F) = \frac{\Gamma_s(f)}{\sigma^2 + \Gamma_s(f)} \qquad (4.42)$$

Où $\Gamma_s(f)$ est la densité spectrale de puissance du signal, *σ2* la variance du bruit blanc qui perturbe les données *H(f)* la fonction de transfert du filtre *h(n)*. Cependant, l'emploi d'un tel filtre suppose que l'on ait une connaissance statistique du second ordre sur le signal.

De plus, le débruitage par filtre, qui repose sur l'idée de traiter les échantillons en fonction de leur fréquence, sera d'autant plus performant que le signal et le bruit occupent des bandes de fréquences distinctes.

4.11.4 DEBRUITAGE PAR ONDELETTES

L'un des plus grands succès des ondelettes est le débruitage [80]. En effet, cette technique repose essentiellement sur des algorithmes simples et performants et s'est avérée souvent beaucoup plus efficaces que les techniques traditionnelles souvent plus lourdes et moins efficaces. Nous allons dons présenter les deux grandes catégories de méthodes pour le débruitage par ondelettes : le débruitage par seuillage et le débruitage par projection dans une base convenable. Pour cela nous allons exploiter deux propriétés intéressantes de la transformée en ondelettes discrète:

- Elle est éparse, c'est-à-dire que seul un petit nombre de coefficients a une amplitude importante,
- Les coefficients sont moins corrélés que les échantillons du signal.

Ces propriétés, jointes à l'utilisation d'algorithmes rapides de calcul de la transformée en ondelettes discrète, font que l'on traite les coefficients de la transformée plutôt que les échantillons du signal.

4.11.5DEBRUITAGE PAR SEUILLAGE

La propriété de dé-corrélation nous suggère de traiter les coefficients indépendamment les uns des autres et le fait que la transformée en ondelettes soit éparse nous incite à traiter les coefficients en fonction de leur amplitude et donc à utiliser des estimateurs à seuils.

Notons *W(·)* et *W-1 (·)* les opérateurs direct et inverse de la transformée en ondelettes discrète. Soit *D(·,λ)* l'opérateur de débruitage associé au seuil λ. En regroupant les observations *yi* du signal observé dans un vecteur

$$y = s + n \qquad (4.43)$$

où *s* représente le signal original informatif, et *n* est un bruit blanc gaussien centré indépendant et identiquement distribué de matrice de covariance $\sigma^2\Gamma$: $n \sim N(0, \sigma^2\Gamma)$, l'algorithme de base de la procédure de débruitage peut être traduit alors en trois étapes essentielles [1] :

- La décomposition par la transformée en ondelettes :
  - $w = W(y)$ (4.44)
- le seuillage des coefficients issus par lé décomposition :
  - $T = D(w,\lambda)$ (4.45)
- la reconstruction par la transformée en ondelette inverse :
  - $\hat{s} = W^{-1}(z)$ (4.47)

En effet, à partir du signal à débruiter, on décompose le signal sur une base orthogonale d'ondelettes. On effet du signal ensuite une opération de seuillage qui consiste à éliminer les coefficients qu'on considère come du bruit ou à les réduire en fonction du seuil calculé. En dernier lieu, on applique la transformée en ondelettes inverse sur les coefficients seuillés et on récupère le signal débruité.

Formulons le problème par le modèle mathématique. Soit y un signal corrompu par un bruit e. on peut ainsi écrire :

$$y(i) = x(i) + e(i) \quad (i = 0,1, \dots, N-1) \qquad (4.49)$$

Où N est la taille du signal y.

La transformée en ondelettes étant une fonction linéaire, on l'applique sur l'équation 11 et on obtient :

$$W_y = W_x + W_e \qquad (4.50)$$

W étant la transformée en ondelettes. Soit t(.) la fonction de seuillage par ondelettes, alors le schéma de débruitage par ondelettes peut être exprimé selon l'équation :

$$\hat{x} = W^{-1}\left(T\left(W_y\right)\right) \qquad (4.51)$$

Où $T\left(W_y\right)$ est le vecteur des coefficients de la TOD seuillés et $\hat{x}$ le signal débruité. ceci s'interprète simplement par l'application de TOD sur le signal bruité y, on obtient ainsi le vecteur des coefficients de la TOD $\left(W_y\right)$ sur lequel on applique la fonction de seuillage$\left(T\left(W_y\right)\right)$. Le signal débruité $\hat{x}$ quand à lui, est obtenu en appliquant la transformée en ondelettes inverse sur le vecteur des coefficients seuillés$W^{-1}\left(T\left(W_y\right)\right)$.

4.11.6 METHODE DE SEUILLAGES :

Les méthodes de seuillage les plus connues, introduites par Donoho [81] sont le seuillage doux et le seuillage dur. Des variantes de ces types de seuillage [82] ont été développées pour tenter d'améliorer ces méthodes.

4.11.6.1.SEUILLAGE DUR (HARD)

Comme nous l'avons souligné, la transformée en ondelettes discrète est éparse: le signal *s* est caractérisé par un faible nombre de coefficients θ de grande amplitude. Le seuillage dur exploite cette propriété en annulant tous les coefficients W_i inférieur au seuil [83]:

$$D_{dur}(w,\lambda) = \begin{cases} 0 \; si \; |w| \leq \lambda \\ w \; sinon \end{cases} \qquad (4.52)$$

si *w* représente un coefficient quelconque de la transformée en ondelettes discrète (figure 4.21). Ceci revient à considérer que les coefficients de faible amplitude (ie inférieure à λ) ne sont pas dus au signal mais uniquement au bruit, ils sont donc considérés comme nuisibles ce qui nous suggère de ne pas en tenir compte en les annulant.

Cependant, un tel estimateur peut entraîner, du fait de sa discontinuité, des oscillations comparables à celles engendrées par le phénomène de Gibbs (conduisant à des variances généralement élevées).

4.11.6.2.SEUILLAGE DOUX (soft) :

Pour palier l'inconvénient du seuillage dur, Donoho suggère de choisir un estimateur doux [66] défini par

$$D_{doux}(w,\lambda) = sgn(w)(|w| - \lambda)_+ \qquad (4.53)$$

où *sgn(·)* est la fonction signe et *(·)+* représente la partie positive de l'expression entre parenthèses. Cet estimateur est continu: il annule tous les coefficients dont l'amplitude |w| n'excède pas λ et diminue les autres de la valeur du seuil (figure 4.21), mais il introduit un biais systématique puisqu'il contracte les coefficients quelle que soit leur amplitude.

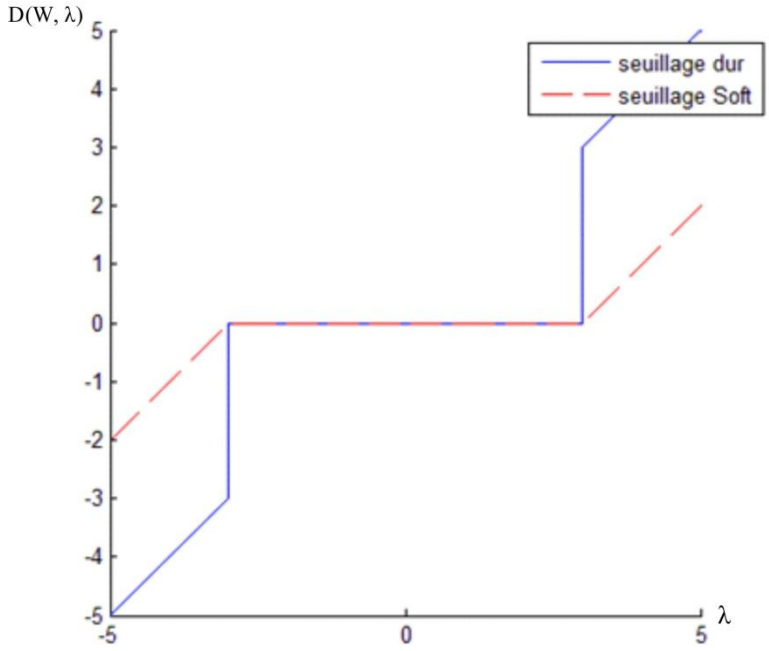
4.11.6.3.SEUILLAGE DUR MODIFIE :

Pour corriger le problème de la discontinuité fréquentielle, Kwon a proposé une version modifiée du seuillage dur [67]. La formule 4.54 représente la fonction du seuillage de Kwon.

$$D_{dur-mod}(w,\lambda) = \begin{cases} x \; si \; |w| > \lambda \\ \frac{1}{\mu} \; sign(w).\lambda.\left((1+\mu)^{\frac{|w|}{\lambda}} - 1\right) \quad |w| < \lambda \end{cases} \quad (4.54)$$

Où λ est la valeur du seuil et μ est une constante. *D(W, λ)*

Cette technique de seuillage procure des résultats très satisfaisant du fait qu'elle est très performante quand à l'élimination d'une partie du bruit et qu'elle altère moins le signal par rapport aux deux autres méthodes de seuillage.



Le seuillage dur et le seuillage doux.

La figure représente le seuillage dur (courbe entrait plein) : les coefficients dont l'amplitude est inférieure au seuil λ, choisi ici égal à 3, sont annulés; les autres restent inchangés. La figure représente aussi quant à elle le seuillage doux (courbe entrait plein): les coefficients dont l'amplitude est inférieure au seuil λ sont annulés, les autres voient leur amplitude diminuée de la valeur du seuil.

4.11.6.4.SELECTION DU SEUILLE :

Le calcul du seuillage est une tâche très importante dans la mesure où on estime le niveau du bruit en fonction de sa nature et la nature du signal. Avec l'hypothèse que le bruit est un bruit blanc gaussien de variance σ^2, Donoho et Johnstone ont alors montré [84] que le risque induit par un seuillage (dur ou doux) sur les coefficients d'ondelettes pouvait être encadré par des valeurs proches de la borne inférieure obtenue avec les estimateurs d'oracle.

Un estimateur d'oracle est un estimateur construit en connaissant le signal recherché. Le risque est définit par l'équation :

$$R(x,\hat{x}) = E(\|x-\hat{x}\|^2) = E\left(\sum_{i=0}^{N-1}\left(x-\hat{x}(i)\right)^2\right) \qquad (4.55)$$

Où N est la taille du signal x.

Le théorème de Donoho et Johnstone définit un seuil T pour les coefficients de la transformée en ondelettes [84] et dont la valeur est :

$$\lambda = \sigma\sqrt{2\ln(N)} \qquad (4.56)$$

Avec un risque $r_d(x)$ qui est encadré par deux valeurs comme le montre l'expression :

$$r_0(x) \leq r_d(x) \leq (2log(N)+1)\left(\sigma^2 + r_0(x)\right) \qquad (4.57)$$

Où $r_0(x)$ est définit comme suit :

$$r_0(x) = \sum_{i=0}^{N-1} min\,(d_i^2 + \sigma^2) \qquad (4.58)$$

Où les $d_i$représente les coefficients de la transformée en ondelettes de *x*.

Lorsqu'on utilise le seuille de Donoho et Johnstone, il est nécessaire d'avoir une estimation de variance σ^2 du bruit. Pour l'estimer à partir des données $r_i(x)$; il faut supprimer l'influence de $r(i)$. L'estimateur utilisé est celui de la médiane des coefficients d'ondelettes proposé par Donoho.

Chapitre V

Débruitage du signal

Principe & Implémentation

5. DEBRUITAGE DU SIGNAL : PRINCIPE ET IMPLEMENTATION

5.1 Introduction

Les cibles reconfigurables « FPGAs » sont de technologies dédiées pour développer les systèmes de traitements numériques qui requièrent des opérations en temps réel. Les circuits intégrés ayant des applications spécifiques (ASIC) sont également un autre substitut pouvant être utilisé dans ce contexte-là mais les FPGAs, à cause de leurs performances en terme de cout et d'optimisation, fournissent plus de flexibilité : ils peuvent être utilisés dans plusieurs disciplines en l'occurrence [85, 86, 78,88]. Ils offrent la possibilité de mettre à jour rapidement et/ou de modifier les architectures pour répondre aux besoins évolutifs imposés par le cahier de charge assigné. Ainsi les FPGAs contiennent des cellules logiques fixes ce qui laisse la possibilité au concepteur de construire des fonctions complexes à partir de ces cellules ou éléments logiques [89,90].

5.2 Méthodologie de développement

La méthodologie de développement d'une amélioration de l'implémentation d'une structure de débruitage d'un signal en utilisant la transformée en ondelettes discrètes à une dimension (1D-DWT) exige plusieurs différentes parties et l'utilisation de différents types de paquetage à cause des problèmes liés à la nature du VHDL qui peuvent être rencontrés. Ces problèmes sont. En effet, contrairement à des langages tels que le langage C, tout ce qui est accepté à la compilation et à la simulation ne sera pas forcément accepté à la synthèse. Cela implique de choisir dés le début de la rédaction du code des architectures simples qui pourront être synthétisées. Les avantages et les inconvénients de ces parties sont expliqués ainsi que les mesures qui ont été prises pour leurs implémentations. Les parties sont comme suit [91]:

- ADC/DAC codec Audio.
- Module d'horloge.
- Transformée en ondelettes discrètes.
- Le seuillage.
- Transformée en ondelettes inverse.

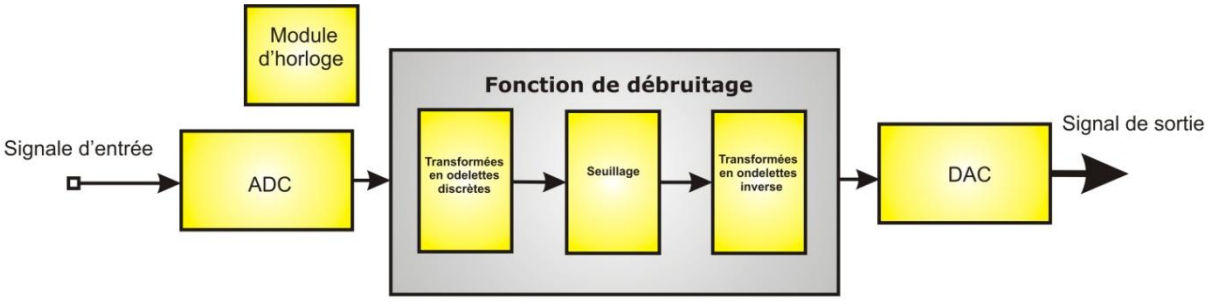


Figure 5.1. Bloc diagramme des différentes parties

Le schéma ci-dessus montre globalement les différents composants décrivant le système à réaliser. Chaque partie sera bien décortiquée dans les sections qui suivent. En effet, l'architecture sera également conçue et implémentée sur une cible matérielle de type FPGA. Ainsi l'implémentation en la matière, sera présentée et discutée ultérieurement.

5.2.1 ADC/DAC CODEC AUDIO

Comme il a été évoqué précédemment, le CODEC Audio (WM8731) se programme grâce à un ensemble de registres via une liaison série I^2C (Inter-Integrated Circuit). Il a été développé au début des années 1980, par Philips pour minimiser les liaisons entre les circuits intégrés numériques de ses produits (Téléviseurs, éléments HiFi, magnétoscopes, ...). Le bus I^2C permet de faire communiquer entre eux des composants électroniques très divers grâce à trois fils seulement: un signal de données (I2C_SDAT), un signal d'horloge (I2C_SCLK), et un signal de référence électrique (masse).

Le contenu de la ligne de données est envoyé dans le même ordre comme il est montré sur la figure 5.4 après une condition de départ : «RADDR ',' R / W», «ACK», DATAB [15-19], et «DATAB [8-0] qui signifient respectivement "adresse de base", "Read / Write", " acknowledge ", "l'adresse de contrôle", et "les données de contrôle". Le bloc modifie les paramètres. Par exemple, si «DATAB [0] 'est à '1', le volume est augmenté. Les adresses de base et de contrôle sont utilisées pour spécifier les registres internes du CODEC qui doivent être accessibles.

La condition de départ START est un front descendant sur la ligne I2C_SDAT tant que la ligne I2C_SCLK est au niveau HAUT. La condition de STOP après transfert de données est un front montant sur la ligne I2C_SDAT avec I2C_SCLK est HIGH. Enfin, le signal ACK est transmis du codec à l'FPGA, l''ACK' signal est envoyé, dans le sens opposé de tous les autres bits de données de la ligne. Ceci introduit la nécessité d'implémenter la ligne 'I2C_SDAT' en broche bidirectionnelle, cela nécessite l'utilisation d'un tampon à trois états, une machine d'état (FSM) a été créé pour mettre l'implémentation de l'interface de bus entre le FPGA et le codec audio. Tout en notant que le signal I2C_SCLK doit opérer entre 0Hz et 400kHz,

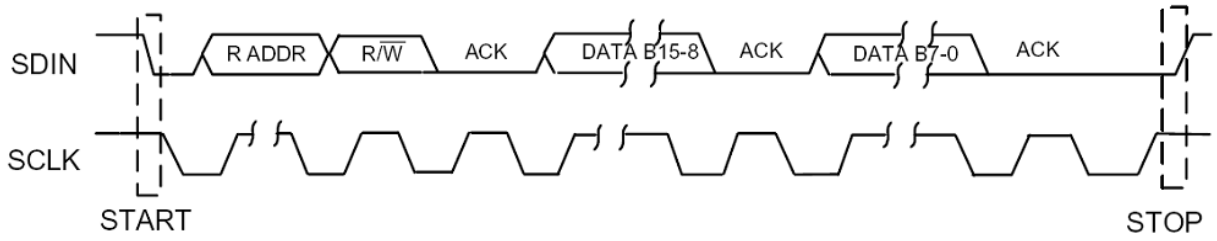


Figure 5.2. Les protocoles des lignes de bus I^{2}C de WM8731 Wolfson

5.2.2 Module d'horloge

Le signal d'horloge est généré par la carte à une fréquence de 50Mhz [46]. Il agit en tant que signal maître du codec audio, ce dernier doit être alimenté avec différents signaux horloges: L'horloge principale audio (XCK), ('ADCLRC'), et ('BCLK') doivent être générer selon les caractéristiques du Wolfson données par le constructeur, les deux signaux (ADCLRC) et (XCK) dépendent de la fréquence d'échantillonnage. Comme cette dernière est de 48Khz le signal d'horloge (ADCLRC) et (BLCK) doivent également être de 48 KHz comme il est montré sur la figure 4.5. Le signal (XCK) est de 12,288 MHz. [64]. (BCLK) doit être d'au moins 2,4 MHz.

Pour produire les trois horloges, un seul module d'horloge a été conçu. Il admet l'horloge 50MHz comme une horloge entrée. L'utilisation d'un compteur où le deuxième bit divise la fréquence d'entrée par 2^2 produisant le signal (XCK) à une fréquence de 12.5MHz (comme on le voit sur la figure 5.3). De même, (ADCLRC) et (BCLK) sont générés en utilisant respectivement le dixième bit et le troisième bit du compteur (divise par 2^{10} et 2^3). Ce qui produit des signaux à des fréquences de 48,83KHz et 6,25 MHz (comme le montre la figure 5.3). Ces valeurs sont des approximations à celles idéales et spécifiques dans les fiches techniques de données, elles sont assez proches pour des raisons pratiques [92].

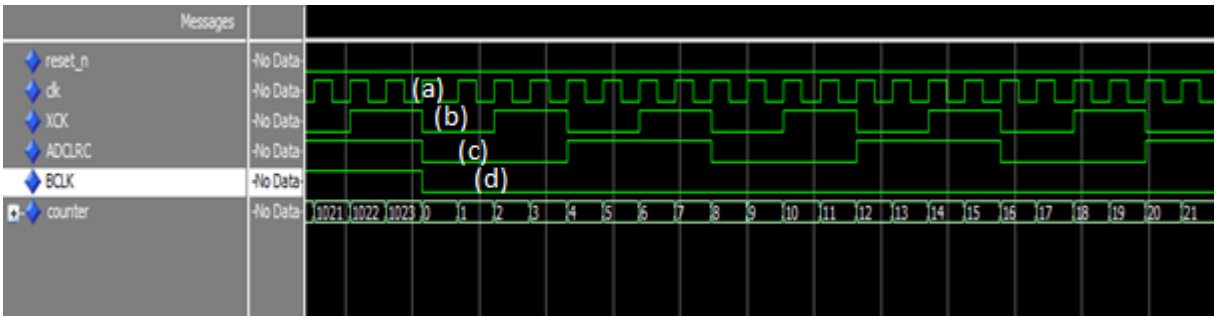


Figure 5.3. Résultat de la simulation avec ModelSim; (a) signal d'horloge 50MHz; (b) signal d'horloge 'XCK' ; (c) signal ADC LR; (d) signal Bit-Stream Clock

5.2.3 L'opération de débruitage

Comme il a été indiqué précédemment dans le chapitre 4 nous étions d'accord que les coefficients d'ondelettes scrutent puis marquent les discontinuités survenues sur le signal reçu. Ils correspondent donc aux détails. Si l'on veut appliquer une opération de seuillage (thresholding) sur ces coefficients, cela consiste à éliminer les détails les plus fins caractérisant la partie nocive du bruit. Il en découle donc deux grandes applications de cette technique de seuillage :

— La compression du signal : où nous n'allons conserver que les coefficients d'ondelettes les plus importants ce qui nous conduit à coder le signal et en représenter chaque échantillon avec un nombre réduit de bits.

— Le débruitage ou ("denoising"): de la même façon, nous ne garderons que les coefficients les plus importants et nous annulons ensuite le reste des coefficients avant de procéder à l'étape de reconstitution du signal. Il est à noter que le bruit occupe l'espace résiduel notifié par les faibles détails ; donc il sera éliminé par cette opération de seuillage appliquée notamment sur l'ensemble des coefficients de détails. Nous obtenons alors un signal nettement meilleur et plus "lisse" donc débruité. Pratiquement par un seuillage bien déterminé nous pouvons filtrer le signal du bruit par l'élimination de l'ensemble ou d'une partie des détails.

Ces deux applications sont très attachées et obéissent au même principe mais nous nous contenterons, dans ce travail, uniquement de l'opération de débruitage.

5.2.4 Principe de débruitage par la transformée en ondelettes

Nous pouvons envisager trois étapes essentielles lors de l'opération de débruitage d'un signal entaché par un bruit additif en moyennant la DWT adoptée dans ce contexte là.

- **L'analyse ou la décomposition :** dans cette étape nous veillons à bien choisir le type d'ondelette analysante ainsi que le niveau de décomposition à utiliser dans cette application.
- **Le seuillage :** Il va de même pour cette étape où la fonction de seuillage, que se soit dur ou doux, voit principalement son application sur les détails et peut être même étendue, le cas échéant, sur les approximations.
- **La synthèse ou la reconstruction :** c'est à cette étape qu'il incombe de restituer le signal ayant subi le traitement de seuillage selon une structure appropriée faisant intervenir la contribution de l'approximation du dernier niveau de décomposition

combiné avec l'ensemble des détails, après traitement, pour les différents niveaux de décomposition.

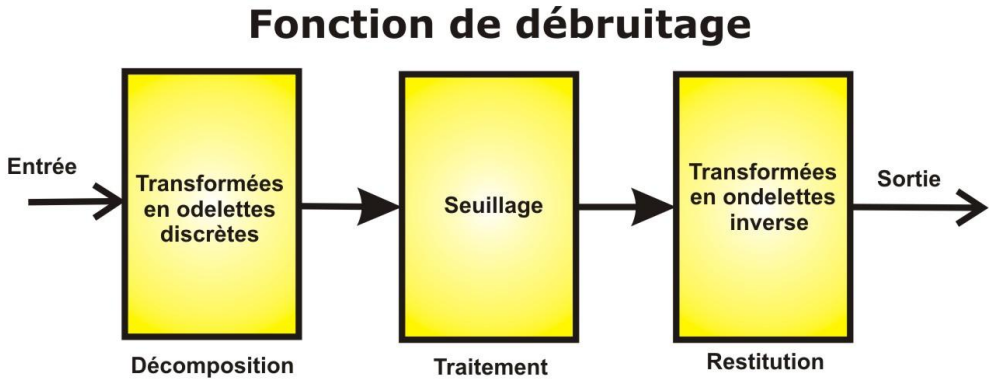


Figure 5.4. Schéma bloc de la fonction débruitage

5.2.4.1 Choix des ondelettes et niveau de décomposition :

Il n'y a malheureusement pas une ondelette qui soit meilleure par rapport aux autres types. Au fait, le choix de l'ondelette dépend de l'application en soi, dans certains cas, l'ondelette la plus simple (Haar) semble optimale par contre dans d'autre cas, elle ne sera pas adaptée. Pour la plupart des applications et en pratique, il semblerait que l'élément le plus important soit le nombre de moments nuls de l'ondelette. Il est souhaitable d'avoir le plus de coefficients d'ondelettes nuls et donc plus de moments nuls implique une meilleure transformation.

Les ondelettes bi-orthogonales se distinguent des celles orthogonales par le type d'ondelette utilisée. Les ondelettes bi-orthogonales utilisent deux familles d'ondelettes toutes différentes, une pour l'analyse et une autre pour la reconstruction. Malgré les temps de calcul plus longs les ondelettes de Daubechies d'ordre 2 sont les plus régulières, elle présente deux moments nuls, contrairement aux ondelettes de Haar.

Le choix d'ondelettes dépend principalement de l'application. En traitement de l'image par exemple on préférera utiliser des ondelettes bi-orthogonales (Cohen-Daubechies-Feauveau) qui permettent de concilier les propriétés de linéarité de phase avec la compacité des supports des filtres utilisés.

Nous avons choisi les ondelettes de daubechies car ce sont des fonctions qui ont beaucoup de moments nuls (une fonction f à n moments nuls si elle est orthogonale à l'espace Rn[X] des polynômes de degrés n ou moins). Ce type d'ondelettes permet une reconstruction rapide de fonctions régulières. Elles ont une réponse en fréquence équilibrée, mais une

réponse en phase non-linéaire justifiant l'utilisation d'un tel type d'ondelettes qui est utile dans la réduction du bruit du signal et il est bien adapté pour des applications de compression de données [93]. Ce type ondelettes s'est avéré plus efficace pour les applications audio que celui de Haar [94]. Les ondelettes daubechies sont utiles dans la réduction du bruit du signal et se prête bien à la compression de données [95].

Les formes d'ondelettes sont faciles à obtenir de même que les coefficients des filtres associés. Pour trouver ces coefficients, il suffit d'utiliser, la fonction Matlab : wfilters('nom d'ondelettes'). Par exemple, pour obtenir les coefficients des filtres associés à une ondelette db2, on effectue la commande suivante :

[lo_d, hi_d, lo_s, hi_s] = wfilters('db2') Où

- Lo_d représente la décomposition du filtre passe-bas.
- Hi_d représente la décomposition du filtre passe-haut.
- Lo_s représente la reconstitution du filtre passe-bas.
- Hi_s représente la reconstitution du filtre passe-haut.

L'exécution de cette commande donne le résultat suivant :

lo_d = -0.1294 0.2241 0.8365 0.4830

hi_d = -0.4830 0.8365 -0.2241 -0.1294

lo_s = 0.4830 0.8365 0.2241 -0.1294

hi_s = -0.1294 -0.2241 0.8365 -0.4830

Le tableau 5.1 illustre les coefficients pré-calculés des différents filtres de daubechies qui vont être utilisée dans cette thèse [43].

TABLEAU 5.1. COEFFICIENTS DES DIFFERENTS FILTRES DE DAUBECHIES

Filtres de décomposition		Filtres de reconstitution	
Hd (Z)	***Gd (Z)***	***Hr (Z)***	***Gr (Z)***
-0.1291	-	0.48301	-0.1291
0.2241	0.48	0.8365	-0.2241
0.8365	301	0.2241	0.8365
0.48301	0.8365	-0.1291	-
	-0.2241		0.48
	-0.1291		301

Le choix du nombre de niveaux est un point important dans l'analyse par ondelette. Il dépend de l'application, c.à.d. la nature du processus en vigueur et du type du signal à traiter. Dans le cas du débruitage, le signal à débruiter est complexe, plus il est difficile de définir un niveau optimal. D'après les tests de simulation, il faut aller au moins au-delà de trois niveau pour obtenir un débruitage d'essai plus au moins acceptable. De ce fait, nous avons opté pour un choix de trois niveaux de décomposition.

5.2.4.2 Banc de filtre d'analyse (décomposition)

Aujourd'hui, les ondelettes et les bancs de filtres ont montré leur importance dans de nombreux domaines d'applications du traitement du signal et images (analyse, débruitage, restauration, compression etc…) et notamment en codage d'image. Par conséquent Plusieurs travaux sont consacrés à l'implémentation de ces bancs de filtres.

Un banc de filtres est un ensemble de filtres possédant une structure spécifique à l'application voulue. Considérons maintenant un banc de filtres à deux bandes d'analyse, avec une entrée, comme s'est indiqué sur la figure 5.5 dont les fonctions de transferts du filtres $H_a(Z)$ et $G_a(Z) = H_a(-Z)$

Elle représente deux filtres, un de type passe-bas H(z) (coefficients d'approximations) et l'autre de type passe- haut G(z) (coefficients de détails), la fonction de transfert d'une telle structure est représentée dans la figure 5.5.

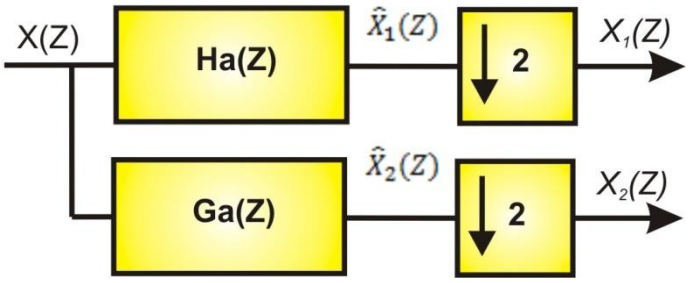


Figure 5.5. Banc de filtre à deux canaux

- **Représentation polyphase**

L'invention de la représentation polyphase d'un filtre a permis la simplification de plusieurs résultats théoriques en traitement du signal et a conduit à des implémentations plus simples des bancs de filtres. L'idée de base pour un banc de filtre à deux canaux est la

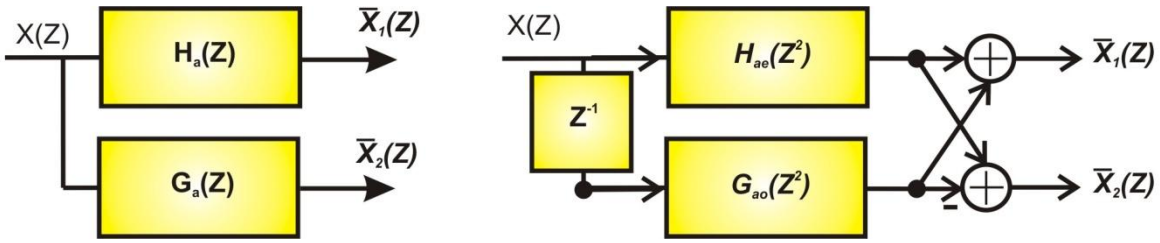
séparation des coefficients d'indices pairs (*even*) de ceux d'indices impairs (*odd*), il vient donc :

$$H_a(Z) = H_{ae}(Z^2) + Z^{-1}H_{ao}(Z^2) \qquad (6.1)$$

$$G_a(Z) = G_{ae}(Z^2) + Z^{-1}G_{ao}(Z^2) \qquad (6.2)$$

A partir de la propriété du filtre $G_a(Z) = H_a(-Z)$ on aura

$$G_a(Z) = H_{ae}(Z^2) - Z^{-1}H_{ao}(Z^2) \qquad (6.3)$$



(a) (b)

Figure 5.6. La décomposition à deux canaux d'un banc de filtre

Chaque bloc a un filtre polyphase commun dans les deux canaux d'analyse. La figure 5.6 (a) représente la structure ordinaire du filtre, par contre la structure polyphase est présentée sur la figure 5.6 (b) et se donne par la relation suivante :

$$\bar{X}_1(Z) = H_{ae}(Z^2) \cdot X(Z) + Z^{-1}G_{ao}(Z^2) \cdot X(Z)$$

$$\bar{X}_1(Z) = H_a(Z) \cdot X(Z) \qquad (6.4)$$

$$\bar{X}_2(Z) = H_{ae}(Z^2) \cdot X(Z) - Z^{-1}G_{ao}(Z^2) \cdot X(Z)$$

$$\bar{X}_2(Z) = G_a(Z) \cdot X(Z) \qquad (6.5)$$

Finalement, un sous échantillonnage (décimation) avec un facteur de 2 est mis en série avec le filtre (on prend un échantillon parmi deux). Exploitons la première identité noble. Donc ces décimateurs peuvent être reportés à l'entrée filtre. De cette manière, les échantillons de sortie sont donc tous utiles. Dans l'optique de ces propriétés de décimation ainsi que le nombre d'opérations divisé en deux.

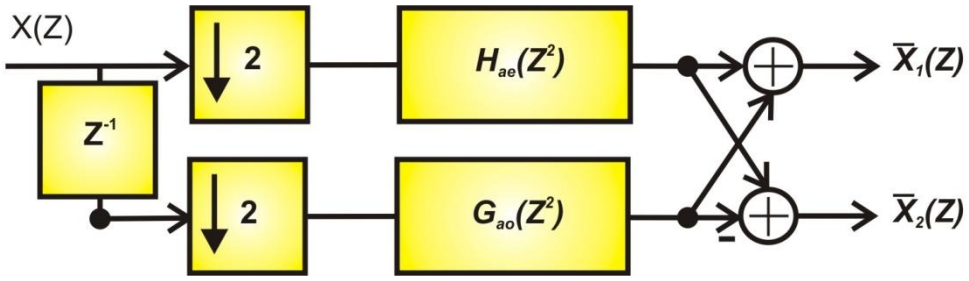


Figure 5.7. Banc de filtre de décomposition à deux canaux

Dans ce qui suit, nous nous concentrons sur l'analyse d'un banc de filtres à deux canaux. En utilisant la décomposition des deux fonctions de transferts $H_a(z)$ et $G_a(Z)$ en leurs structures polyphase on obtiendra :

$$H_a(Z) = H_{ae}(Z^2) + Z^{-1}H_{ao}(Z^2) \qquad (6.6)$$

$$G_a(Z) = G_{ae}(Z^2) + Z^{-1}G_{ao}(Z^2) \qquad (6.7)$$

Il découle de cette représentation la structure polyphasé d'un niveau de décomposition comme s'est illustré sur la figure 5.8 :

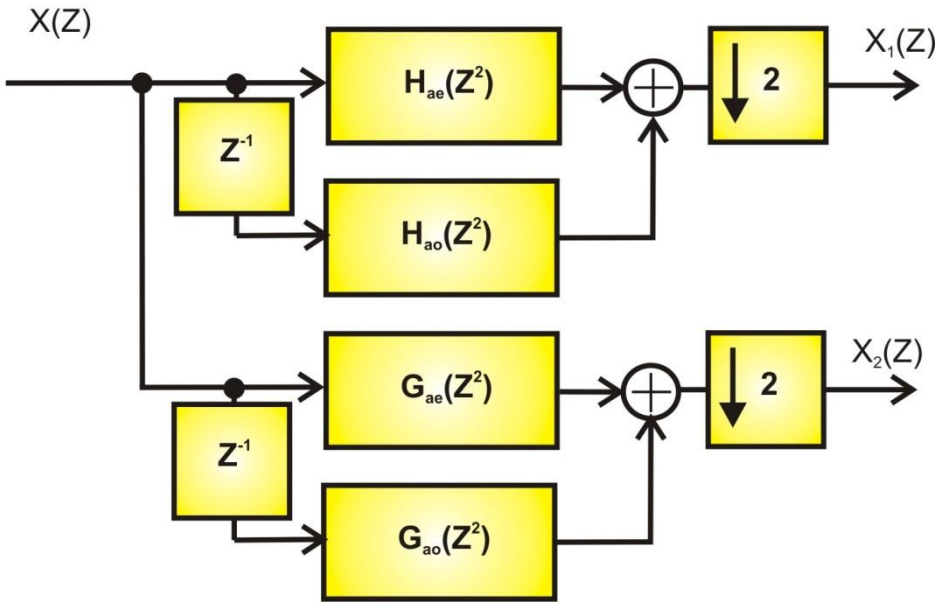


Figure 5.8. Décomposition polyphase de $H_a(Z)$ et $G_a(Z$)

En utilisant la première noble identité, nous obtiendrons ainsi la représentation de la figure 5.9.

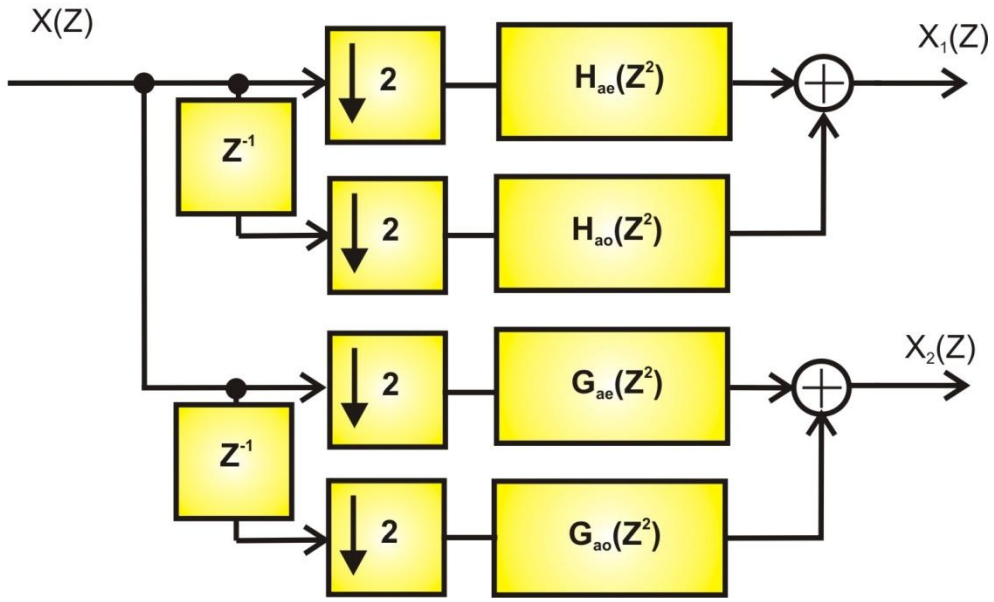


Figure 5.9. la structure polyphase d'analyse d'un banc de filtres

Le banc de filtre d'analyse avec la structure polyphase de la figure 5.9 est composé de deux parties : un aiguilleur (démultiplexeur) et un ensemble de filtre qui travaillent avec une cadence d'horloge réduite de facteur de 2. Ce système se définit par l'équation suivante :

$$\begin{bmatrix} X_1(Z) \\ X_2(Z) \end{bmatrix} = P(Z) \begin{bmatrix} X_e(Z) \\ X_o(Z) \end{bmatrix} \qquad (6.8)$$

$$P(Z) = \begin{bmatrix} H_{ae}(Z) & H_{ao}(Z) \\ G_{ae}(Z) & G_{ao}(Z) \end{bmatrix} \qquad (6.9)$$

5.2.4.3 Banc de filtre de synthèse (reconstitution)

Le banc filtre de synthèse est composé de deux filtres un passe-bas de fonction de transfert $H_s(Z)$et un autre passe-haut de fonction de transfert $G_S(Z)$, qui sont reliées par la relation :

$$G_s(Z) = -H_S(-H) \qquad (6.10)$$

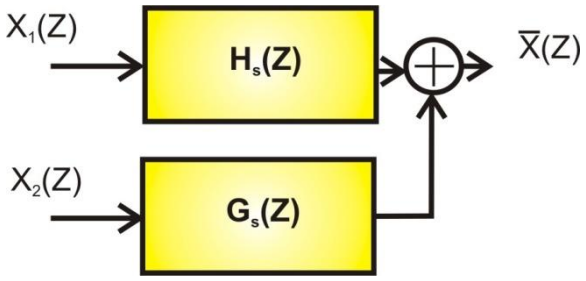
Si on utilise un filtre polyphase il vient donc :

$$H_S(z) = H_{se}(Z^2) + Z^{-1}H_{so}(Z^2) \qquad (6.11)$$

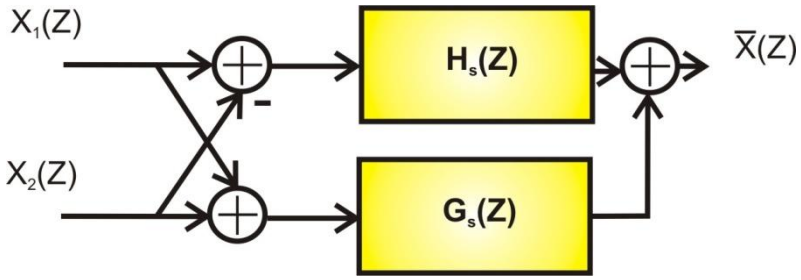
La fonction de transfert du filtre $G_S(Z)$ du filtre passe haut se déduit directement comme suit :

$$G_S(z) = -H_{se}(Z^2) + Z^{-1}H_{so}(Z^2) \qquad (6.12)$$

Les deux composantes polyphase peuvent être employées conjointement pour réaliser la structure d'un banc de filtre de reconstitution ou synthèse comme le montre la figure 5.12 :



(a)



(b)

Figure 5.12 : banc de filtre de reconstitution

$$\bar{X}(Z) = H_{se}\left(Z^2\right)[X_1(Z) - X_2(Z)] + Z^{-1}H_{so}\left(Z^2\right)[X_1(Z) - X_2(Z)]$$

$$\bar{X}(Z) = H_s(Z)X_1(Z) + G_s(Z)X_2(Z) \qquad (6.13)$$

Exploitons la deuxième identité noble (seconde noble identity). L'interpolation en amont des filtres peut en être déplacée en aval, par conséquence nous obtenons la structure d'un banc de filtre polyphase efficace, comme s'est représenté sur la figure 5.10 :

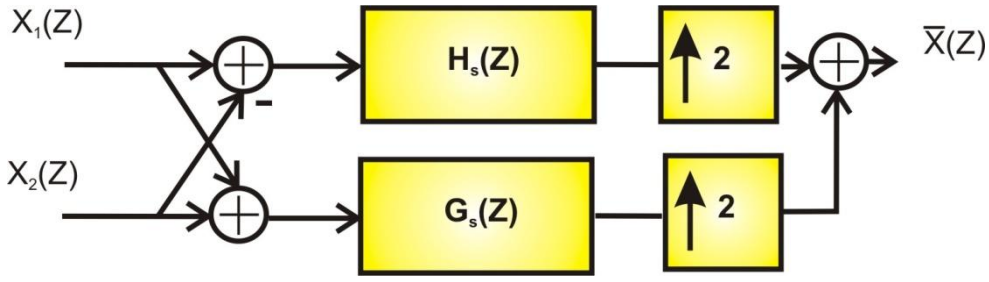


Figure 5.10. Banc de filtre de reconstitution (synthèse)

Un tel banc de filtre possède une structure un peu complexe qui est réduite rapport à la version originale. Nous allons utiliser un banc de filtres de deux canaux sous une structure polyphase comme s'est illustré sur la figure 5.11 :

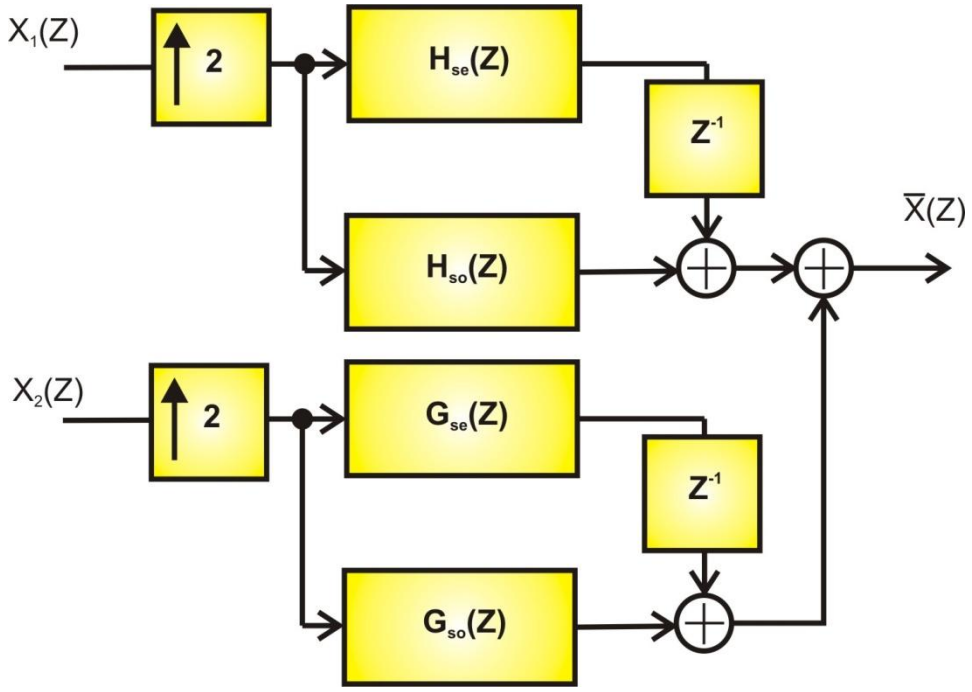


Figure 5.11. la structure polyphase de $H_s(z)$ et $G_s(Z)$

Pour relier, de façon simple, les vecteurs issus des filtres d'analyse à l'entrée des filtres de synthèse les fonctions de transferts $H_s(z)$ et $G_s(Z)$ doivent être exprimées par leurs quantités polyphasées.

$$H_a(Z) = Z^{-1}H_{se}(Z^2) + H_{so}(Z^2) \qquad (6.14)$$

$$G_a(Z) = Z^{-1}G_{se}(Z^2) + G_{so}(Z^2) \qquad (6.15)$$

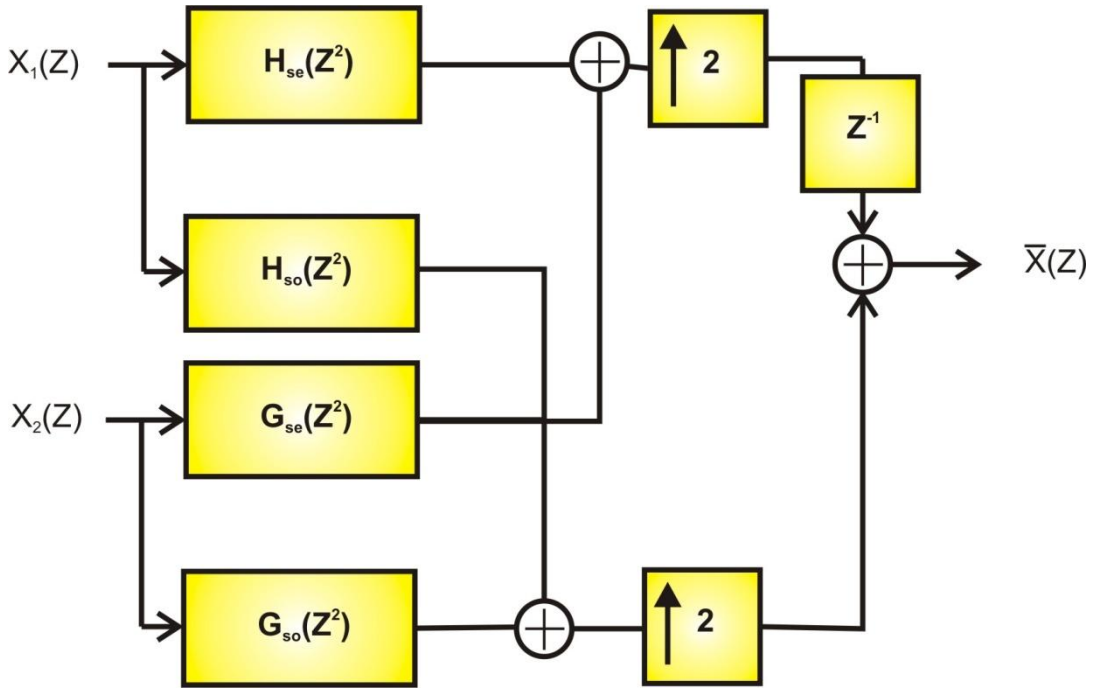


Figure 5.12. banc de filtre polyphase à deux canaux de la phase de synthèse

D'une manière identique, en utilisant la deuxième noble identité la représentation de la figure 5.11 devient comme le montre la figure 5.12

Le banc de filtre représenté sur la figure 5.12 est composé de deux parties distinctes. Une sortie multiplexée, et un système discret à deux entrées et deux sorties qui sont définis par l'évolution matricielle suivante :

$$\begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \end{bmatrix} = \begin{bmatrix} H_{se}(Z) & G_{se}(Z) \\ H_{so}(Z) & G_{so}(Z) \end{bmatrix} \begin{bmatrix} X_1(Z) \\ X_2(Z) \end{bmatrix} \qquad (6.16)$$

Cette structure est dite la matrice polyphase du banc de filtres de synthèse (reconstitution)

5.2.4.4 Le seuillage

Le seuillage est une technique utilisée pour le débruitage des signaux et des images. La transformation discrète en ondelette utilise deux types de filtres. Le filtrage moyenneur et le filtrage du détail. Quand nous décomposons un signal à l'aide de la transformation en ondelette nous obtenons une suite de coefficients qui inter-corrèlent avec la sous-bande de haute fréquence (détails). Si ces détails sont assez petits ils peuvent être négligés sans pour autant affecter les données de base. En outre, ces détails sont souvent liés directement au bruit

et en remplaçant ces coefficients par zéro on diminue les composantes de bruit. Dans un souci d'optimisation on pourra représenter le spectre du signal afin de pouvoir déterminer les composantes les plus importantes de ce dernier et donc celles que l'on doit garder. Ceci est la mission que doit accomplir l'opération de seuillage. Il existe deux principaux types de seuillage utilisés pour le débruitage du signal [96] qui sont le seuillage doux et le seuillage dur. La majorité des recherches effectuées utilisent toujours ces techniques comme principes de base pour le débruitage et tente de les modifier afin d'améliorer leurs performances. Pour cette raison, et dans notre cas, nous avons choisi d'implémenter un système avec ces deux types de seuillage décrit dans le chapitre VI qui sont :

- ***Seuillage doux (Soft tresholding)***
- ***Seuillage dur (Hard tresholding)***

5.2.4.5 Filtre décimateur de daubechies

Considérons maintenant le filtre de Daubechies de longueur 4 (ordre du filtre) représenté sur la figure 5.13 dont $H(Z)$ désigne sa fonction de transfert et $M = 2$ représente le facteur de décimation/interpolation [72]

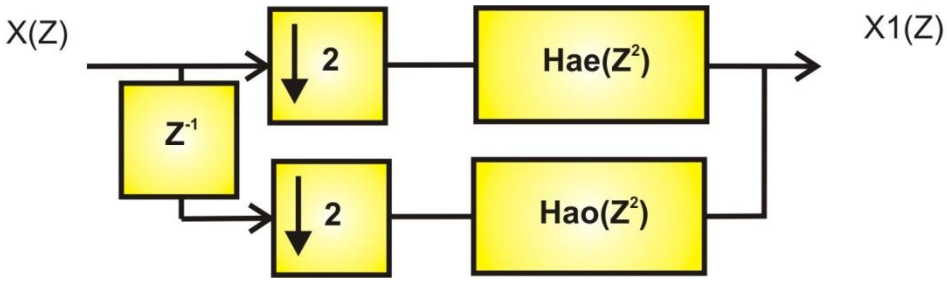


Figure 5.13. Filtre daubechies sous la structure polyphase

$$H(Z) = \left[\left(1+\sqrt{3}\right)+\left(3+\sqrt{3}\right)Z^{-1}+\left(3-\sqrt{3}\right)Z^{-2}+\left(1-\sqrt{3}\right)Z^{-3}\right]\frac{1}{4\sqrt{2}}$$

$$H(Z) = 0.48301 + 0.8365Z^{-1} + 0.2241Z^{-2} - 0.1294Z^{-3} \qquad (6.18)$$

Ainsi, le modèle mathématique décrivant le filtre de Daubechies peut être représenté par l'évolution matricielle suivante avec un facteur M = 2.

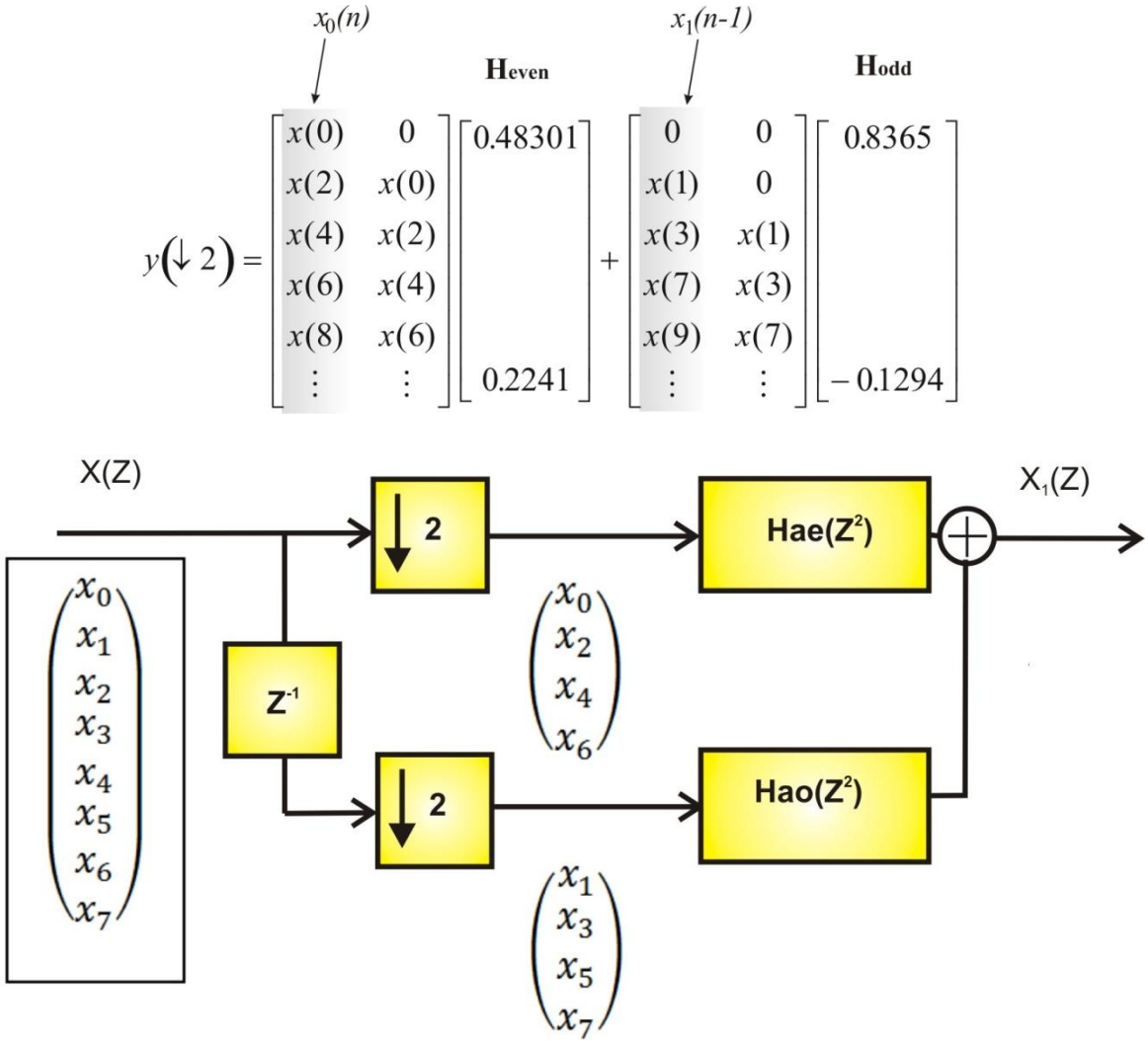


Figure 5.14. L'évolution de la structure polyphase du filtre de daubechies

5.2.5 Implémentation de l'opération de débruitage

5.2.5.1 La transformée en ondelettes discrète

L'avantage d'utiliser la transformée en ondelettes discrète sur transformée en ondelettes continue est de permettre un calcul rapide des ressources matérielles. Dans la transformée en ondelettes discrète, les composantes du signal sont traitées avec une résolution différente selon leurs fréquences. Le signal est transmis vers des multiples filtres numériques. Plusieurs niveaux de bancs de filtres numériques ont été utilisés pour effectuer le filtrage [97].

La transformée en ondelettes discrète utilise essentiellement des coefficients de filtrage pour traiter le signal entrant en provenance du convertisseur ADC qui se trouve sur 8 bits. En vertus de la base de ces échantillons la transformée en ondelettes discrète peut offrir les échantillons de sortie.

5.2.5.2 Analyse multi-résolution

La figure 5.15 représente la décomposition à trois niveaux, où chaque niveau est composé de deux filtre passe bas *Hd(Z)* et passe haut *Gd(Z)* précédé d'un décimateur. Les coefficients de sortie du filtre passe bas s'appelle les approximations, les coefficients de sortie du filtre passe haut s'appelle les détails. Le nombre de coefficients après chaque niveau de décomposition à cause du décimateur est la moitié du nombre initial ce qui donne une fréquence divisée par deux. Le signal d'entrée *xin[n]* est décomposé en différents coefficients qui sont l'approximation *A3[N]*, et l'ensemble de détails *D1[n]*, *D2[n]*, *D3[n]* transportant le bruit.

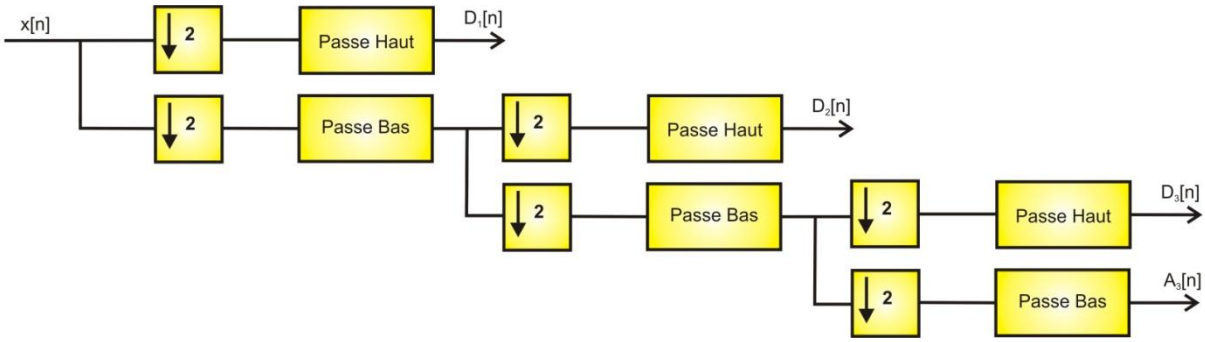


Figure 5.15. Les trois niveaux de décomposition

La reconstruction ou la synthèse du signal est exactement la procédure inverse que celle obtenue dans la phase d'analyse comme le montre la figure 5.16. dans cette phase de synthèse où le signal de sortie se compose de la dernière approximation *A3[N]*, et de l'ensemble de détails *D1[n]*, *D2[n]*, *D3[n]* qui ont subis un traitement apriori de débruitage efficace en ayant recours au seuillage approprié. L'interpolation ou sur-échantillonnage avant les filtres consiste à insérer des zéros entre les échantillons ce qui double la taille du signal ce qui réclame d'introduire un doubleur de fréquence [97].

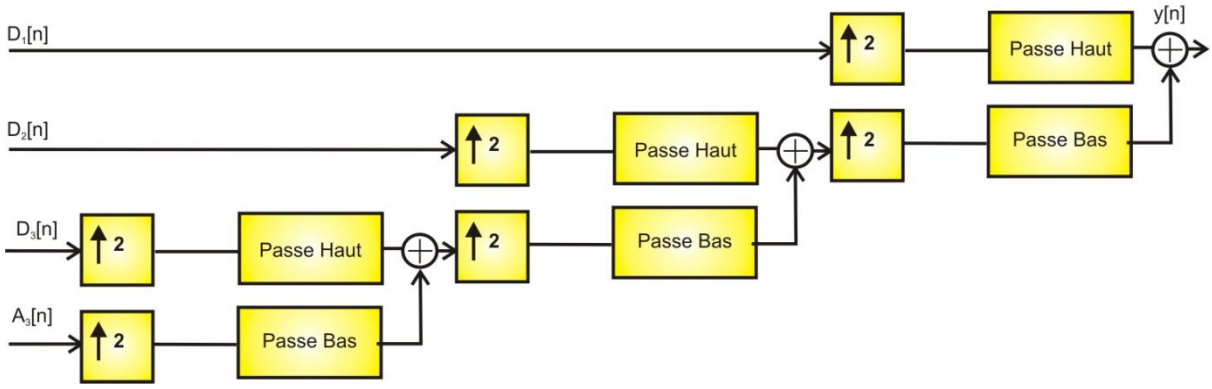


Figure 5.16. les trois niveaux de synthèse (reconstitution)

La figure 5.17 présente l'architecture globale de la procédure de débruitage d'un signal sensé être entaché par un bruit additif. Cette représentation montre les différents blocs constituant le design à mettre en application.

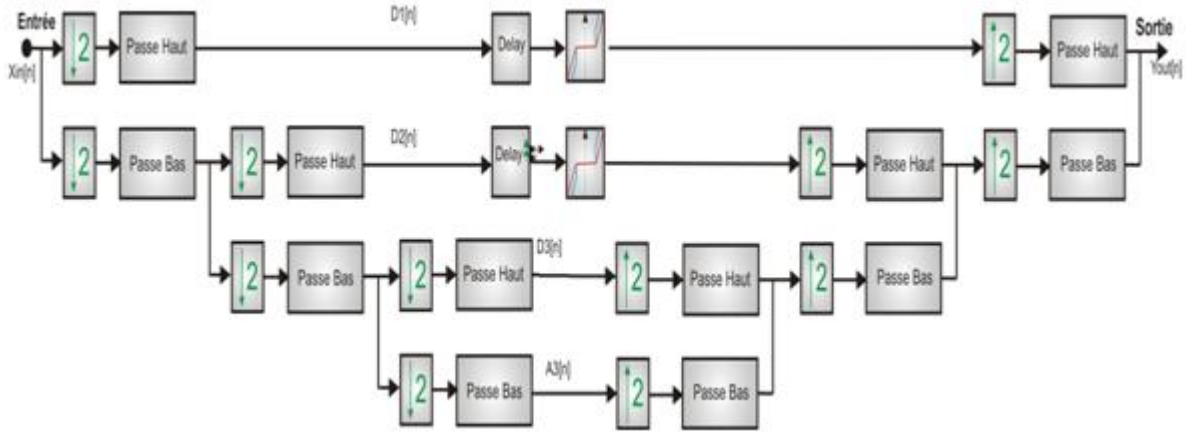


Figure 5.17. Schéma bloc de la fonction débruitage.

5.2.5.3 La multiplication par une constante

Il existe plusieurs méthodes d'implémentation de la transformée en ondelettes discrète et la transformée en ondelettes discrète inverse, qui requièrent des processus répétées de multiplications. Pour remédier à toute utilisation des multiplieurs classiques qui ont un temps d'exécution important et qui sont gourmand au niveau des éléments logique. Et comme, l'un des multipliant est une constante (constant coefficient multiplier ou KCM), alors il est plus commode d'utilisé un tableau qui ne dispose que d'une colonne correspondant aux résultats pré-calculées de la multiplication de la donnée par le coefficient et utilisé la donnée pour adresser ce tableau. En effet, l'utilisation des LUTs est une technique couramment utilisée pour accélérer le traitement numérique dans les applications temps réel [91]. L'idée de base est de pré-calculer le résultat d'opérations complexes qui peuvent être exprimées en fonction d'une valeur entière. Les résultats pré-calculées sont généralement stockées dans un tableau qui va utiliser la valeur de l'échantillon de la donnée comme une adresse, puis faire sortir adéquatement la valeur qui correspond au résultat de calcule, plutôt que d'effectuer la multiplication classique qui se voit gourmande en terme des éléments logiques et qui va prendre plus de temps ce qui introduit des retard de calcule remarquable. Ainsi, lors de l'opération de filtrage soit dans la phase d'analyse ou de synthèse l'utilisation d'une LUT est essentielle pour décrire le concept du multiplieur. C'est une technique recommandée pour

réduire le temps d'exécution et nombres des éléments logiques ce qui est bien adapté aux circuits FPGAs.

Dans notre cas, la taille des résultats pré-calculés étant de 16-bits, et la taille des échantillons d'entrée est de 8-bits, et si on veut construire une LUT il faut bien réserver l'espace mémoire correspondant. Notre LUT doit disposer de 2^8=256 cases de 16-bits figure.5.18, chose qui n'est pas optimale à notre application.

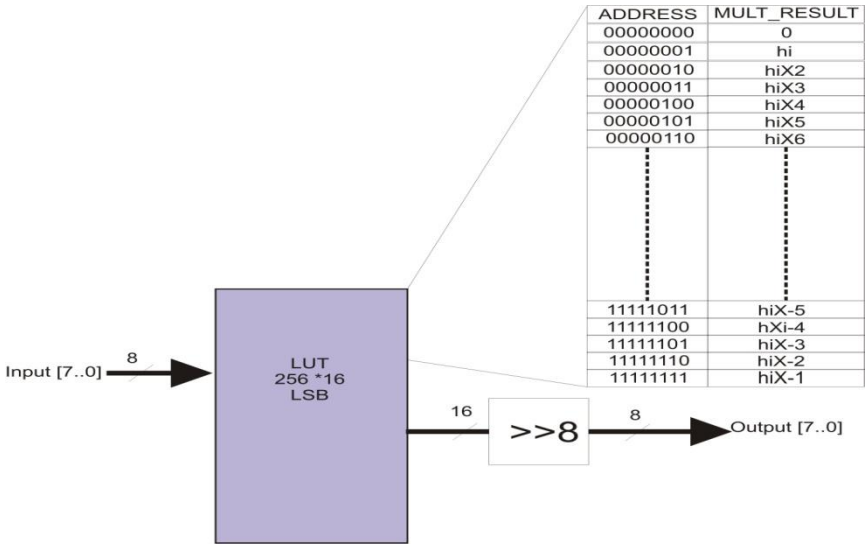


Figure 5.18. l'implémentation d'une LUT en utilisant le concept KCM.

Afin de représenter les différents coefficients de l'ondelette de Daubechies utilisés nous avons pensé à réduire la taille des différents LUTs utilisées pour chaque coefficient. Cette approche réduit la taille de la LUTs de : 2^8 à 2^5 cases mémoire de 12-bits, et elle la divise en deux partie chaque une est constitue de 2^4 cases mémoires de 12-bits. Les données d'entrée sont fractionnées en deux parties (LSB et MSB) pour se référer à l'emplacement de chaque partie de la mémoire où se trouve une partie du résultat de la multiplication des données et de coefficient normalisé figure.5.19.

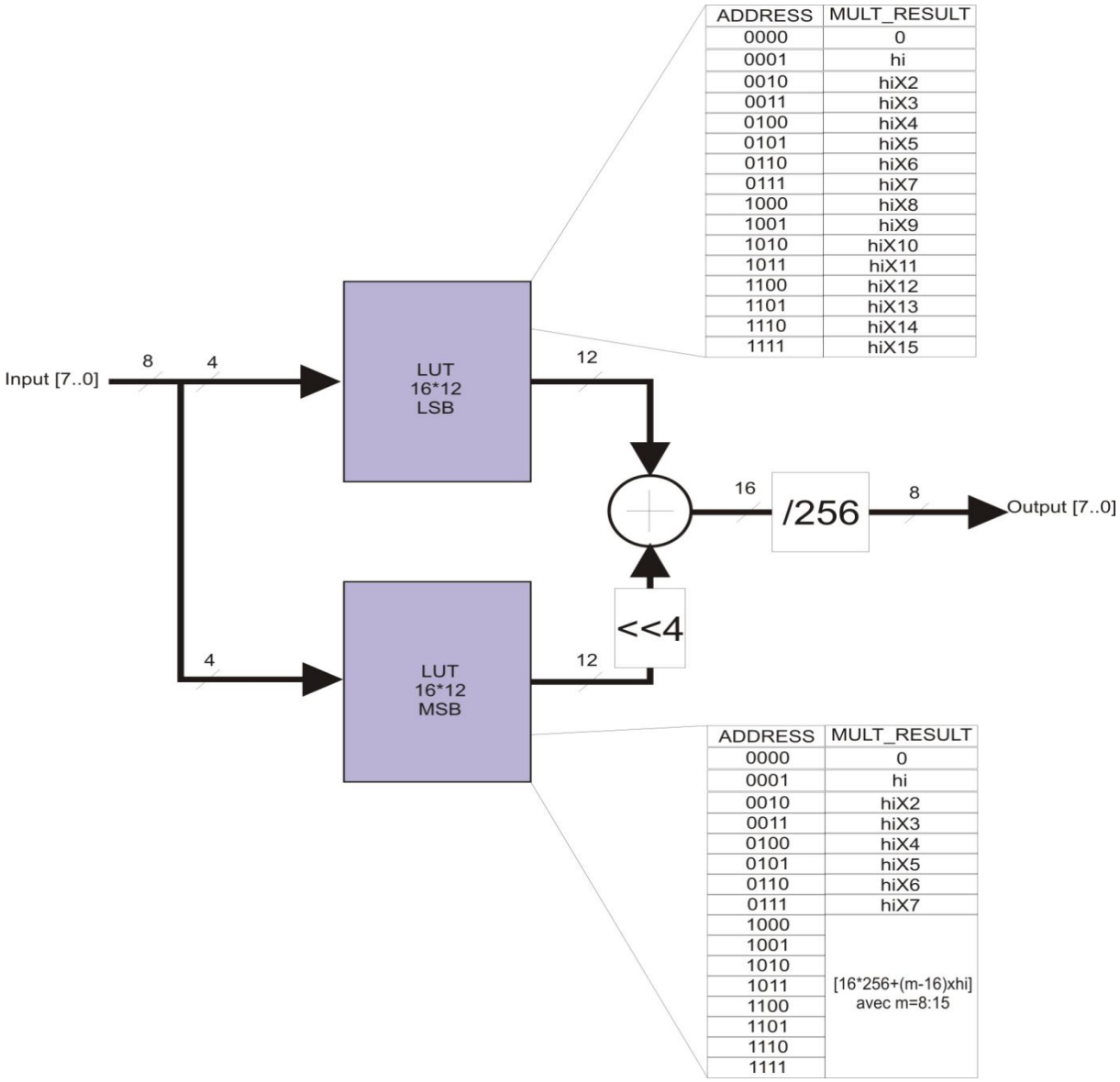


Figure 5.19. l'implémentation d'une LUT en utilisant le concept KCM.

Pour le cas des données négatives, il est important de noter qu'un décalage est utilisé pour les traiter dans l'intervalle [-128, -1]. Ce décalage peut être déterminé par l'équation suivante :

$$[16 \times 256 + (m - 16) \times h_i] \quad avec\ m = 8 \dots 15 \qquad (5.19)$$

Ces nouveaux résultats sont stocké dans la partie [1000…1111] de la LUT contenant la partie référer par la partie MSB parce que cette partie de la donnée est signée, par contre la LUT référer par la partie LSB elle contient des valeurs qui sont non signées.

Les deux exemples suivants vont clarifier la procédure de calcul des nombres négatifs.

Exemple1 : $x = -128\left(\frac{57}{256}\right) = -28.5 \cong -29$

Où -128 représente la valeur de l'échantillon d'entrée, et 57 la valeur du coefficient.

$(-128)_{10} = (1000\ 0000)_2$

Li[0000]= $(0000000000000000)_2=(0)_{10}$

Hi[1000]= $(1110001110000000)_2=(58240)_{10}$

$$x = \frac{(Hi + Li)}{256} = \frac{58240}{256} = (227)_{10} = (11100011)_2 = -128 + 99 = -29$$

Exemple2 : $x = -3\left(\frac{214}{256}\right) = -2.5 \cong -2$

Où -3 représente la valeur de l'échantillon d'entrée, et 214 la valeur du coefficient.

$(-3)_{10} = (1111\ 1101)_2$

Li[1101]= $(0000101000001000)_2=(2568)_{10}$

Hi[1111]= $(1111001110100000)_2=(62368)_{10}$

$$x = \frac{(Hi + Li)}{256} = \frac{2568 + 62368}{256} = \frac{64936}{256} = (253{,}65625)_{10} = (11111110)_2 = -2$$

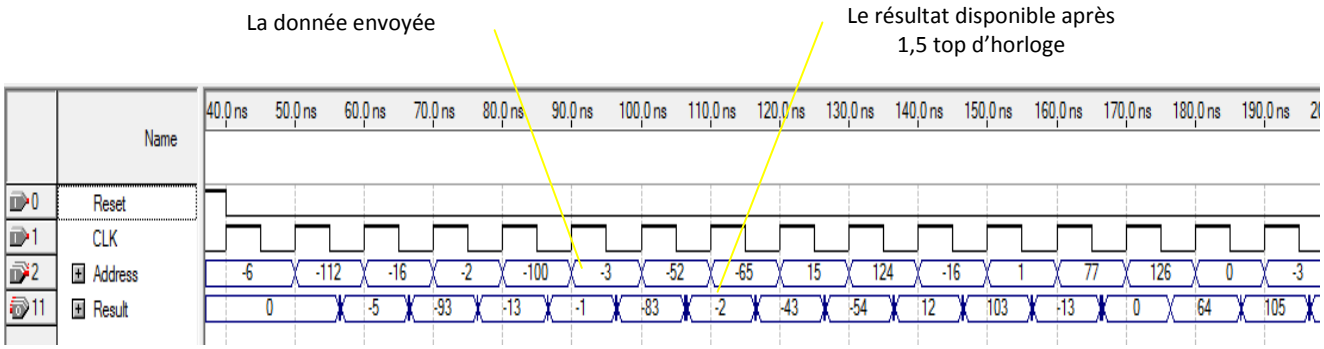


Figure 5.20. Résultat de la simulation par Quartus ® II de la LUT du coefficient 0.8365.

La figure 5.20. montre le résultat de la simulation pour le coefficient 0.8365. Le cas de l'exemple 2. Où il est multiplié l'entrée (Adress) qui présente une valeur décimale de -3 par le coefficient, on observe bien sur la simulation que la sortie (Result) est égale à -2 ce qui concorde avec ce qui montre l'exemple.

Cette technique de multiplication admet à chaque top d'horloge une donnée elle fournit le résultat à la sortie après 1.5 top d'horloge (une latency =1.5). Ce qui le rend utile à des traitements

nécessitant des grandes vitesses. L'autre avantage de cette technique est que la vitesse de traitement est indépendante de la taille de la donnée.

La figure 5.21. donne la structure RTL générée par **Quartus ® II** de ce type de multiplicateur :

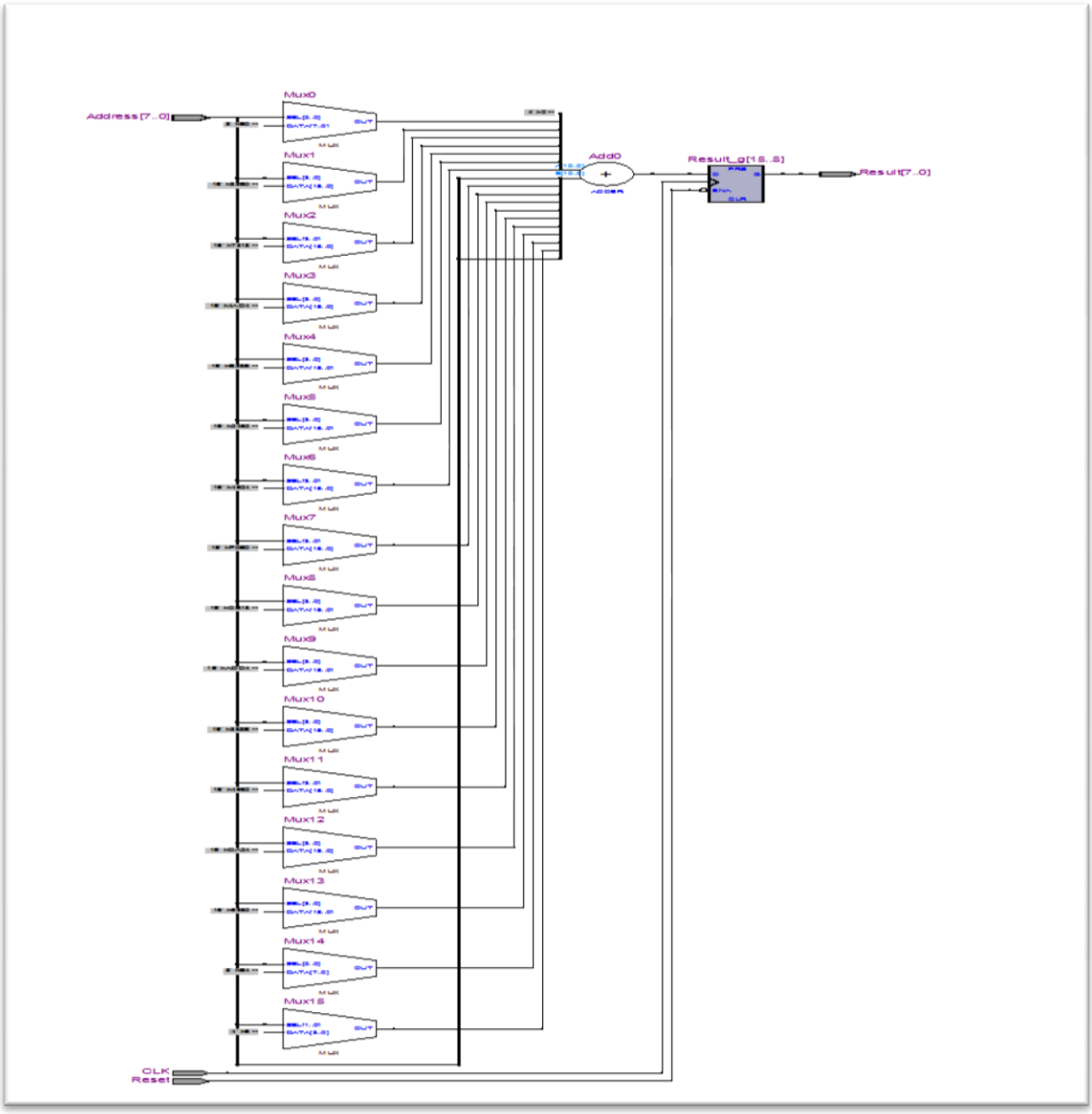


Figure 5.21. Architecture RTL donnée par Quartus ® II du multiplicateur par le coefficient 0.8365

5.2.5.4 Représentation binaire des nombres décimaux

Les chiffres représentés dans le système sont des nombres normalisés, ce qui signifie que le nombre le plus élevé vaut 1. Toutes les autres valeurs ont une amplitude inférieure à 1, ce qui constitue une exigence, alors, pour un système de représentation binaire à 8 bits pour les données du signal se sont des nombres biaisés. Par exemple, le nombre 0.056202 peut être représenté par la séquence binaire 00001110 avec un biais de 8. Le nombre 00001110 représente la valeur (2^1+2^2+ 2^3), mais avec un biais de 8, il s'en suit donc:

$$00001110 = (2^1 + 2^2 + 2^3) \times 2^{-8} = (2^{-7} + 2^{-6} + 2^{-5}) = 0.0546875$$

Avec cette représentation, tous les biais d'un nombre donné peuvent être utilisés pour représenter les nombres à point fixes. Cependant, le coût d'utilisation d'une telle représentation est directement proportionnel à la résolution imposée par ce choix-là. Dans notre application nous avons choisi un biais de 8 pour en faciliter l'implémentation.

Au sein du LUT, les valeurs sont de 0 à 255 (toutes les valeurs possibles de l'entrée à 8 bits). Pour obtenir la représentation binaire des coefficients la procédure suivante a été retenue :

$$h = 0.2241$$

$$0.2241 \times 2^8 = 57.3696$$

$$57.3696 = 57 = (00111001)_{binaire}$$

Ce processus multiplie les coefficients par 256 ou 2^8 (ce qui représente le biais) et le résultat doit être arrondit à l'entier le plus proche. Il s'agit de la valeur binaire approximative du nombre en utilisant un biais de 8. Tout au long du processus de débruitage, le biais de 8 a été utilisé pour maintenir la simplicité du circuit de traitement.

Il ya de nombreux coefficients nécessaires pour le processus de la transformation en ondelette. Quatre coefficients de la transformée en ondelettes discrète basée sur la théorie des ondelettes de Daubechies. Tel que mentionné précédemment, Ces coefficients sont 0.483, 0.8365, 0.2241 et -0.1294 qui sont désignés par h_1, h_2, h_3, et h_4, respectivement. La représentation des coefficients utilise un biais de 8. Cependant, puisque h_4 est un coefficient négatif. Le système nécessite l'utilisation de la représentation complément à 2.

5.2.5.5 Un niveau de décomposition

Les différentes parties du système approprié dédié à un niveau de décomposition du signal d'entrée en deux parties caractérisant les approximations et les détails sont montrés sur la figure 5.22 :

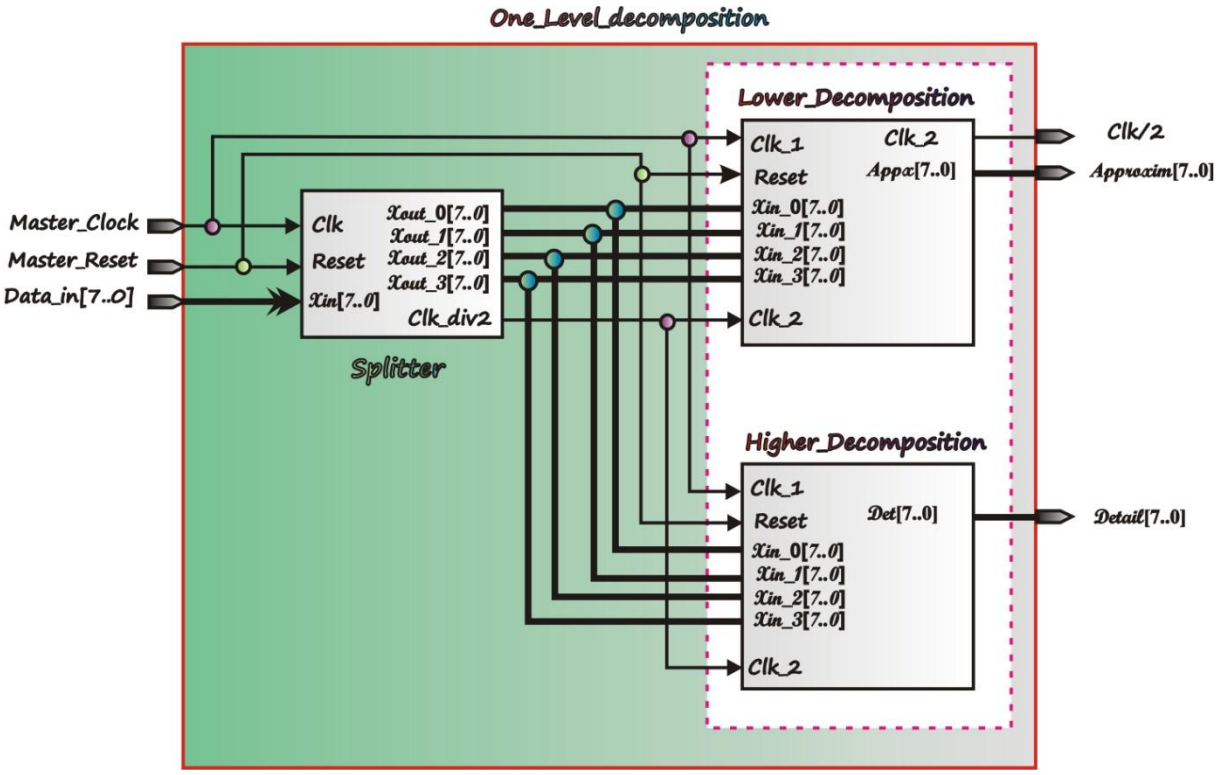


Figure 5.22. L'Architecture d'un niveau décomposition

A l'instant de la construction nous avons rencontré deux problèmes majeurs, à savoir, la causalité du filtre et le décimateur qui doit être mis juste avant les deux types de filtre. La causalité du filtre est atteint par l'adoption d'une fenêtre coulissante qui agit comme un buffer dont la taille correspond au nombre de coefficients du filtre (4 coefficients dans notre cas) pour éviter l'effet de bord. Le décimateur est modélisé par une machine d'état bi-state (FSM) dont le but est de prélever un échantillon parmi deux à une cadence de *Fs/2*. Cela nous à conduit à la conception d'un composant spécial que nous avons baptisé «séparateur-diviseur» comme il est illustré sur la figure 5.23 et qui a pour vocation d'effectuer ces deux tâches (il consiste à scinder le flux de données en deux familles pairs et impairs en plus la division de la fréquence initiale par deux) [82,83].

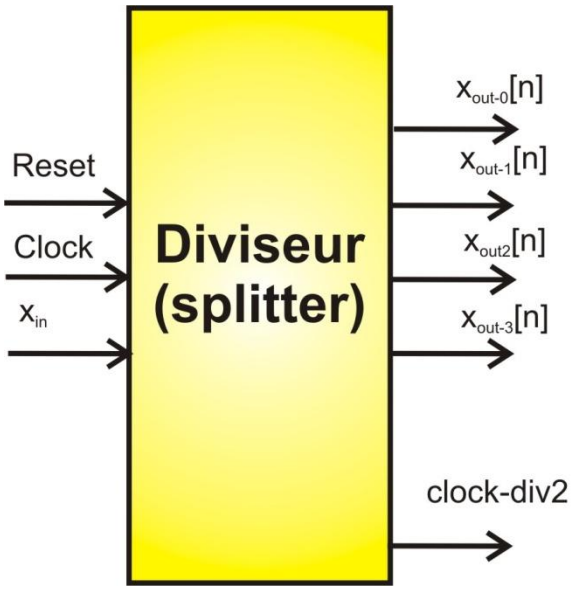
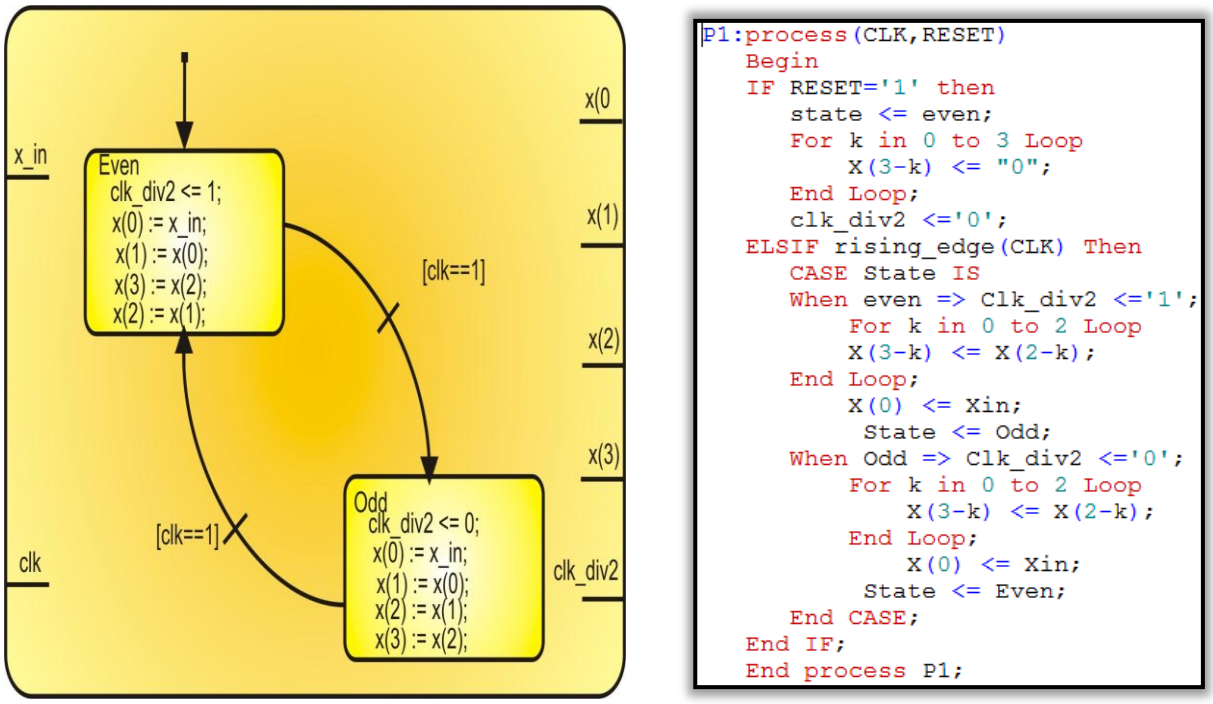


Figure 5.23. Schéma bloc du diviseur.




```
P1:process(CLK,RESET)
   Begin
   IF RESET='1' then
      state <= even;
      For k in 0 to 3 Loop
          X(3-k) <= "0";
      End Loop;
      clk_div2 <='0';
   ELSIF rising_edge(CLK) Then
      CASE State IS
      When even => Clk_div2 <='1';
          For k in 0 to 2 Loop
          X(3-k) <= X(2-k);
      End Loop;
          X(0) <= Xin;
           State <= Odd;
      When Odd => Clk_div2 <='0';
          For k in 0 to 2 Loop
              X(3-k) <= X(2-k);
          End Loop;
              X(0) <= Xin;
           State <= Even;
      End CASE;
   End IF;
   End process P1;
```

Figure 5.24. Machine à états finis et programme VHDL du Diviseur.

Comme évoqué précédemment, le composant "splitter" ne fournit pas seulement les données sollicitées à l'entrée des filtres, mais il agit également comme un diviseur de fréquence par deux de l'horloge d'échantillonnage mère afin de bien effectuer l'opération de décimation ou le sous échantillonnage comme on peut le voir facilement sur le diagramme de la simulation de la figure 5.25. Évidemment, pour répondre à cette exigence, on doit choisir le front descendant pour calculer la somme des quatre opérations résultant des différents blocs pour chaque coefficient. Ainsi, un détecteur de dépassement supérieur / inférieur devrait

être envisagé après chaque calcul arithmétique pour être compatible avec la contrainte de la mise en œuvre (taille des données). Notez que le même principe s'applique à l'ensemble des organes constituant l'application que ce soit dans la phase d'analyse ou dans la phase de synthèse [100], [101].

Le résultat de la simulation par Modelsim de ce composant est donné sur la figure.5.26, comme le montre la simulation, la fréquence de l'horloge de la sortie (clk2) est la division par deux de la fréquence de l'horloge mère (clk) et l'effet de la fenêtre glissante (xout0, xout1, xout2 et xout3) sur le signal d'entrée (xin). Le signal state représente les deux états de la machine d'état finie (odd et even).

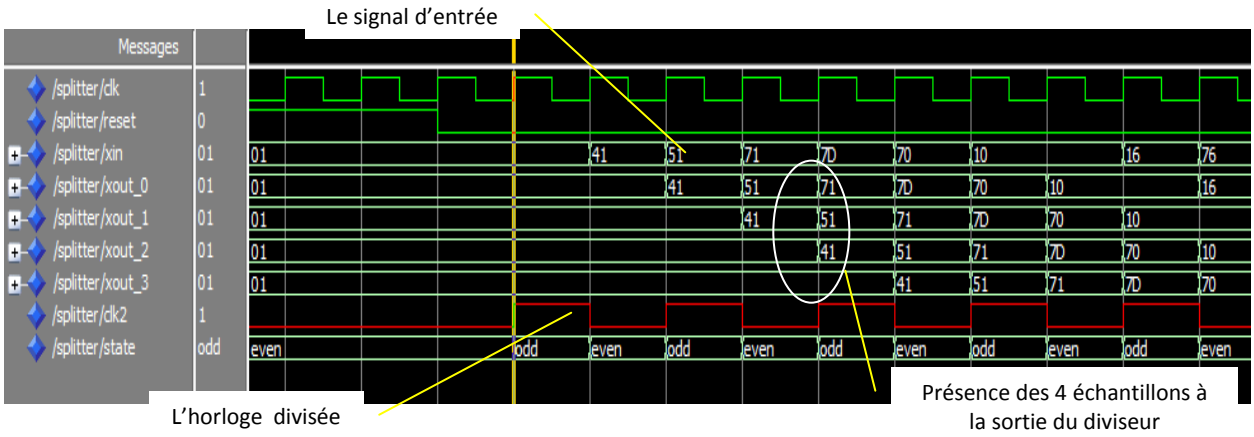


Figure 5.25. Le résultat de la simulation par Modelsim du diviseur.

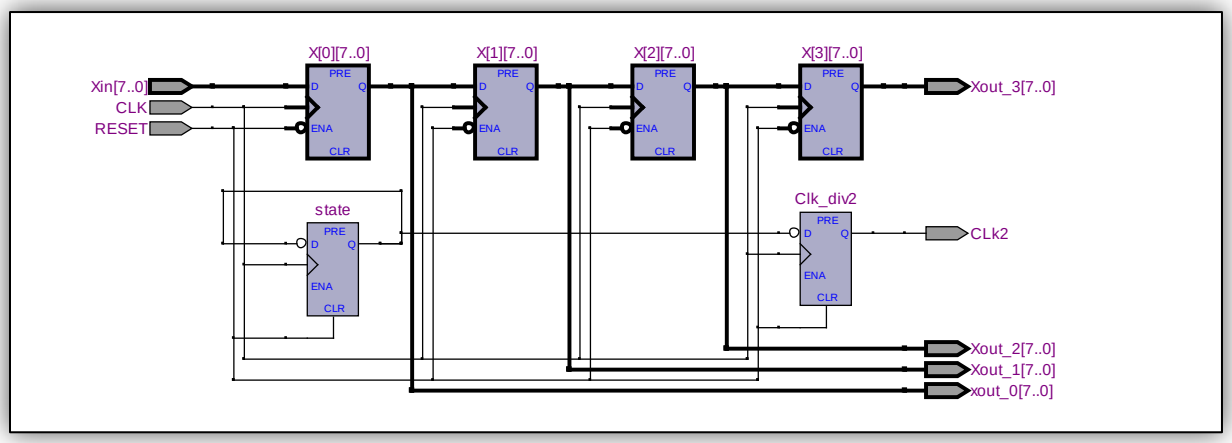


Figure 5.26. Vue interne (RTL) du diviseur.

La figure 5.27 donne la structure RTL de l'architecture des deux filtres (low-decomposition, high-decomposition). Comme on peut le remarquer les sorties du diviseur

attaquent directement ces deux parties. Chaque partie contient l'ensemble des multiplicateurs nécessaires pour l'ensemble des coefficients.

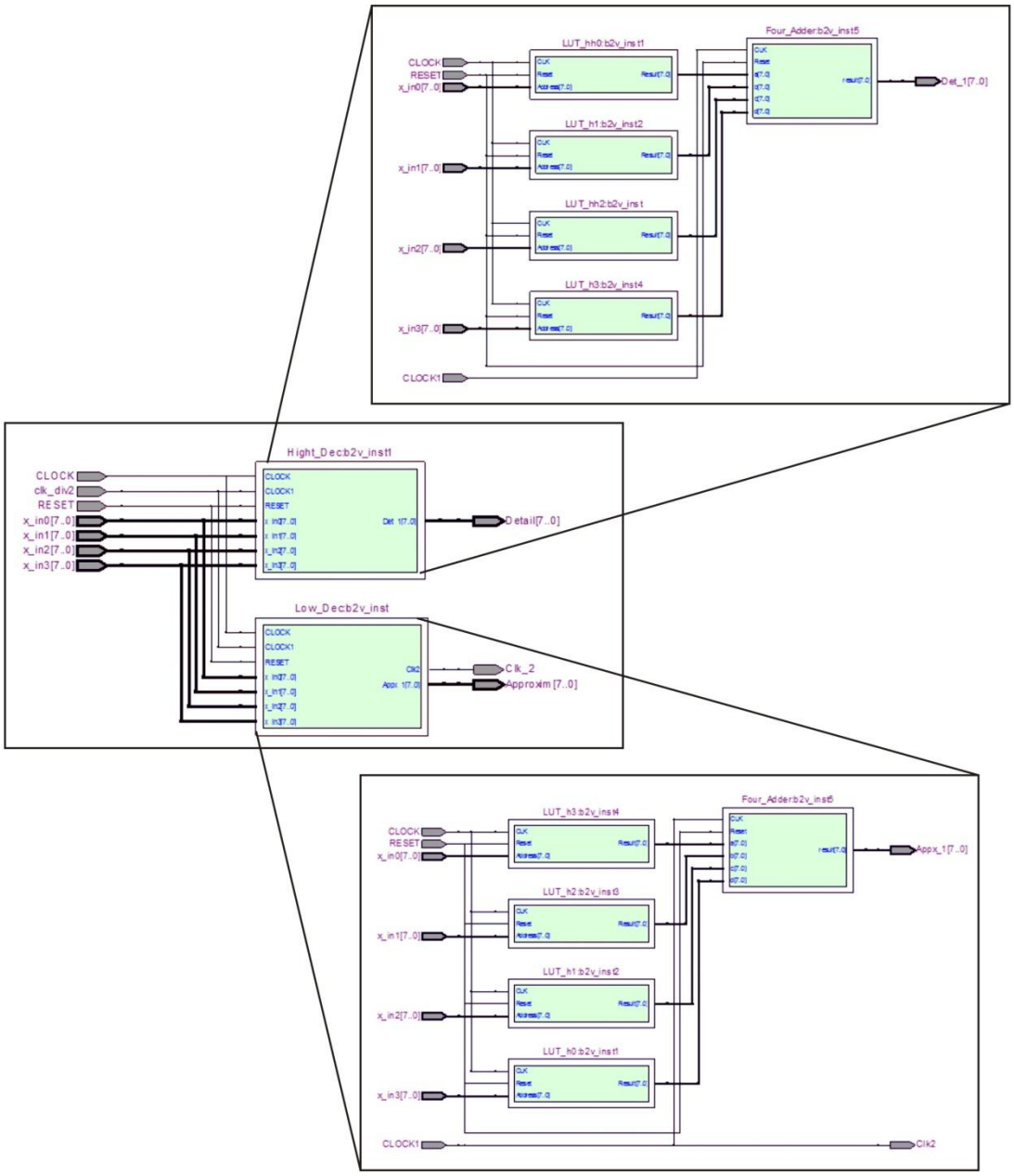


Figure 5.27. Vue interne (RTL) des deux filtres décomposition.

Le chronogramme de la figure 5.28 montre, notamment, la séquence des données (Data_in) appliquées à l'entrée de ce système et sa réponse comprenant les approximations (Appx) et les détails (Det) d'un telle niveau de décomposition. On peut voir, également, que la réponse se produit après un retard remarquable et significatif de 3.5 tops d'horloge (latency=3.5). Il est aussi claire que l'on a un échantillon chaque deux tops d'horloge due à l'effet du sous échantillonnage.

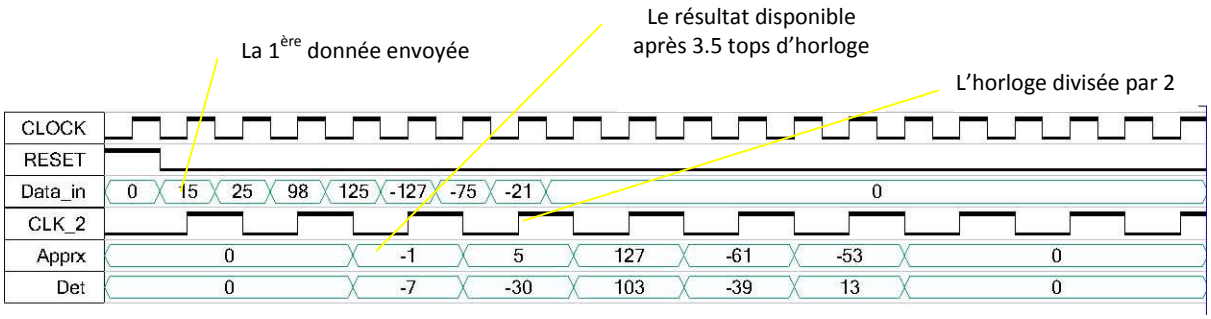


Figure 5.28. Le résultat de la simulation par Quartus ® II d'un niveau de décomposition.

Les valeurs énumérées dans le tableau 5.2 sont les résultats d'une simulation Matlab et les résultats de simulation par quartus. Il est clairement certifié la ressemblance des résultats obtenus par notre approche.

TABLEAU 5.2 : LE RESULTAT DE LA SIMULATION PAR MATLAB ET QUARTUS

	Simulation par Matlab		Valeurs à point fixe	
Data séquence	Approximation : A1	Détail : D1	A1	D1
15	-1.9411	-7.2444	-1	-7
25	5.4692	-29.7796	5	-30
98	138.5057	140.6995	127	127
125	-59.9604	-40.3064	-61	-39
-127	-53.7891	14.4127	-53	14
-75	0.0000	0.0000	0	0
-21	0.0000	0.0000	0	0

La simulation de la figure 5.29 est vue globale d'un niveau de décomposition. Le signal d'entrée (X_IN) est infecté par un bruit. Il est décomposé en deux signaux l'un est le signal d'approximation (Appro) qui représente la lire du signal d'entrée, et l'autre qui est le signal de détail (Detail). Cette simulation décrit le bon fonctionnement de l'architecture d'un niveau décomposition.

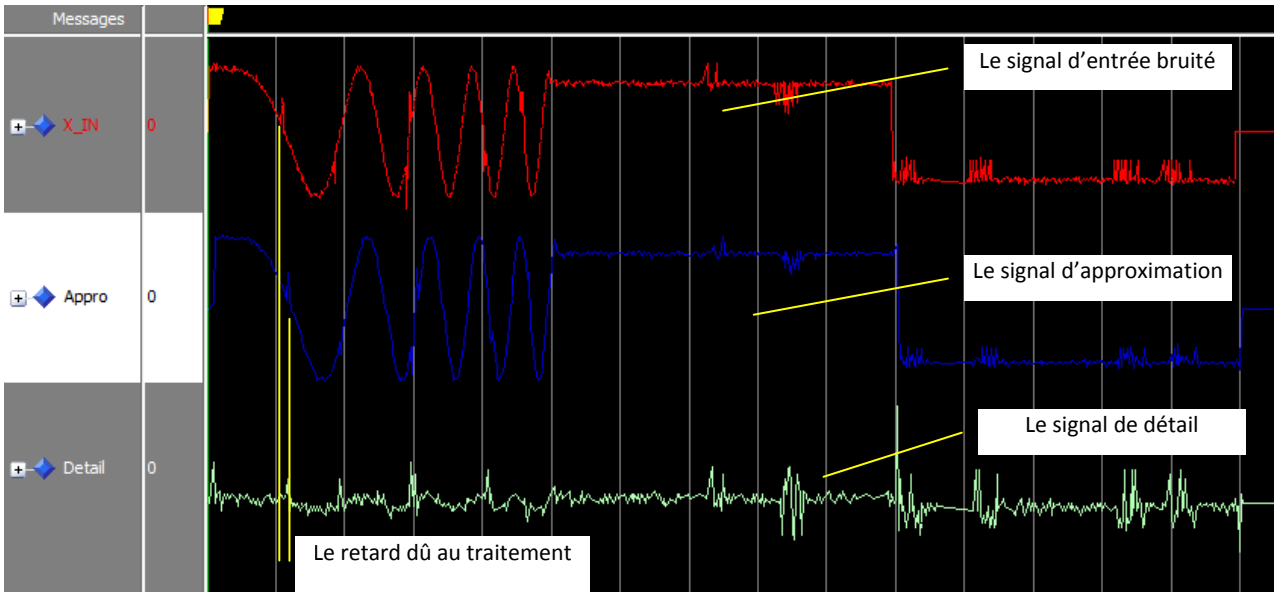


Figure 5.29. Simulation par ModelSim d'un niveau décomposition.

5.2.5.6 Le seuillage

L'architecture de la composante de seuillage doit réaliser les deux types de seuillage dur et doux (hard et soft) selon le choix pris par l'utilisateur suivant les équations suivantes :

$$D_{dur}(w,\lambda) = \begin{cases} 0 \; si \; |w| \leq \lambda \\ w \; sinon \end{cases} \qquad (6.20)$$

$$D_{doux}(w,\lambda) = sgn(w)(|w| - \lambda)_+ \qquad (6.21)$$

Afin d'avoir une architecture globale réduite de ce composant, nous avons cherché les points communs entre les deux types de seuillage. Ce qui nous a amené à une équation de seuillage soft en fonction du seuillage dure la suivante :

$$D_{doux}(w,\lambda) = \begin{cases} D_{dur}(w,\lambda) - \lambda \; si \; D_{dur}(w,\lambda) > 0 \\ D_{dur}(w,\lambda) + \lambda \; si \; D_{dur}(w,\lambda) < 0 \\ 0 \; otherwise \end{cases} \qquad (6.21)$$

D'après cette équation du seuillage soft, on peut facilement constater que le circuit de seuillage hard représente une partie du circuit de seuillage soft.

Quand à la valeur de seuil, elle doit être introduite aussi par l'utilisateur, pour ce faire nous allons allouer des lignes λ. Sur la figure 5.30 nous nous constatons l'architecture globale de la composante de seuillage.

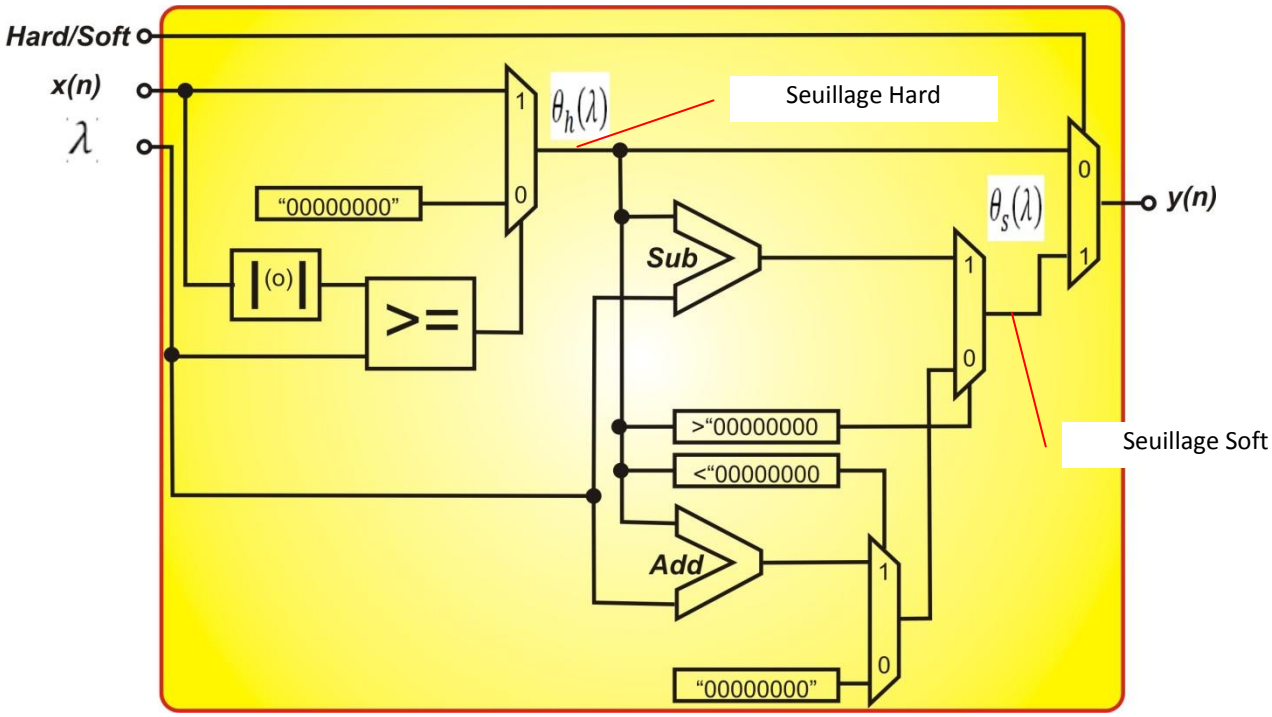


Figure 5.30. Architecture du composant de seuillage.

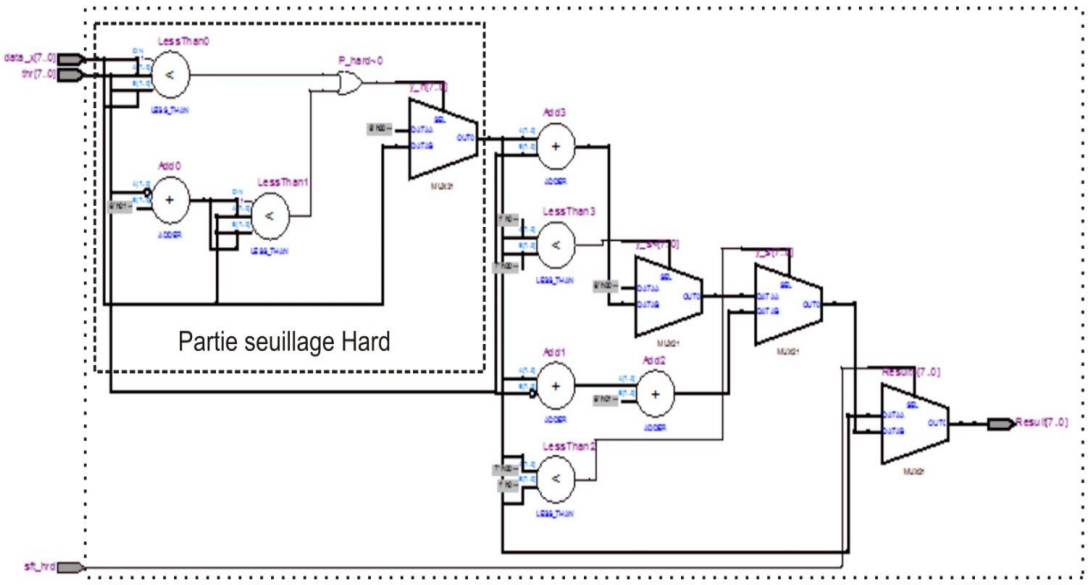


Figure 5.31. Vue interne (RTL) du composant de seuillage.

Dans la figure 5.32 on observe la simulation par Modelsim de ce composant. L'effet de seuillage Hard (Y_Hard) sur les échantillons du signal d'entrée (X_IN), où tous les échantillons inferieure de la valeur de seuille (seuille) égale à 43 sont mis à zéro. Par contre le reste des échantillons supérieur au seuille gardent leur amplitude. On observe aussi sur la même figure l'effet de seuillage Soft (Y_Soft) sur les mêmes échantillons du signal d'entrée

(X_IN) où tous les échantillons inferieure de la même valeur de seuille (seuille) seront mis à zéro. Par contre le reste des échantillons seront atténué par la valeur du seuille.

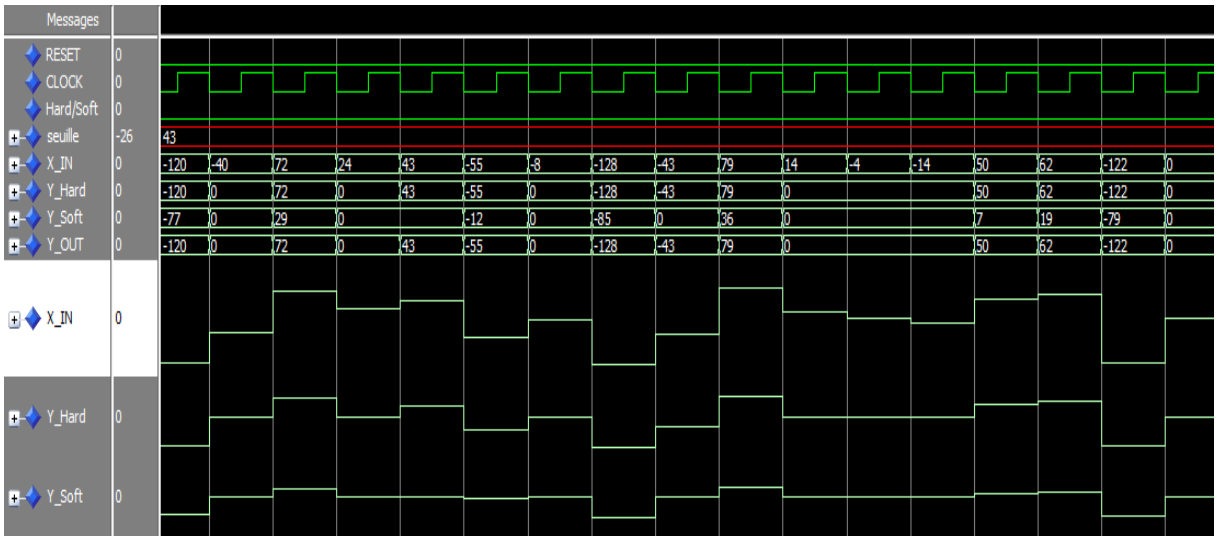


Figure 5.32. Simulation par ModelSim de l'architecture de seuillage.

Le tableau 5.2 illustre un exemple de calcule du seuillage Hard et Soft pour quelque échantillons d'entrée avec une valeur de seuille égale à 43 comme exemple.

TABLEAU 5.3 : RESULTAT DE CALCUL DU SEUILLAGE HARD ETSOFT (SEUILLE=43)

	Type de seuillage	
Data séquence	Hard	Soft
-120	-120	-77
-40	0	0
72	72	29
24	0	0
43	43	0
-55	-55	-12
-8	0	0
-128	-128	-85
-43	-43	0
79	79	36
14	0	0
-4	0	0
-14	0	0
50	50	7
62	62	19
-122	-122	-79

5.2.5.7 Un niveau de reconstitution

Dans un niveau de reconstitution de la phase de reconstitution. L'information résultante se compose de l'approximation *A[N]* et de détail *D[n]* . Les différentes parties de ce système approprié sont montrés sur la figure 5.33.

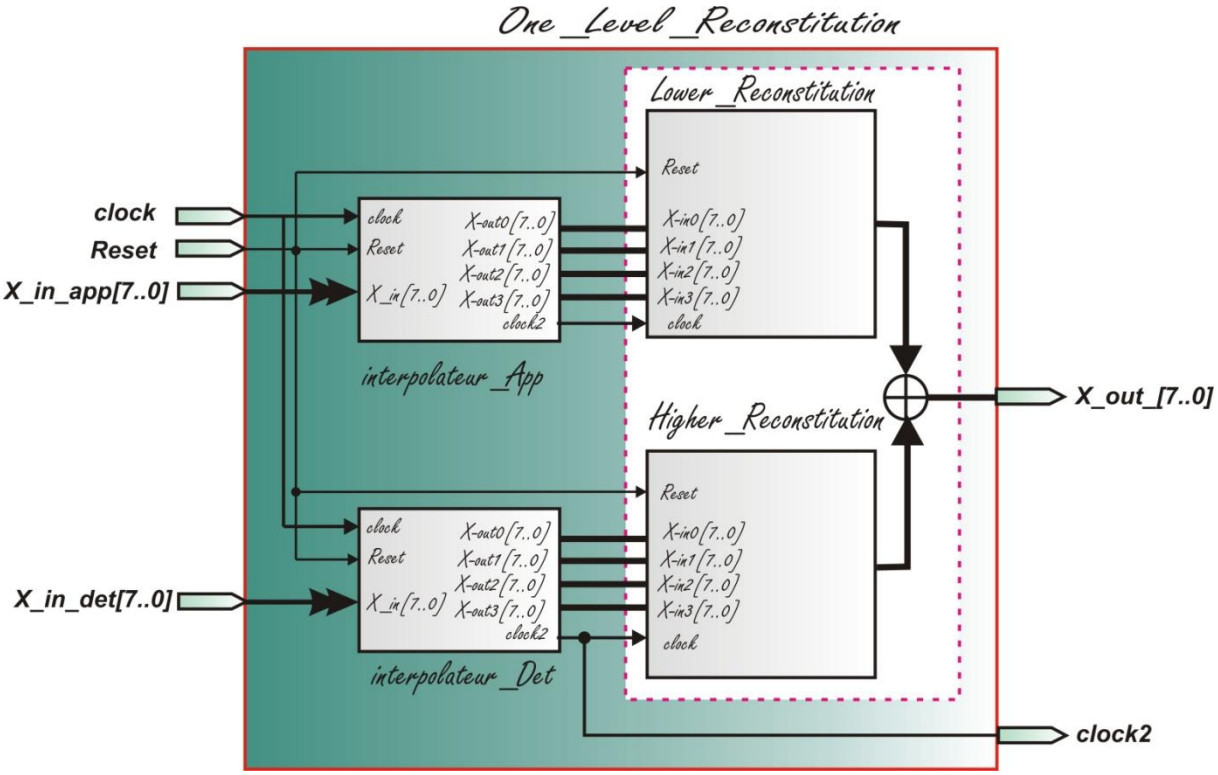


Figure 5.33. L'architecture d'un niveau de reconstitution.

Chaque filtre de reconstitution doit être précédé par une opération nécessaire c'est l'opération de l'interpolation (sur-échantillonnage). Elle est matérialisée par un dispositif qui consiste à insérer un zéro entre deux échantillons consécutifs qui peut être qualifié à un «bloc interpolateur". La structure de base d'un tel interpolateur est illustrée sur la figure .5.34. L'architecture comprend un doubleur de fréquence associé à un registre à décalage série/parallèle à 4 cases mémoire de 8-bits chacune, et un multiplexeur à 2 entrées qui sont le signal *x_in* et la valeur zéro.

Le multiplexeur joue le rôle d'un aiguilleur qui laisse passer dans la première demi période de l'horloge l'échantillon *x_in* dans la seconde période la valeur zéro pour assuré l'interpolation. La sortie du multiplexeur charge le registre à décalage, à la cadence de l'horloge doublé résultante du doubleur de fréquence. Les sorties parallèles de l'interpolateur, qui assure la causalité du filtre, sont préparés à être utiliser par les différents filtres de reconstitution apriori.

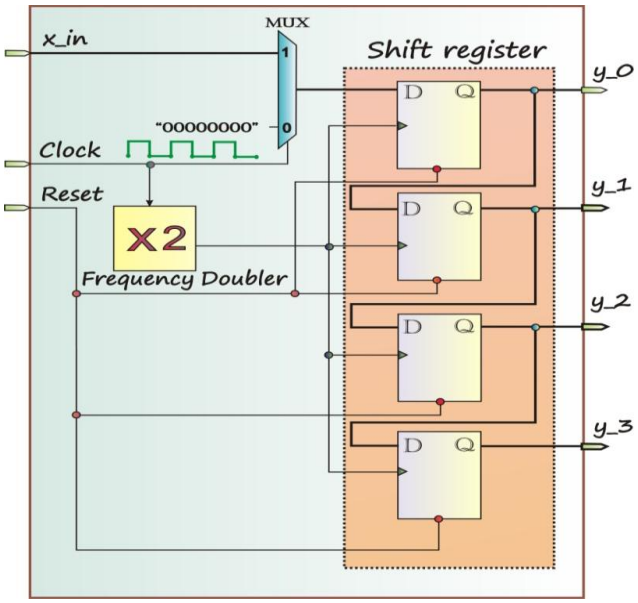


Figure 5.34. L'Architecture de l'interpolateur.

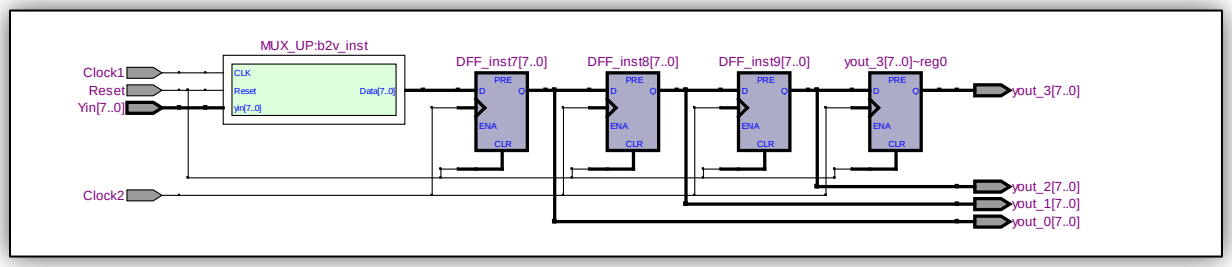


Figure 5.35. Vue interne (RTL) de l'interpolateur.

Afin de clarifier la conception, la simulation suivante montrent le comportement correct de cette composante, il est clairement visible sur le chronogramme de la figure 5.36 l'insertion d'un zéro entre deux échantillons consécutifs du signal d'entrée (Yin). Avec l'augmentation de la fréquence de travail (clock2). La deuxième mission de ce composant la préparation des quatre échantillons (yout0, yout1, yout2 et yout3) pour pallier le problème de la causalité des filtres de reconstitutions.

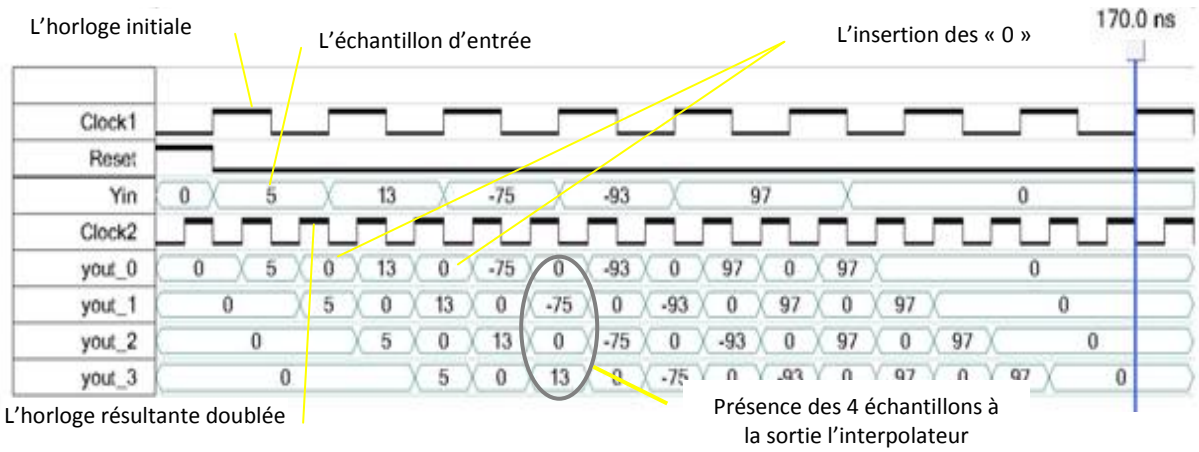


Figure 5.36. Digramme de la procédure d'interpolation.

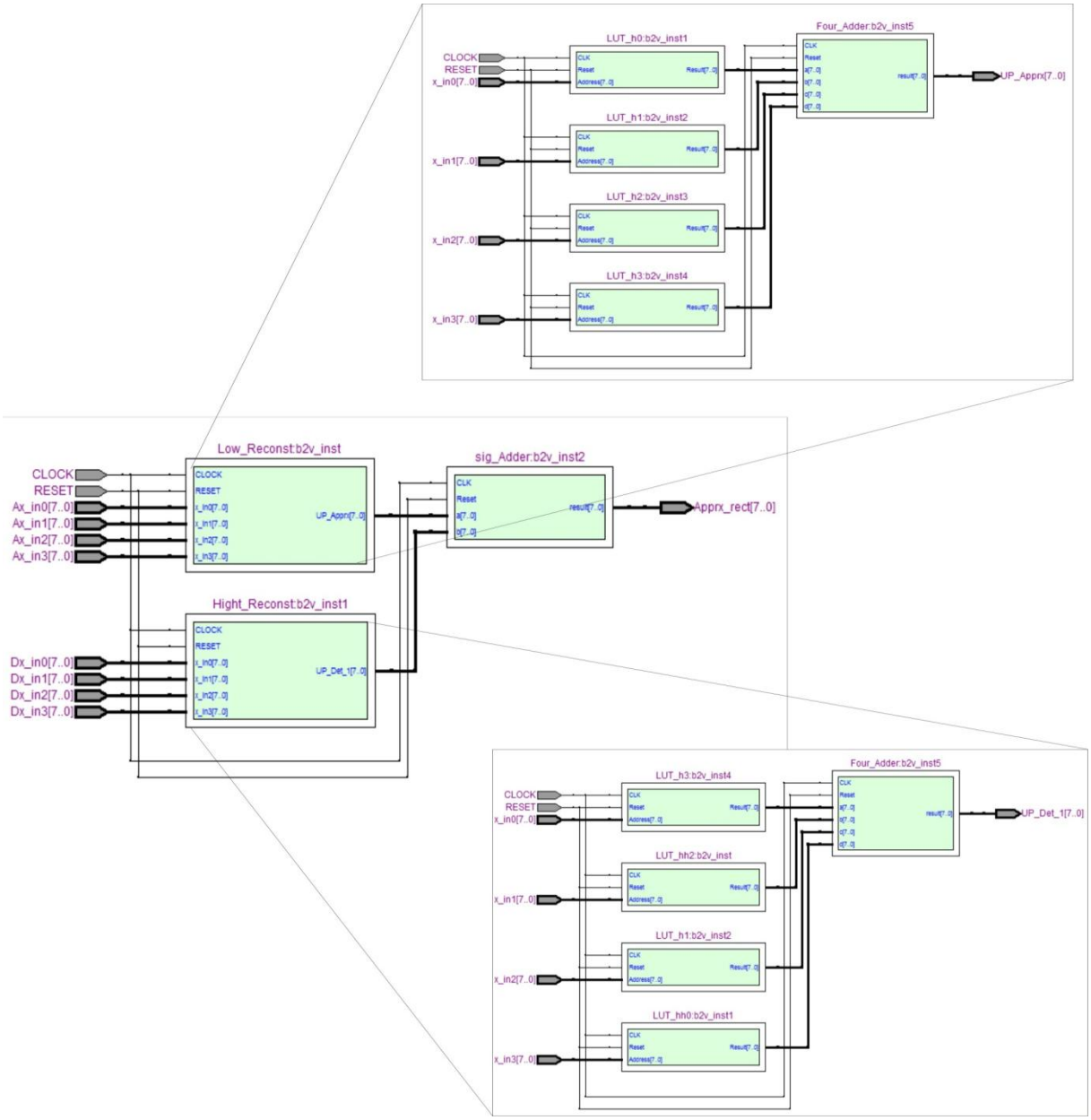


Figure 5.37. Vue interne (RTL) des deux filtres reconstitution.

La figure.5.38 décrit un exemple de simulation d'un niveau de reconstitution où on observe les signaux de détail et d'approximation (détail/appro, appro/inter) après l'opération d'interpolation qui consiste dans se cas à introduire un zéro entre chaque deux échantillons du signal d'entrée sans modifie la lire générale du signal. Les deux signaux (detail et Appro) représentent respectivement la sortie du filtre passe haut de reconstitution et la sortie du filtre passe bas de reconstitution. Le dernier signal y_out pour lui il représente la sortie d'un niveau de reconstitution qui est la somme des deux signaux (detail et Appro).

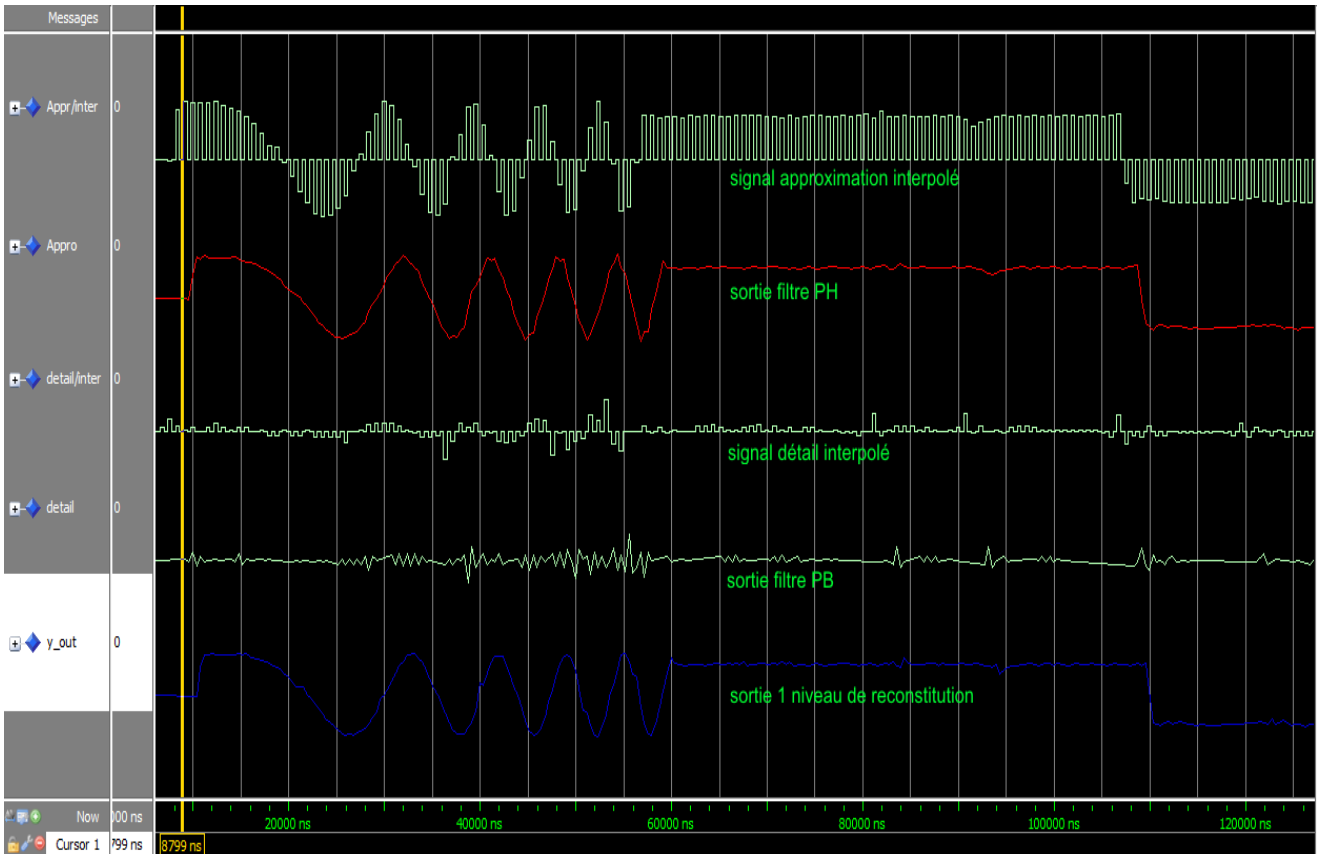


Figure 5.38. Simulation par ModelSim de l'architecture d'un niveau de reconstitution.

L'architecture globale de la phase de reconstitution donnée par la figure.5.39. L'information résultante se compose de la dernière approximation A3[N] et l'ensemble des détails D1[n], D2[n], D3[n] transportant le bruit, qui sont destinés à avoir subir un traitement efficace de débruitage en moyennant l'opération du seuillage.

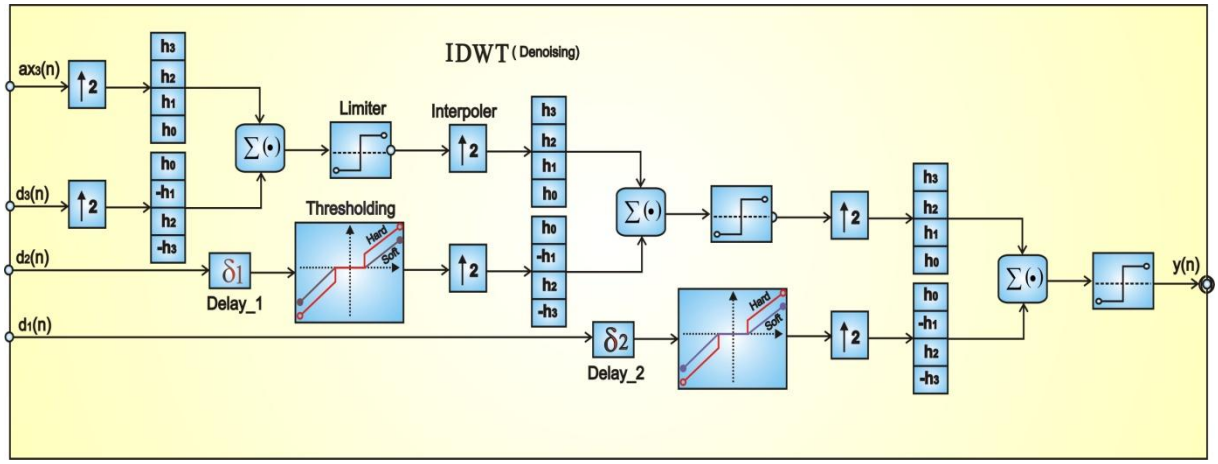


Figure 5.39. Schéma bloc de la phase de décomposition.

Dans l'architecture adoptée nous pouvons aisément constater la présence d'une série de blocs qui sont nécessaires et indispensables pour le bon fonctionnement de la structure

d'implémentation. Ce sont, en particulier, les blocs de retard (δ1 et δ2) et des dispositifs de quantification avec un détecteur dépassement underflow/overflow.

A chaque fois qu'un signal parcours un dispositif il subie un retard due au temps de calcule nécessaire. Et comme les coefficients $D_1[n]$ et $D_2[n]$ ont subie le moins de traitement que les coefficient $D_3[n]$ et $A_3[n]$. Donc ils doivent avoir des blocs de retard pour les synchroniser avec les autres coefficients.

δ1: est l'ensemble des temps de traitement d'un niveau de décomposition et d'un niveau de synthèse moins le temps alloué à l'opération de seuillage.

δ2: désigne le temps perdu pendant deux niveaux de reconstruction et deux niveaux de décomposition et moins le temps alloué à l'opération de seuillage.

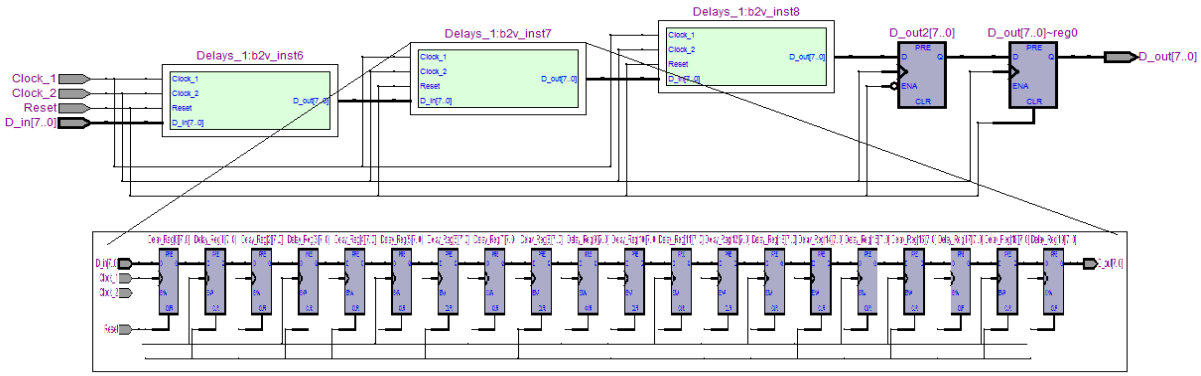


Figure 5.40. Vue interne (RTL) d'un exemple de bloc de retard.

La taille des données de toutes les sorties des circuits arithmétique dans ce design doit être la même que la taille des entrées afin de simplifie la synthèse. Qui est de 8 bits. Donc chaque bloc doit être suivi par un dispositif de limitation, qui est situé dans l'intervalle de -128 à 127. Le rôle de se dispositif est d'éliminer tous excédent de cet intervalle.

5.2.5.8 Filtre moyenneur

A la sortie du bloc de synthèse le signal de sortie global reste un peu infecté par un bruit impulsif. Implémenter un filtre moyenneur dont la mission principale est d'améliorer la qualité du signal restauré est une solution adoptée. En effet, Pour bien renforcer l'effet de débruitage on applique sur le signal de sortie un lissage. C'est donc, quelque part, faire un moyennage du signal. On comprend alors qu'il suffit de faire une somme pondérée de quatre échantillons successifs. Une structure est donnée ci-après. En effet, lorsque chaque nouvelle

donnée est reçue, la sortie est la moyenne des quatre données les plus récemment reçus. Si nous avions un flux d'entrée de données (x) indexées par le temps (par exemple xin est la valeur de xin au moment k) et nous avons reçu un nouveau échantillon de données dans chaque cycle d'horloge, l'équation de la sortie serait:

$$y_{out}(k) = [x_{in}(k) + x_{in}(k-1) + x_{in}(k-2) + x_{in}(k-3)]/_4 \qquad (5.21)$$

L'architecture de la structure du filtre moyenneur est illustrée sur la figure 5.41.

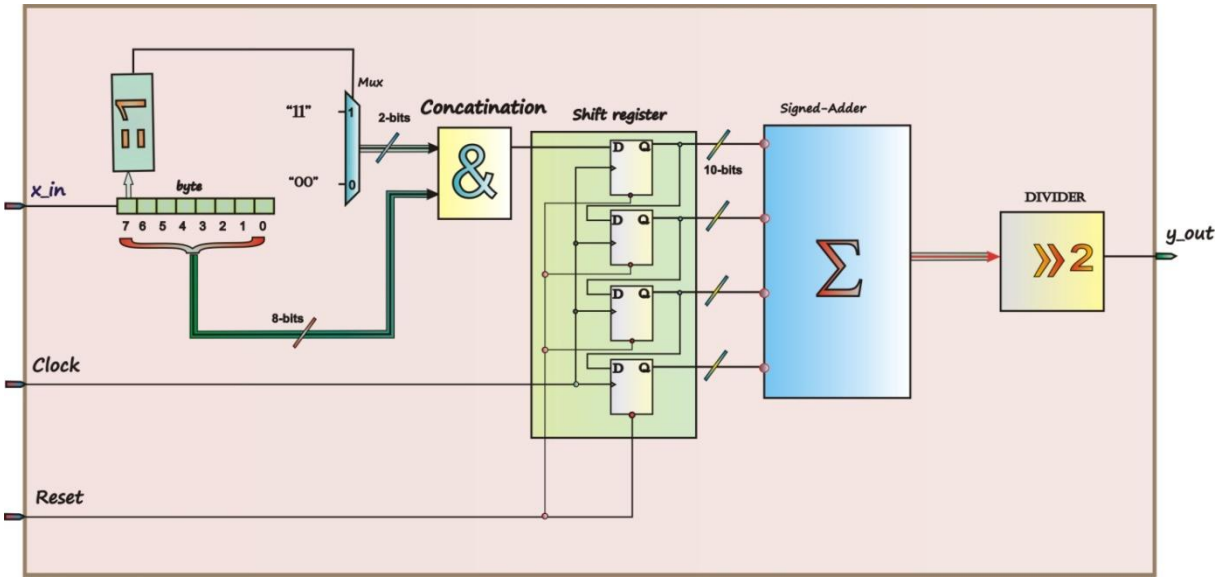


Figure 5.41. L'Architecture du filtre moyenneur.

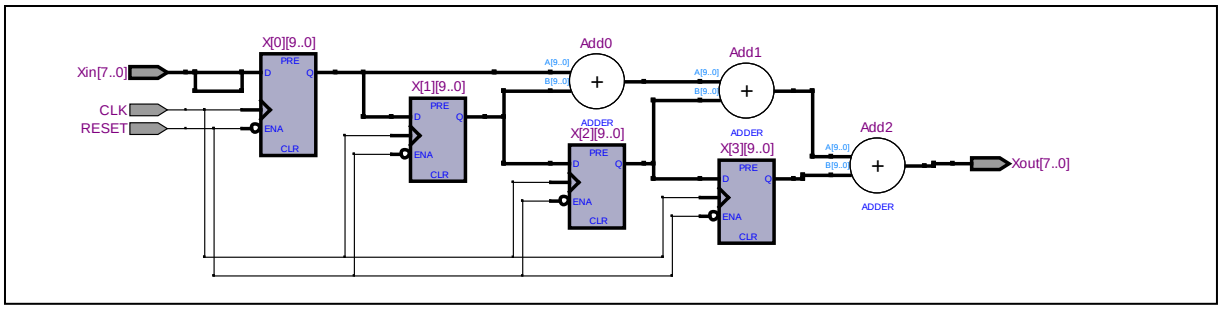


Figure 5.42. Vue interne (RTL) du filtre moyenneur.

Le multiplexeur *MUX* et le premier additionneur allongent la taille de l'échantillon x_in de 8 à 10 bits par la propagation du bit de signe, de façon que si le 8^{iém} bit est égal à un le 9^{iém} et 10^{iém} seront aussi par contre si le 8^{iém} bit est égal à zéro de même les bits 9 et 10. La sortie du premier sommateur charge le registre à décalage série/parallèle à la cadence du signal d'horloge. Les sorties parallèles attaquent l'additionneur signé (*signed-Adder*). Le résultat est sur 10-bits ce qui nécessite sa normalisation à la taille de 8-bits, c'est pourquoi on le décale à droite par deux bits cette opération représente une division par quatre. La

simulation de figure 5.43 à l'aide du Modelsim. Le signal d'entrée (a) est pollué par un bruit par contre le signal de sortie (b) de ce filtre est filtré. Donc cette simulation montre bien l'effet de ce type de filtre.

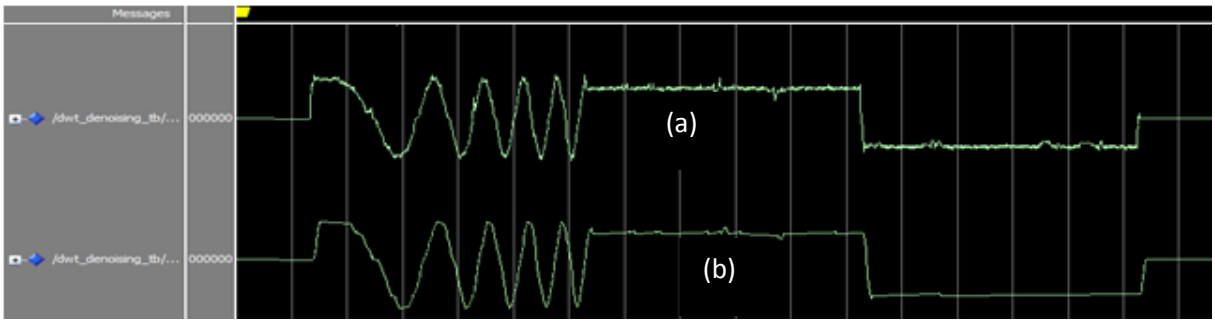


Figure 5.43. Simulation par ModelSim du filtre.

5.2.6 SIMULATION

La simulation a été effectuée dans un environnement MATLAB^{TM} version 7.0.0. La programmation est très simple. Plusieurs fonctions sont implémentées, et la manipulation graphique se distingue par sa facilité et sa portabilité.

Comme notre souci principal doit présenter la tâche d'exécution de notre algorithme d'une façon explicite, concise et efficace en dépit des limitations. Ainsi, pendant la phase de développement, quelques précautions ont été prises pour aboutir à notre but et il semble utile de se rappeler quelques considérations importantes.

Pour tester le bon fonctionnement des différents blocs écrits en VHDL, nous avons utilisé le logiciel ModelSim ALTERA WEB EDITION 6.3g_p1 où nous avons crée un ensemble de banc d'essai (Test Bench). c.a.d, pour chaque composant un Test Bench, et un Test Bench pour le composant global. Les mêmes données de Matlab ont été utilisées pour stimuler notre conception pour vérifier surtout l'effet de la troncature. Cette troncature est due à l'utilisation des données sur 8 bits Ainsi la sortie a aussi la même taille. Pour clarifier notre conception, a chaque étape, on effectuera les comparaisons des résultats qui ont été commentés pour déduire les conclusions appropriées.

La fiche de simulation de la figure 5.44, est une vue globale de simulation de toutes les phases de la procédure de débruitage. Nous avons stimulé notre désigne par une composante (a) d'un mélange caractérisée par l'association des trois différents segments entachés par un bruit occasionné par des variations intempestives et indésirables pour augmenter le degré de

sévérité en ce qui concerne cette méthode. Le signal (b) représente le signal d'entrée retardé par un temps égale au temps d'exécution globale du désigne. Afin de bien clarifier les déférences et effectué les comparaisons. D'après la simulation, on voie que le signal de sortie (c) est fournit après un retard remarquable et il reste un peut polluer. Comme le montre la figure le signal (d), qui est le résultat global de la procédure de débruitage, est bien filtrer.

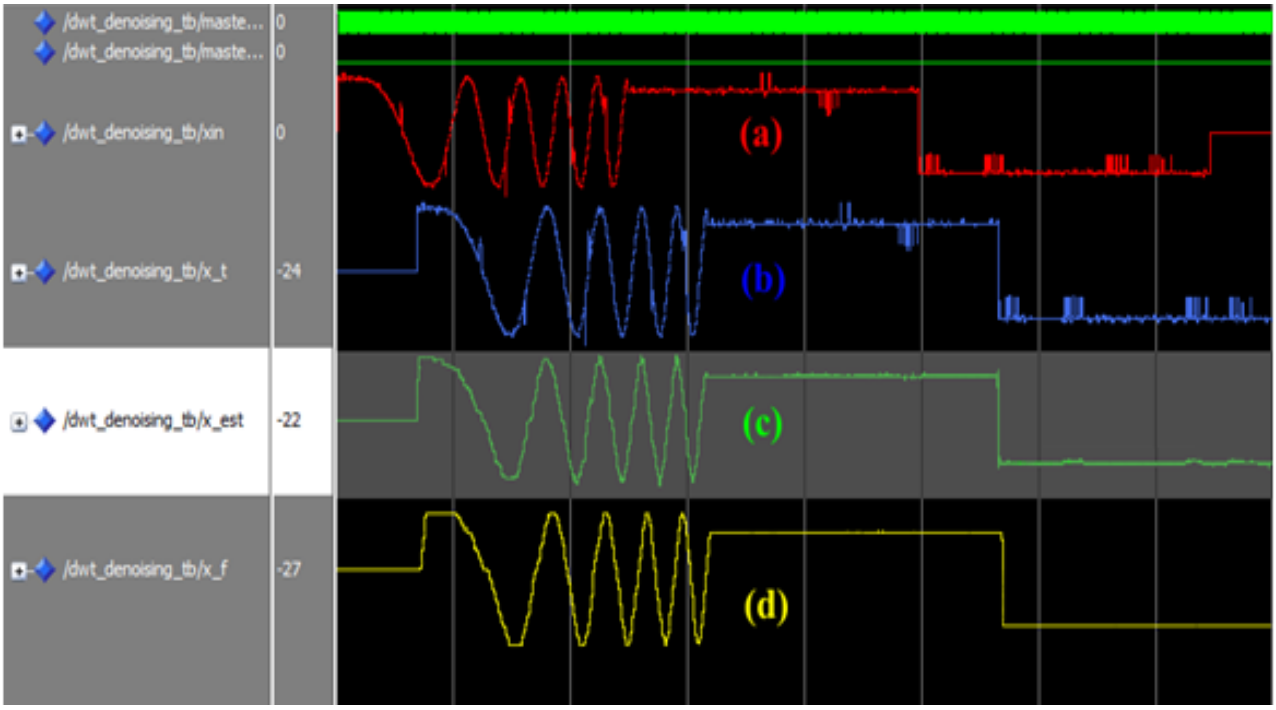


Figure 5.44. Le résultat de la simulation avec ModelSim; (a) signal original ; (b) signal original retardé; (c) signal débruité; (d) signal débruité et filtré.

5.2.7 Implémentation hardware

Nous avons implémenté notre conception en utilisant la carte DE2, cette carte de développement contient le circuit Cyclone ® II FPGA 2C35 d'Altera, EP1C6Q240. Ce circuit contient 33216 éléments logiques comme il est cité précédemment. Le résumé du résultat de la consommation des ressources matérielles pour notre conception est donné par le tableau 5.4.

TABLEAU 5.4. CONSOMMATION DES RESSOURCES MATÉRIELLES

	Utilisées	Disponible	Pourcentage

			d'utilisation
Logic elements	821	33216	2%
Combinational functions	802	33216	2%
Dedicated logic registers	148	33216	<1%
PLLs	1	4	25%

5.3 Conclusion

Ce travail a pour vocation de proposer une implémentation efficace de la procédure de débruitage du signal en utilisant la transformée en ondelettes. La conception proposée est testée et confirmée en utilisant des outils offerts par MATLAB. En outre le logiciel ModelSim ALTERA WEB EDITION 6.3g_p1, à l'aide d'un ensemble de banc de teste, utilisé pour vérifier le bon comportement de l'ensemble des programme VHDL développés. Ensuite nous avons procédé à l'implémentation avec succès en utilisant une cible programmable de la technologie FPGA. L'utilisation de la carte DE2 pour cette application a été présentée. Notre architecture est acceptable en termes de vitesse de fonctionnement. En outre, les ressources utilisées sont relativement faibles. Enfin, tous les résultats obtenus révèlent l'efficacité de notre conception.

Conclusion & Perspectives

CONCLUSION

Ce rapport de thèse a présenté notre travail de recherche. Nous le concluons en résumant les résultats principaux et les apports avant de proposer quelques perspectives.

L'implémentation numérique, en temps réel, d'une technique de débruitage d'un signal en ayant recours à la DWT est au centre des préoccupations de nombreux scientifiques. En effet, la DWT donne une représentation efficace d'un tel signal que l'on peut décrire par un nombre assez important de coefficients d'ondelettes. Donc La transformée en ondelettes discrète décompose le signal entrant en approximations (basse fréquence) et en détails (haute fréquence). Et comme le bruit est généralement de haute fréquence. Il devient donc nécessaire d'appliquer une opération de seuillage sur les coefficients de détails. A pour but d'éliminer le bruit. Lorsque l'opération de seuillage est terminée, la transformée inverse reconstitue le signal débruité.

En raison de l'orthogonalité de la DWT, la présence du bruit nous assure, en valeur moyenne, la même contribution sur tous les coefficients de cette transformée. Lors de la phase de l'implémentation sur FPGA, de la technique adoptée dans ce contexte là, un choix judicieux des algorithmes d'optimisations, tant sur le plan des éléments logiques internes nécessaires à la synthèse de la description comportementale des composants décrivant l'architecture de cette technique que sur la contrainte du temps réel, en vue de satisfaire à notre attente qui consiste en la qualité du signal à restituer, sans pourtant, nuire à la robustesse de cette approche. En utilisant les LUTs, la fonction débruitage est implémentée en VHDL dans un design qui élimine la nécessité d'utiliser des multiplicateurs classiques relativement longs et donc, réduire le temps d'exécution c.a.d augmenter les performances en termes de fréquence de travail. Dans la suite de nos contributions, et comme les LUTs sont gourmands en terme des éléments logiques, nous avons étudié et implémenté nos multiplicateurs avec des LUTs en réduisant leurs tailles ce qui nous amène à bien réduire l'utilisation des ressources matérielles. Une autre contribution est le fusionnement des deux circuits de seuillages HARD et SOFT en un seul circuit nécessitant moins de ressources matérielles.

Enfin, pour valider les résultats obtenus plusieurs bancs de tests de simulations et de confrontations ont été faits en moyennant, entre autres, l'environnement de la programmation et la simulation Modelsim. Les performances ont été justifiées en termes de ressources matérielles utilisées, et du temps d'exécution.

Cependant, comme perspective de ce travail, il serait intéressant d'étudier l'implémentation de la transformée en ondelettes discrète appliquée au débruitage avec un plus grand nombre de bits pour une meilleure précision et pour améliorer encore les performances de l'algorithme de débruitage. Le seul inconvénient de cette conception avec l'augmentation des nombres de bits est la nécessité d'un plus grand espace pour les LUTs parce que les valeurs requises seront stockées sous une forme plus grande et le calcul devient long.

Notre perspective vise, entre autre, à optimiser d'avantage cette technique en faisant intervenir la notion de « lifting » ainsi que la structure appropriée des multiplieurs basées sur les mémoires ou « look up table » qui se montrent très efficaces, en termes, des éléments logiques et du temps d'exécution, pour rédimer le problème de redondance et par conséquent simplifier considérablement les calculs.

Dans l'optique d'améliorer la qualité de la technique de débruitage d'un signal pour qu'elle puisse être utilisée efficacement, il serait intéressant d'étudier et implémenter un seuillage avec seuils adaptatifs.

Nous souhaitons que cette étude va aider et inciter les chercheurs à focaliser leurs attentions sur l'implémentation des techniques de traitement du signal sur des cibles reconfigurables de types FPGAs débouchant sur des applications palpables.

Bibliographie

BIBLIOGRAPHIE

[1] D.L. Donoho, Denoising by soft-thresholding, IEEE Trans. Information Theory 41 3 1995 613–627.

[2] the fiald-programmable gate array (FPGA) : Expanding its Boundaries, InStat Market Research, Avril 2006.

[3] J-M DUTERTRE « Circuits Reconfigurables Robustes »Thèse de doctorat, Université de Montpellier II, Oct. 2002.

[4] S.Brown and J.Pose FPGA and CPLD Architectures. A tutorial IEEE Design & Test of computers Summer 1996.

[5] the Stratix III handbook. http://www.altera.com/literature/lit-stx3.jsp

[6] Xilinx Inc.,"Virtex-II ProTM Platform FPGAs: Functional Description," DS083-2(v3.0), December10, 2003.

[7] Xilinx. The programmable logic, data book. Xilinx Inc, USA. (1994)

[8] D.SMITH « HDL Chip Design : A pratical Guide for Designing, Synthesis & Simulating Asics & FPGAs using VHDL or Verilog » Doone Pubns . ISBN : 0965193438

[9] Project Veripage, retrieved from: http://www.angelfire.com/ca/verilog/history.html.

[10] Xilinx. Synthesis and simulation design guide. Xilinx Inc, USA. (1998)

[11] BDTI focus report : FPGAs for DSP, Deuxième Edition, BDTI Benchmarking, 2006

[12] FPGAs accelerate time to market for industrial designa », M thompson, EE Times, 2 juillet 2004

[13] O. Spaniol : Computer Arithmetic: Logic and Design (John Wiley & Sons, NewYork, 1981)

[14] I. Koren: Computer Arithmetic Algorithms (Prentice Hall ,Englewood Cliffs, NewJersey,1993)

[15] E. E. Swartzlander: Computer Arithmetic, Vol. I (Dowden, Hutchingon and Ross, Inc., Stroudsburg, Pennsylvania, 1980), also reprinted by IEEE Computer Society Press 1990

[16] E. Swartzlander: Computer Arithmetic, Vol. II (IEEE Computer Society Press, Stroudsburg, Pennsylvania, 1990)

[17] K. Hwang: Computer Arithmetic: Principles, Architecture and Design (John Wiley & Sons, NewYork, 1979)

[18] IEEE: "Standard for Binary Floating-Point Arithmetic, " IEEE Std 754-1985 pp. 1-14 (1985).

[19] IEEE: “A Proposed Standard for Binary Floating-Point Arithmetic,” IEEE Transactions on Computers 14 (12), 51-62 (1981). TaskP754.

[20] J. ROSE , A. EL GAMAL and A. SANGIOVANNI-VICENTILLI, "Architecture of Field-Programmable Gate Array" in Proceeding of the IEEE Vol. 81 N° 7 JUL. 93.].

[21] O.Spaniol: Computer Arithmetic: Logic and design (John Wiley & Sons, NewYork, 1981.

[22] D.Bull, D.Horrocks: “Reduced-Complexity Digital Filtering Structures using Primitive Operations,” Electronics Letters pp. 769-771 (1987)

[23] D. Bull, D. Horrocks: “Primitive operator digital filters,” IEE Proceedings-G 138, 401-411 (1991).

[24] A. Dempster, M. Macleod: “Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters,” IEEE Transactions on Circuits and Systems II 42, 569-577 (1995).

[25] A. Dempster, M. Macleod:” Comments on “Minimum Number of Adders for Implementing a Multiplier and Its Application to the Design of Multiplierless Digital Filtersfl, “ IEEE Transactions on Circuits and Systems II 45,242-243(1998).

[26] Abdelhakim SAHOUR, Mohamed Benouaret, FPGA Implementation of Daubeshies Polyphase-Decimator filter, International Journal of Computer Applications (0975 – 8887)Volume 7– No.10, October 2010.

[27] A. Croisier, D. Esteban, M. Levilion, V. Rizo: (1973), « Digital Filter for PCM Encoded Signals,” US patent no.3777130.

[28] A. Peled, B. Liu: “A New Realization of Digital Filters,” IEEE Transactions on Acoustics, Speech and Signal Processing 22 (6), 456-462(1974).

[29] K. Yiu: “On Sign-Bit Assignment for a Vector Multiplier,” Proceedings of the IEEE 64,372-373(1976).

[30] K. Kammeyer: “Quantization Error on the Distributed Arithmetic,” IEEE Transactions on Circuits and Systems 24 (12),681-689(1981).

[31] F. Taylor: “An Analysis of the Distributed-Arithmetic Digital Filter,” IEEE Transactions on Acoustics, Speech and Signal Processing 35 (5), 1165-1170 (1986)

[32] S. White: “Applications of Distributed Arithmetic to Digital Signal Process-ing: A Tutorial Review,” IEEE Transactions on Acoustics, Speech and Signal Processing Magazine, 419 (1989).

[33] K. Kammeyer: “Digital Filter Realization in Distributed Arithmetic,” in Proc. European Conf. on Circuit Theory and Design (1976), Genoa, Italy

[34] F.Taylor: Digital Filter Design Handbook (Marcel Dekker, NewYork, 1983)

[35] H. Nussbaumer: Fast Fourier Transform and Convolution Algorithms (Springer, Heidelberg, 1990).

[36] S. Mallat, "multirésolution approximations and wavelet orthonormal bases of l2(R)", Trans. Am. Math. Soc., Vol. 315, N°1, pp. 69-87, sep 1989

[37] S. Mallat, "multifrequency channel decomposition of images and wavelet models", IEEE Trans. On acoustic speech and signal Proc., Vol. 37, N°12, pp. 2091-2110

[38] S. Mallat, "A theory for multifresolution signal decomposition: the wavelet representation", IEEE, Pami, Vol. 11, N°7, pp. 674-693, july1989.

[39] I. Daubechies, "orthonormal bases of compactly supported wavelets", Com. On Pure Appe. Math., vol. 41, PP. 909-996, Nov. 1988.

[40] Meyer, Y., (1986), Ondelettes, fonctions splines et analyse graduées, lectures données à l'université de Torino, Italie.

[41] Meyer, Y., (1990), Ondelettes et opérateurs, ED. Herman, France.

[42] Goswami, J. C., and Chan, A . K., (1999), Fundamentals of wavelets: theory, algorithms, and applications, New York, N.Y. : J.Willey and Sons.

[43] birslawn, C M., (1995), Fingerprints go digital, Notices of the AMS, vol. 42, pp. 1278-1283.

[44] Usevitch, B. E., (2001), A Tutorial on modern lossy wavelet image compression : Fondations of JPEG 2000, IEEE Signal Processing Magazine, vol 18, n°8, pp. 22-35.

[45] Mitisi M., Y. Mitisi, (2003), G. Oppenheim et J. Poggi, « les ondelettes et leur applications », Paris : Hermès Sciences Publication.

[46] Johnstone, I. M., and Silverman, B.W., (1997), Wavelet threshold estimators for data with correlated noise, J. Roy. Statist. Soc. B, 59: 319-351.

[47] http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html ; the Wavelet Tutorial by Robi Polikar.

[48] Mallat, S. (1998), A wavelet tour of signal processing. New York: Academic press.

[49] Daubechies, I., (1992), Ten lectures on wavelets, Philadelphia, Pa.: Society for industrial and applied mathematics.

[50] Daubechies, I., (1988), Orthonormal bases of compactly supported wavelets, commun. Pure Appl. Math., 91, PP 909-996.

[51] Strang, G.? and Nguyen, T., (1996), wavelets and filter banks, Wellesley, Mass.: Wellesley-Cambridge Press.

[52] Nico M. Temme, (1997), Asymptotics and Numerics of Zeros of Polynomials that are related to Daubechies Wavelets,

[53] William H.Press, Brian P.Flannery, Saul A. Teukolsky, and William T. Vetterling. Numerical Recipes in C; The Art of scientific Computing; The second Edition. ISBN 0-521-43108-5. Cambridge

[54] Stéphane Mallat. Une exploration des signaux en ondelettes. ISBN2-7302-0733-3. Les Editions de l'Ecole Polytechnique, 2000.

[55] Frédéric Truchetet. Ondelettes pour le signal numérique. ISBN 2-86601-672-6. Hermes, 1998.

[56] P.P. Vaidyanathan, Multirate Systems and Filter Banks, P T R prentice Hall, Englewood Cliffs, NJ, 1993.

[57] R.D. Koipillai and P.P. Vaidyanathan, "Cosine-Modulated FIR Filter Banks Satisfying Perfect Reconstruction", IEEE transactions on Signal Processing, vol. 40, pp. 770-783, April 1992.

[58] T.Q. Nguyen and R.D. Koilpillai, "The Theory and Design of Arbitrary-Length Cosine-Modulated Filter Banks and Wavelets, Satisfying Perfect Reconstruction", IEEE Transactions on Signal Processing, vol. 44, pp. 473-483, March 1996.

[59] N.J. Fliege, Multirate Digital Signal Processing, Chichester, U.K.: Wiley, 1994.

[60] T. Karp and N.J. Fliege, "Modified DFT Filter Banks with Perfect Reconstruction", IEEE transactions on Circuits and Systems-II: Analog and digital signal processing, vol. 46, pp. 1404-1414, November1999.

[61] T. Karp and N. J. Fliege, "Computational efficiency realization of MDFT filter banks," in Proc. EURASIP European Signal Processing Conf., Trieste, Italy, Sept. 1996, PP. M83-1186.

[62] N.J. Fliege, "Modified DFT polyphase SBC filter banks with almost perfect reconstruction," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, vol. 2, pp. 149–152, April 1994.

[63] Mitisi M., Y. ITISI, (2003), G. Oppenheim et J. Poggi, « les ondelettes et leur applications », Paris :Hermès Sciences Publications.

[64] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. Biometrika, vol. 81: pp. 425–455, 1994.

[65] Sungwook Chang, Y. Kwon, Sung-il Yang, and I-Jae Kim, Speech Enhancement for non-stationnary noise environnement by adaptive wavelet packet, Proccedings of IEEE International Conference on Acoustics, Speech, end Signal processing (ICASSP '02), vol.1, pp. 561_564, 2002. Texas instruments, (1999),

[66] David L. Donoho. De-noising by soft-thresholding. IEEE Transactions on In-formation Theory, vol. 41 (3): pp. 613–627, May1995.
[67] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. Biometrika, vol. 81: pp. 425–455, 1994.
[68] Gerard Aranguren, Mariano Barron, Joseba, Arroyabe And Gonzalo Garcia-Carreira pipe-IEEE conference in fuzzy systems (FUZZ-IEEE), Volume: 2, p- 635- 640, Barcelona,
[69] Valentin Mure San, Dan and Xiaojun Wang rom VHDL to FPGA. A case study of a international conference of young lecturers and PHD students, pp. 83 - 90, Miskolc, Hungary, 11-17 august 1997.
[70] Oscar Montiel, Yazmin Maldonado, Roberto Sepulveda, and Oscar Castillo," Simple Tuned Fuzzy Controller Embedded into an FPGA", Annual Meeting of the North American, Fuzzy Information Processing Society, NAFIPS, ISBN: 978-1-4244-2351-4, p. 1-6, New York City, NY, 19-22 May 2008.
[71] Shabiul Islam, Nowshad Amin, M.S.Bhuyan, Mukter Zaman, Bakri Madon Fuzzy Temperature Controller for Industrial Application", WSEAS Transactions On Systems And Control, ISSN: 1991-8763, Issue 10, Volume 2, p. 484- 490, Sep. 17, 2007.Spain, 1997.
[72] Philip t. Vuong, asad m. Madni and jim b. 13100 telfair avenue, sylmar, california 91342 USA, 2006.
[73] Handous « introduction to FPGAs » June 2009.

[74] http://www.wolfsonmicro.com/products/codecs/WM8731
[75] M.Benouaret, A.Sahour, S.Harize, Real time implementation of a signal denoising approach based on eight-bits DWT.
[76] Mateos Albiach, “interfacing a processor core in FPGA to an audio system” . 2006
[77] S. Zhuang, J. Carlsson, W. Li, K. Palmkvist, L. Wanhammar, “GALS based approach to the implementation of the DWT filter bank,” Proc. 7th International Conf. on Signal Processing, ICSP’04, vol. 1, 2004, pp. 567 – 570
[78] Comparison between Haar and Daubechies Wavelet Transformions on FPGA Technology Mohamed I. Mahmoud, Moawad I. M. Dessouky, Salah Deyab, and Fatma H. Elfouly
[79] Gerard Aranguren, Mariano Barron, Joseba, Arroyabe And Gonzalo Garcia-Carreira pipe-IEEE conference in fuzzy systems (FUZZ-IEEE), Volume: 2, p- 635- 640, Barcelona,
[80] D. L. Donoho, “De-noising by soft-thresholding,” IEEE Trans on information theory, vol. 41, pp. 612-627, 1995.

[81] Nico M. Temme, (1997), Asymptotics and Numerics of Zeros of Polynomials that are related to Daubechies Wavelets,

[82] M.Nibouche, A.Bouridane and O.Nibouche "Rapid Prototyping Of Biorthogonal Discrete Wavelet Transforms on FPGAs" To appear in IEEE International Conference on Electronics, Circuits and Systems, Malta, Sep 2001.

[83] M. Nagabushanam, Design and Implementation of Parallel and Pipelined Distributive Arithmetic Based Discrete Wavelet Transform IP Core, European Journal of Scientific Research ISSN 1450-216X Vol.35 No.3 (2009), pp.378-392.

[84] P. Y Chen, "VLSI implementation for one-dimensional multilevel lifting-based wavelet transform," IEEE Trans. on Computers, vol. 53, no. 4, April 2004, pp. 386-398.

[85] J.Chilo, T. Lindblad, "Hardware implementation of 1D wavelet transform on an FPGA for infrasound signal classification," IEEE Trans. on Nuclear Science, vol. 55, Issue 1, Part 1, 2008, pp.9 – 13.