REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AKLI MOHAND OULHADJ-BOUIRA

Faculté des Sciences et Sciences Appliquées
Département Génie Electrique

Mémoire de fin d'étude

Présenté par :

MECHTA Amel
METIDJI Rabia

En vue de l'obtention du diplôme de **Master en :**

Filière : TELECOMMUNICATION
Option : Système de télécommunication

Thème :

Simulation d'une application de traitement de signal optimisé sur un circuit logique programmable

Devant le jury composé de :

Labandji mourad			Encadreur
Saidi mohamed	MAA à	UAMOB	Président de jury
Haroun ismail	MCB à	UAMOB	Jury
Medjdoub ismail	MAA à	UAMOB	Jury

Année Universitaire 2019/2020

Dédicaces

A nos chers parents,

A nos familles Metidji et Mechta,

Et à nos amis proches.

Remerciements

Avant tout nous remercions ALLAH le tout puissant pour nous avoir donné le courage, la santé, et la patience pour réaliser ce travail.

Au terme de ce travail, nous tenons à exprimer notre gratitude et nos respects envers Monsieur M. Labandji l'encadrant de notre thème, pour sa disponibilité et le temps qu'il a su nous consacrer, pour toutes ses idées et tous ses conseils sans lesquels ce travail de thèse n'aurait pas aboutir, pour son suivi au quotidien de l'avancée de notre travail et surtout pour ses encouragements.

Finalement, nous remercions nos familles pour le soutien qu'ils nous avons apportés tout au long de ce travail et plus généralement tout au long de nos études.

Liste des abréviations

ASIC Application Specific Integrated Circuit

Ath Adder Depth, nombre maximum des opérations d'additions série de l'entré à la sortie

Avg Average number of additions

DSP Digital-Signal-Processor/Processing

FA Full Adder

FPGA Field Programmable Gate Array

HDL Hardware Description Language

LTI Linear Time Invariant

MAC Multiply-And-Accumulate

MCM Multiple Constant Multiplication

MEMS Micro Electro Mechanical Systems

SCM Single Constant Multiplication

VHDL Very high speed integrated circuit Hardware Description Language

Hcub Cumulative Benefit Heuristic

NP-difficile Non-deterministic Polynomial-time, problème complet pour la classe NP

OM Odd Multiple

Résumé

Beaucoup d'applications dans le traitement de signal, d'image, des systèmes de contrôle… etc. basés sur les systèmes LTI, ces systèmes sont basés généralement sur des multiplications par des constantes SCM/MCM qui consomment une quantité importante des ressources.

L'objectif de ce travail est le développement d'une solution pour la résolution de ces systèmes avec une optimisation des ressources consommées, la technique basée sur l'arithmétique RADIX-2r qui a des hautes capacités en matière de performance, et réduction du nombre des additions, de plus l'utilisation de cette heuristique au niveau bit permet plus de réduction sur les additionneurs de 1 bit, la solution décrire une description VHDL robuste et optimisé.

La solution développée est une plateforme qui peut être utilisée par tout un système LTI, elle assuré la construction de ce système avec une surface réduite, consommation de puissance réduit, et une vitesse élevée.

Mots clé : RADIX 2r, SCM, MCM, LTI, filtre RIF.

Abstract

Many applications in signal processing, image processing, control systems, etc. based on LTI systems, these systems are generally based on multiplications by SCM / MCM constants that consume a significant amount of resources.

The objective of this work is the development of a solution for the resolution of these systems with an optimization of the resources consumed, the technique based on arithmetic RADIX-2r which has high performance capacities, and reducing the number of additions, moreover the use of this heuristic at the bit level allows more reduction on the 1-bit adders, the solution describing a robust and optimized VHDL description.

The solution developed is a platform that can be used by an entire LTI system, it ensured the construction of this system with a reduced surface area, reduced power consumption, and high speed.

Key words: RADIX 2r, SCM, MCM, LTI, FIR filter.

الملخص

العديد من التطبيقات في معالجة الاشارات ومعالجة الصور وأنظمة التحكم وما إلى نلك . تعتمد غالبا على عمليات الضرب في ثابت أو مجموعة من الثوابت ,التي تستهلك الكثير من الموارد .

والهدف من هذا العمل هو تطوير حل على مستوى البتات لهذه الانظمة مع الاستفادة المثلى من المواد المستهلكة , اعتمادا على تقنيةRADIX-2r التي تتميز بالاداء العالي والحد الادنى من عمليات الجمع

,اضافة إلى هذا على كل أنظمة, فهو يضمن انشاء نظام ذو مساحة منخفضة ,سرعة في الاداء ,مع استهلاك أقل للطاقة .

الكلمات المفتاحية :

RADIX 2r, SCM, MCM, LTI, FIR filter.

Liste des figures

Liste des tableaux

Sommaire

Chapitres II **Multiplication par une constante SCM/MCM et le recodage RADIX-2r**

Introduction générale

Introduction générale

Les systèmes linéaires invariants jouent un rôle important en mathématique et sont répandus dans un large éventail d'application telle que le traitement numérique de signal (DSP), le traitement d'image et le système cryptographique, etc. Un système linéaire invariant est un modèle mathématique basé sur des opérations linéaires telles que l'addition, la soustraction, la multiplication et la division (+,-, ×, /) [1].

La multiplication par des multiples constants est une opération arithmétique qui multiplie un ensemble des constantsC_0, C_1, C_2, C_3 avec la même variable X représenté en virgule fixe. Cette opération domine la complexité des nombreux systèmes linéaires invariants dans le temps (LTI) tels que les filtres RIF/RII, les contrôleurs LTI, etc. Pour une implémentation présente des caractéristiques de rendement élevé, de vitesse, de compacité et de faible consommation d'énergie, la partie MCM doit éviter l'utilisation de multiplicateurs coûteux, l'alternative hardware n'utilisera pas de multiplicateurs, c'est-à-dire n'utilisera que l'addition, la soustraction, et le décalage. Nous supposons que les opérations d'addition et de soustraction ont le même coût en surface et en vitesse, et le décalage est gratuit, car il peut être effectué sans aucune porte, c'est-à-dire que seul le câblage est utilisé. Par conséquent, le problème MCM est défini comme le processus de trouver le nombre minimum d'opérations d'additions/soustractions.

L'objectif de ce travail est le développement d'une heuristique appelée RADIX-2^r complètement prédictible et sous-linéaire pour le problème MCM au niveau bloc d'additionneurs, elle présente l'avantage majeur par rapport aux algorithmes existants de n'avoir aucune limite sur la complexité du système LTI. RADIX-2^r présente les meilleurs résultats en termes de vitesse, surface, puissance en particulier dans les blocs MCM de moyenne/haute complexité.

Mis à part l'introduction et conclusion générale, notre mémoire est structuré en trois chapitres :

- ⇨ Le premier chapitre sera consacré aux systèmes linéaires invariants, ses déférents types FIR et RII, ainsi que ses différentes structures.

- ⇨ Le deuxième chapitre est consacré à l'opération de multiplication par une constante. Il présente une formulation des problèmes SCM/MCM (Single Constant Multiplication / Multiple Constant Multiplication) suivi par la présentation d'une nouvelle heuristique appelée RADIX-2r.

- ⇨ Le dernier chapitre nous appliquons la nouvelle heuristique Radix-2r présentée dans le chapitre précédent pour les filtres FIR.

Chapitre I :

Les systèmes linéaires invariants

I.1. Introduction

Le filtrage est une opération courante et très importante en traitement du signal. Le filtrage s'applique à des signaux représentés sous forme analogique (filtres analogiques), ou sous forme numérique après échantillonnage du signal (filtres numériques).

Dans le cas numérique, le filtrage s'effectue par une succession d'opérations mathématiques sur un signal discret. Il peut être réalisé par des circuits intégrés, des processeurs programmables (microprocesseurs, microcontrôleurs, etc.), ou par logiciel dans un ordinateur.

II.2. Les systèmes LTI

Un système linéaire est un modèle mathématique basé sur des opérations linéaires. Une opération linéaire se conformer par deux propriétés, à savoir l'additivité et l'homogénéité. Étant donné deux vecteurs x et y et un scalaire c, ces Propriétés sont décrites comme suit :

- Additivité : $f(x+y) = f(x) + f\,y\,)$ II.1
- Homogénéité : $f(c\times x) = c \times f\,(x)$ II.2

I.2.1. Le rôle des systèmes linéaires invariants

Les systèmes linéaires ont des rôles comme des abstractions mathématiques pour des modèles de calcul dans un grand nombre d' applications, dont principalement : la théorie du contrôle automatique, traitement du signal et télécommunications.

I.2.2. Formulation des systèmes linéaires invariants

Un système LTI est formellement décrit comme un vecteur d'entrée *X* multiplié par une matrice de transformation *C* pour obtenir un vecteur de sortie *Y*. Où chaque valeur de *C* est une constante.

Le système LTI s'écrit comme suit :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \quad \text{I.3}$$

La matrice de transformation C est une matrice m×n, où $C_{i,j}$ représente la constante *(i, j)*. Un signal de sortie Y_i est le produit de la $i^{ème}$ ligne de la matrice de transformation c avec n échantillons de l'entrée *X* :

$$Y_i = \sum_{j=1}^{n} C_{ij} \times X_j \quad \text{I.4}$$

I.3. Le filtrage numérique

Il est difficile de donner une définition formelle de la notion de filtrage, L'ingénieur électronicien pense souvent a une modification des caractéristiques fréquentielles d'un signal donné d'entrée. D'un point de vue théorique, le domaine fréquentiel est couplé au domaine temporel, le filtrage modifie donc également la réponse dans ce dernier.

A une séquence d'échantillons d'un signal d'entrée a temps discret x(n), un filtre numérique défini par sa réponse impultionnelle h(n) ou par sa fonction de transfert en H(z), répond par une séquence d'échantillons d'un signal de sortie y(n) (figure I.1) [2].

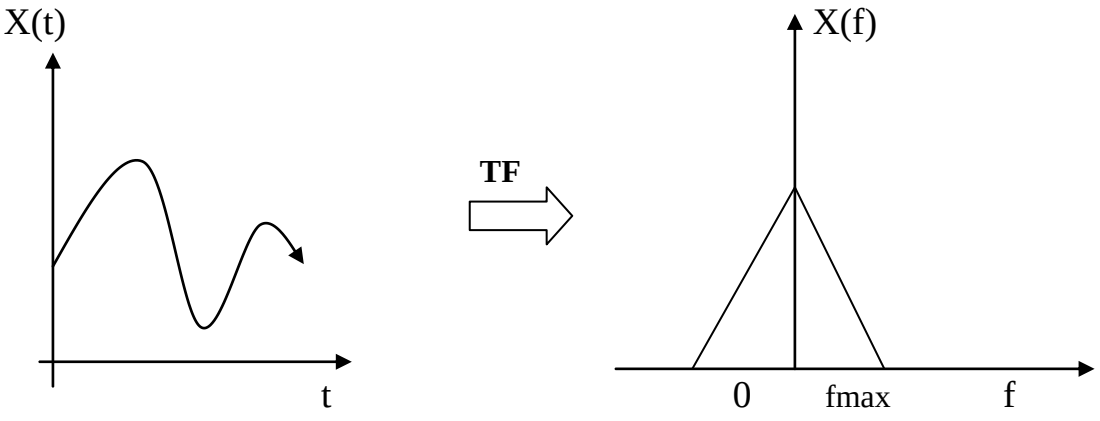


Figure I.1 : le domaine temporel et le domaine fréquentiel.

f_{max} : Fréquence maximale du spectre d'amplitude X(f) de X(t).

La fréquence f_{max} est obtenue par la conversion du signal x(t) du domaine temporel vers le domaine fréquentiel par la transformée de Fourier (figure I.2)

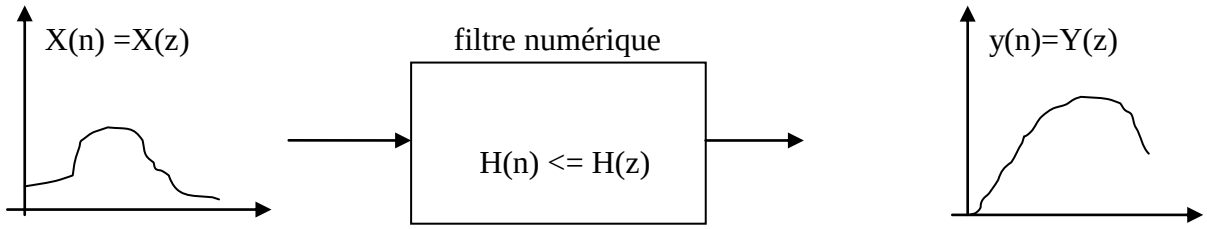


Figure I.2. Représentation sous forme de fonction de transfert en z.

Des exemples de filtrage sont donnés ci après.

- Réduction de bruit pour des signaux radio, des images issues de captures, ou encore des signaux audio.
- Modification de certaines zones de fréquence dans un signal audio ou sur une image.
- Limitation a une bande fréquentielle prédéfinie.

- Fonctions spéciales (dérivation, intégration, transformée de Hilbert…).

I.4. Représentation d'un filtre numérique

Un filtre numérique est un algorithme de calcul qui fait correspondre à une suite d'échantillons *X(n)* une autre suite d'échantillons. Dans le cas le plus général, l'échantillon *Y (n)* peut être écrit comme suit [1] :

$$Y(n) = b_0.x(n) + b_1.x(n-1) + \cdots + b_q.x(n-q) - a_1.y(n-1) + a_2.y(n-2) + \cdots + a_p.y(n-p) \quad \text{I.5}$$

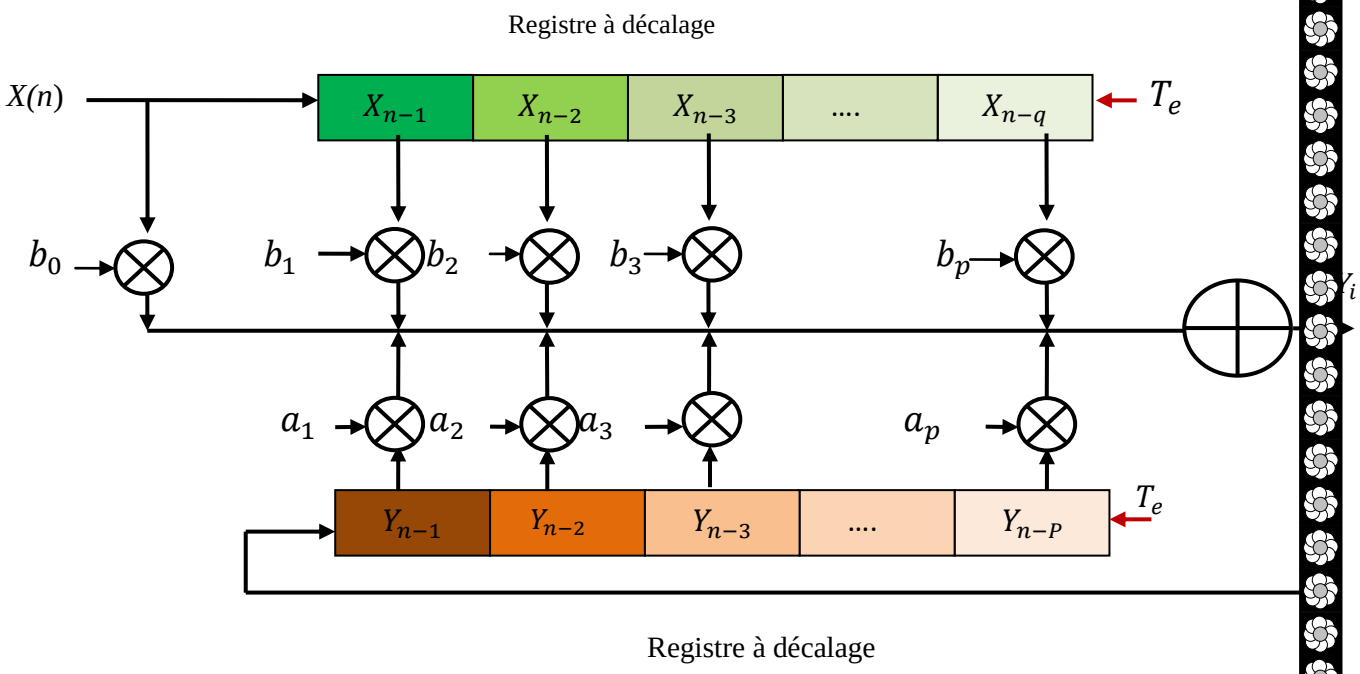


Figure I.3. La structure générale d'un filtre numérique.

Un filtrage numérique peut être représenté en utilisant plusieurs types de spécifications.

a. ***La fonction de transfert en z*** : ce code de représentation est le plus usuel. Il permet de lier l'entrée et la sortie dans le plan z par Y(z)=H(z).X(z). On posera dans la suite :

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{N} b_i.z^{-i}}{1+\sum_{I+1}^{N} a_i.z^{-i}} \qquad \text{I.6}$$

Où N(z) est le polynôme du numérateur de la fonction de transfert, tandis que D(z) son dénominateur. N est ici l'ordre de filtre. Dans le cas ou H(z) possède des poêles, on parlera de filtre RII. Si N(z) =1, On parlera de filtre tous-pôles. Dans le cas où D(z) =1, le filtres ne possède que de zéros. Cette famille de filtre correspond au cas des filtres RIF. Celle-ci n'a pas d'équivalent en filtrage analogique, et nous verrons que ses propriétés en font une fonction très utilisée en traitement numérique du signal.

b. ***Réponse impulsionnelle :*** La réponse impulsionnelle est la fonction en z inverse de H(z).

$$H(z) = \sum_{n=0}^{\infty} h(n).z^{-n} \qquad \text{I.7}$$

Comme en filtrage analogique, la sortie d'un filtre y(nT) est le résultat de la convolution du signal d'entrée représente de manière temporelle x(nT) avec la réponse impulsionnelle du filtre h(nT). On a alors y(nT) = x(nT) * h(nT), ou, si on fait abstraction de la période d'échantillonnage T :

$$y(n) = x(n) * h(n) = \sum_{k=0}^{\infty} x(k).h(n-k) = \sum_{k=0}^{\infty} x(n-k).h(k) \qquad \text{I.8}$$

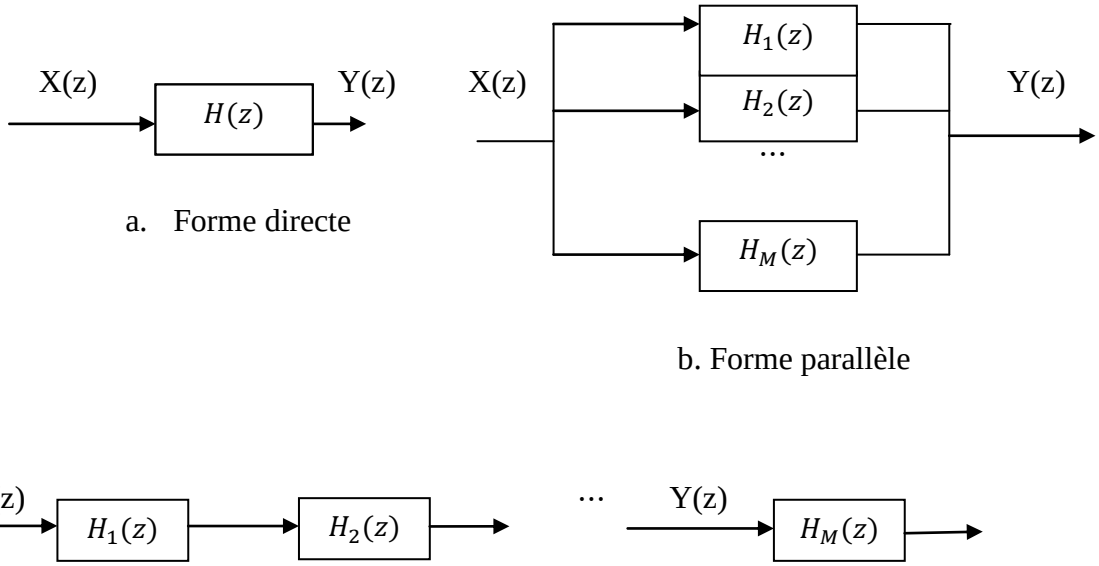
Dans le cas où x(n) est une impulsion δ(n), on retrouve bien y(n) = h(n).

Selon les cas où h(n) est à support infini ou fini, on retrouvera respectivement les deux types de filtres RII et RIF.

c. ***Équation aux différences :*** Une transformation en z inverse de l'´equation permet d'aboutir à la forme suivante :

$$y(n) = \sum_{i=0}^{N} b_i.x(n-i) - \sum_{i=0}^{N} a_i.y(n-i) \qquad \text{I.9}$$

On identifie ici deux parties distinctes : une partie fonction de la valeur courante et des valeurs précédentes de l'entrée x(n), et une partie fonction des valeurs précédentes de la sortie y(n). Selon si les a_i sont non nuls ou nuls, on parlera donc de filtres récursifs ou de filtres non récursifs.



a. Forme directe

b. Forme parallèle

c. Forme cascade

Figure I.4. Eprésentation sous la fonction de transfert en z.

I.5 : Spécification d'un filtre numérique

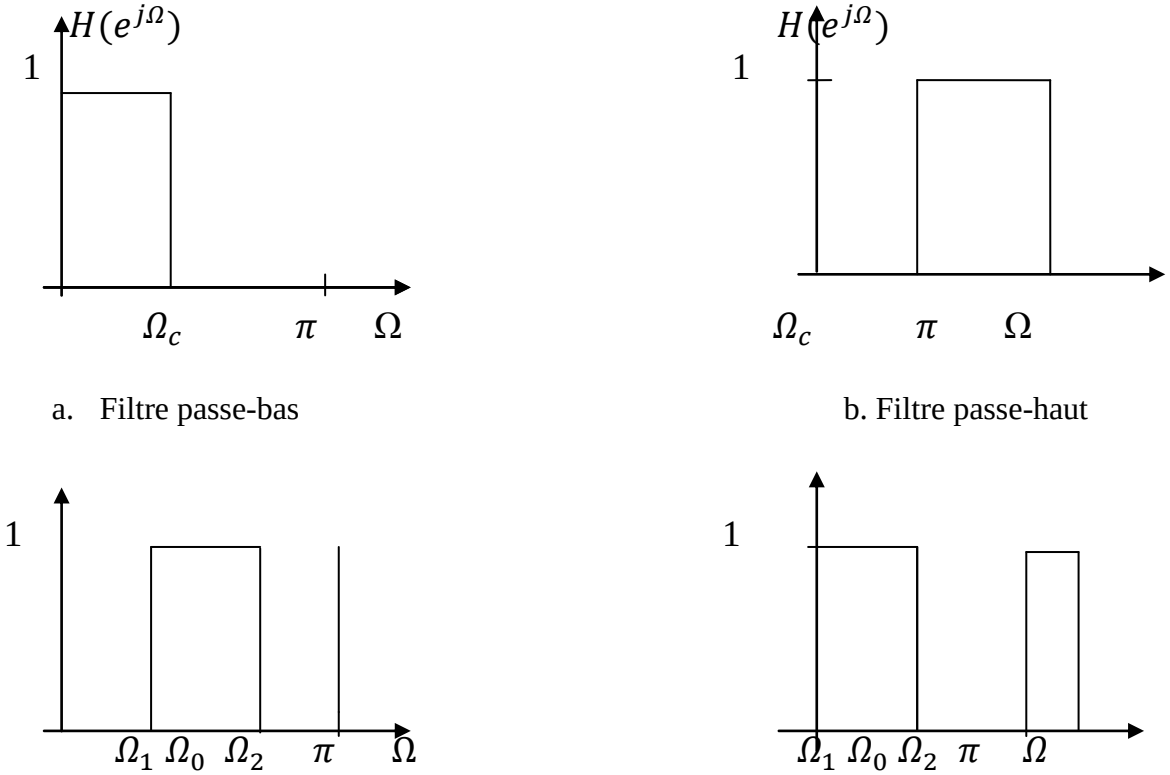
Avant qu'un filtre numérique soit conçu et implanté, nous avons besoin de définir ses spécifications. Un filtre doit laisser passer certaines fréquences, alors qu'il doit en atténuer d'autres. Nous devons donc pouvoir représenter ces contraintes. Il y a quatre filtres de bases :

- les filtres passe-bas laissent passer les fréquences inférieures à une fréquence de coupure f_c et bloquent celles qui lui sont supérieures (figure I.4.a),

- les filtres passe-haut bloquent les fréquences inférieures à une fréquence de coupure f_c et laissent passer celles qui lui sont supérieures (figure I.4.b),

- les filtres passe-bande laissent passer les fréquences autour d'une fréquence centrale f (ou comprises entre f_1 et f_2) et bloquent les autres (figure I.4.c),

- les filtres réjecteur-de-bande bloquent les fréquences autour d'une fréquence centrale f_c (ou comprises entre f_1 et f_2) et laissent passer les autres (figure I.4.d)[1].



a. Filtre passe-bas

b. Filtre passe-haut

c. Filtre passe-bande

d. Filtre réjecteur-de-bande

Figure I.5. Réponse fréquentielle idéales des 4 filtres de base.

Les filtres représentés en figure I.5 sont idéaux. Dans un cas réel il ne peut y avoir de discontinuités. Le passage entre zones passantes et zones atténuées se fait par des zones dites de transition dont la largeur va exprimer la sélectivité du filtre. Les bandes passantes et atténuées ne sont également pas idéales, elles contiennent des ondulations dont l'amplitude est exprimée par les paramètres d'ondulation en bande passante et d'atténuation. Pour toutes ces raisons, la spécification d'un filtre est habituellement réalisée à partir d'un gabarit fréquentiel, défini entre 0 et π.

I.6. Avantages et inconvénients des filtres numériques

- *Reproductibilité* : les caractéristiques de tous les filtres numériques établis sur une même configuration sont rigoureusement identiques,
- *Souplesse* : la réponse en fréquence peut être très aisément modifiée en changeant les coefficients arithmétiques ; le domaine des fréquences de travail est facilement déplacé par modification de la fréquence d'échantillonnage,
- *Précision* : les différentes manipulations étant effectuées sur des nombres, la précision ne dépend, en grande partie, que de celle du CAN et de celle du CNA,
- *Association des filtres* : la mise en série de filtres numériques ne pose aucun problème d'interaction, tel que celui que l'on rencontre pour l'adaptation des impédances des filtres analogiques,
- *Stabilité des caractéristiques* : il n'y a pas de vieillissement des composants dû à l'influence de la température sur les caractéristiques du filtre.

Les principaux inconvénients sont liés au problème de l'échantillonnage (spectre du signal toujours limité) nécessitant l'utilisation de processeurs ayant une bonne rapidité d'exécution pour pouvoir traiter des signaux ayant une forte dynamique (fréquences élevées) en temps réel[1].

I.7. Classification des filtres numériques

Les filtres numériques peuvent être classés selon plusieurs critères :

- La longueur de la réponse impulsionnelle implique deux types de filtres RII et RIF,

- Le type de représentation, ou de structure, implique deux types de filtres récursifs et non récursifs. Nous verrons qu'à l'exception d'un cas particulier, les filtres récursifs et non récursifs sont respectivement équivalents aux filtres RII et RIF.

I.7.1. les filtres à réponse impulssionnelle infinie (ou filtre RII)

Ils sont nommés ainsi parce que, leur réponse impulsionnelle est de durée théorique infinie.

$$h(n) \neq 0 pour\ \ n = 0,1,2, \ldots + \infty$$

Ce sont des filtres numériques tels que le signal de sortie dépend, à la fois, du signal d'entrée et des échantillons précédents de ce signal de sortie. Pour cette raison, ils sont aussi nommés filtres récursifs.

I.7.1.1.Les principales caractéristiques des filtres RII

Les principales caractéristiques des filtres RII sont :

- La relation entre le signal d'entrée et le signal de sortie :

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i) \qquad \text{I.10}$$

- La fonction de transfert :

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-1}}{1+\sum_{i=1}^{M} a_i z^{-1}} \qquad \text{I.11}$$

- Les RII peuvent être instables en réseau de leurs structures à base de pôles et de zéros :

$$H(z) = b_0 z^{M-N} \frac{\prod_{i=0}^{N}(z-z_i)}{\prod_{i=1}^{M}(z-p_i)} \qquad \text{I.12}$$

- Une plus grande sensibilité numérique (quantification des coefficients, bruits de calcul)

I.7. 2. Les filtres à réponse impulsionnelles finies (ou filtre RIF)

Ce sont des filtres numériques caractérisés par une réponse uniquement basée sur les valeurs du signal d'entrée. Du fait du nombre fini des échantillons de la réponse impulsionnelle, ces filtres sont toujours stables.

$$\begin{cases} h(n) = 0 & \text{pour } n \geq N \\ h(n) \neq 0 & \text{pour } n = 0,1,2,\cdots N-1 \end{cases} \quad \text{I.13}$$

I.7.2.1.Les principales caractéristiques des filtres RIF

- Une stabilité inhérente $\sum_{n=0}^{N-1}|\mathrm{h(n)}| < +\infty)$ I.14

- La relation entre le signal d'entrée et le signal de sortie est donné par :

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) \quad \text{I.15}$$

Où les $b_i$sont des constantes et N désigne la longueur de filtre.

- La fonction de transfert de filtre s'écrit :

$$H(z) = \sum_{i=0}^{N-1} b_i . z^{-i} \quad \text{I.16}$$

- Une phase linéaire.
- Une plus grande stabilité numérique que les RII.
- Une grande facilité d'implantation dans un système numérique de traitement.

I.7.2.2. Applications du monde réel sur filtres RIF

Quelques applications pour les filtres FIR sont énumérées ci-dessous:

- Annulation d'écho ;
- Télécommunications
- La communication de données
- Les communications sans fil

- DTV
- Le traitement de la vidéo
- La synthèse vocale
- Synthèse de forme d'onde
- ADSL

I.8. Structures des filtres numériques

On distingue deux types de réalisation de filtre numérique : la structure transversale et la structure récursive. Ces réalisations sont effectuées à partir de circuits numérique de base (sommateurs, multiplieurs,..).

I.8.1. Structure transversale

Cette structure est dite non récursive ou transversale car elle ne fait apparaitre aucun bouclage de la sortie sur l'entrée (la sortie ne dépond que des entrées aux instants précédents). Elle est associée exclusivement aux filtrez RIF. C'est essentiellement la partie numérateur de la fonction de transfert H(z).

L'équation aux déférences est donnée par :

$$y(n) = \sum_{i=0}^{N} b_i \,.\, x(n-i) = b_0.x(n) + b_1.x(n-1) + .. + b_{N-1}.x(n-N+1) + bn.x(n-N) \quad \text{I.17}$$

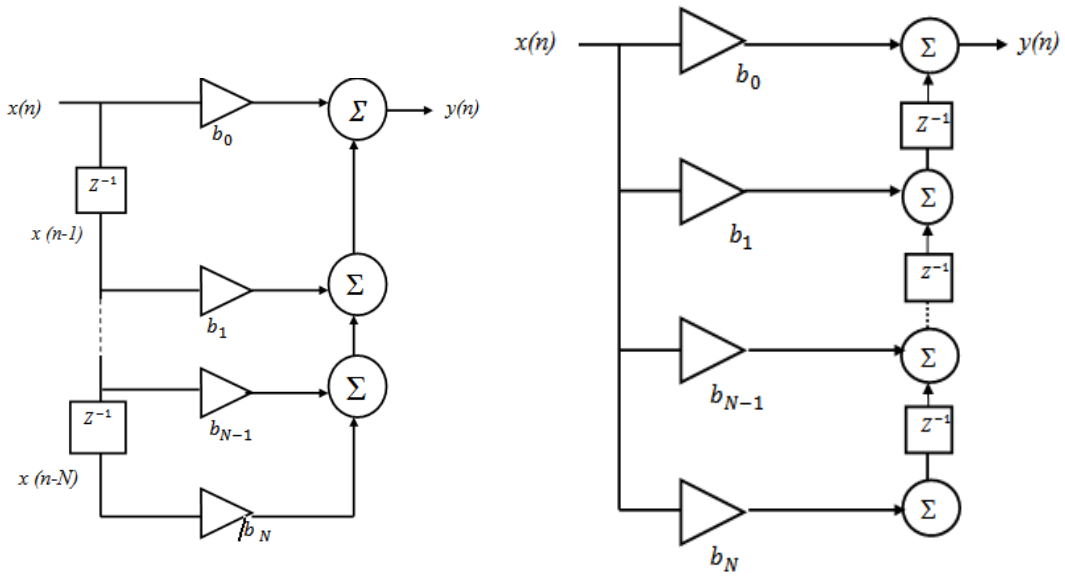
L'équation précédente représente le comportement temporel d'un filtre RIF. On peut en déduire immédiatement la structure directe d'un filtre RIF qui est représentée à la figure est obtenue après manipulation de cette équation.

On a donc, dans le domaine des z :

$$H(z) = \sum_{i=0}^{N} b_i . z^{-i} \quad \text{I.18}$$

Pour la réalisation de ce filtre il suffit d'effectuer un nombre fini de multiplications et d'additions. Le nombre de multiplications est égal à (N) et le nombre d'additions à (N-1).

Deux types de modèles sont possibles, dits structure directe et structure transposée, représentés par la figure I.6.



a. Structure directe b. structure transposée

Figure I.6. Structures de réalisation transversale

Un filtre RIF nécessite N + 1 opérations de multiplication, N opérations d'addition pour chaque nouvel échantillon à filtrer. On peut également exprimer la complexité en nombre de multiplication-accumulation (MAC), qui, dans le cas du filtre RIF, vaut N + 1. Le coût mémoire d'un filtrage RIF est de 2(N +1) (N +1 coefficients *bi* et N +1 points mémoire pour le vecteur des entrées x(i)). Si la fréquence d'échantillonnage du signal vaut Fe, cela signifie que le calcul d'un filtre devra être réalisé en un temps T calcul inférieur à Te = 1 / Fe.

Sur un processeur de type DSP capable d'exécuter une multiplication-accumulation (MAC) à chaque cycle, de puissance de calcul P_{calcul} exprimée en MIPS (Million d'Instruction Par Seconde), le temps de calcul sera : T_{calcul}= (N+1). T_{cyclel}= (N+1) / P_{calcul}. Aussi, la puissance de calcul d'un DSP pour l'implantation d'un filtrage RIF vaut :

$$P_{calcul}(\text{MIPS}) = (N+1).\ Fe\ /10^6 \qquad \text{I.19}$$

Généralement, le type de réalisation préféré se présente sous une forme transposée, à cause de sa performance et sa consommation d'énergie qui sont relativement meilleurs. La multiplication des coefficients de filtrage avec l'entrée du filtre à un impact significatif sur la complexité et les performances de la conception car un grand nombre de blocks de multiplication par une constante sont nécessaires.

Ces structures ne possèdent pas de boucle de retour, elles sont inconditionnellement stables [2].

I.8.2. Structure récursive

Elle correspond au cas où la sortie dépend de l'entrée et des sorties précédentes. C'est essentiellement le cas de filtre RII.

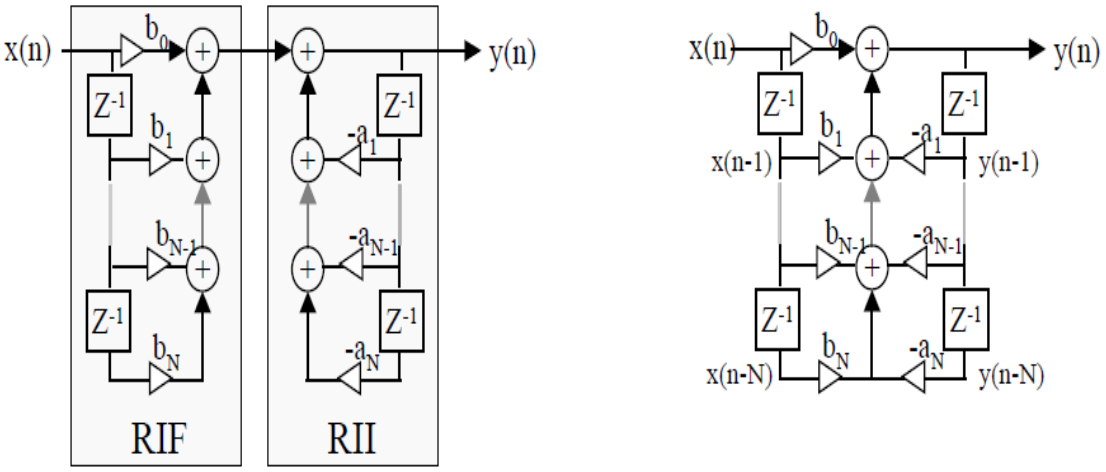
L'équation à la différence est donnée par :

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) = \sum_{i=1}^{M} a_i\, y(n-i) \qquad \text{I.20}$$

Et la fonction de transfert en z par :

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1+\sum_{i=1}^{M} a_i z^{-i}} \qquad \text{I.21}$$

On distingue alors la structure directe et celle en cascade :



a. Cascade b. Structure directe

Figure I.7. Structures de réalisation récursive.

I.9. Conclusion

Dans ce chapitre, nous avons décrit les notions fondamentales de système linéaire invariant ainsi que le filtrage numérique, ses différentes types RII et FIR, et ses structures transversale et récursive.

Le chapitre suivant sera consacré principalement à l'étude de la multiplication par une constante simple/multiple (SCM/MCM) et le nouvel heuristique Radix-2r.

Chapitre II :

Multiplication par une constante SCM/MCM et le recodage RADIX-2r

II.1.Introduction

Ce chapitre introduit le problème de l'optimisation matérielle des systèmes linéaires invariants dans le temps (LTI), et plus particulièrement celui des filtres FIR. Il fournit une discussion sur la multiplication par une constante, en se focalisant principalement sur la multiplication par de simple/multiple (SCM/MCM) constantes. Nous formalisons le problème SCM/MCM, ensuite nous présentons une nouvelle heuristique SCM entièrement prédictible accompagnée de ses complexités de limite supérieures, de moyen et de profondeur d'additions.

II.2. La contrainte de la multiplication par constante

Le problème de la multiplication par une constante est apparu dans les années 1970 pour implémenter une multiplication par des constantes en logiciel. Plusieurs microprocesseurs à l'époque (tels que les Intel 8008) n'avaient pas de multiplieurs, d'où la multiplication doit être fait avec des additions, soustractions et des décalages. Et même lorsque l'instruction de la multiplication est devenue disponible, l'exécution nécessite généralement plusieurs cycles d'horloge.

Maintenant, et avec la notion de pipeline la résolution de problème de la multiplication par une constante pourrait conduire à une réduction pour le temps d'exécution mais pas comme une solution satisfiable.

II.3. Les différents types de multiplication par des constantes

Dans cette section on va présenter les différents types de multiplication par des constantes qui apparaissent dans les systèmes LTI, tel que les filtres FIR avec ses trois formes (directe, transposé et hybride), et les contrôleurs numériques, etc. la multiplication des échantillons de données par des coefficients constants dans les systèmes LTI peut être catégorisée en quatre classes principales. Comme il est présenté dans la figure II.1 :

- La multiplication par une simple constante (SCM) : cette opération réalise la multiplication d'un seul coefficient par une seule variable. Elle fréquemment utilisée dans des contrôleurs numériques et des systèmes DSP telle que la transformée de Fourier rapide (FFT) ;

- La multiplication par de multiples constantes (MCM) ; cette opération calcule la multiplication d'un ensemble de constantes par une seule variable. Cette opération est utilisée dans la forme transposée des filtres FIR ;

- La multiplication par une matrice de constantes (CAVM) ; cette opération réalise la multiplication d'un vecteur d'entrée (CAVM). Cette opération apparait dans les filtres à réponse impulsionnelle infinie (IIR) et la forme directe des filtres FIR ;

- La multiplication par une matrice de constantes (CMVM) ; cette opération réalise la multiplication d'une matrice de constantes par un vecteur d'entrée. CMVM est le cas le plus générale de multiplications par des constantes elle est utilisée dans la forme hybride de filtres FIR.

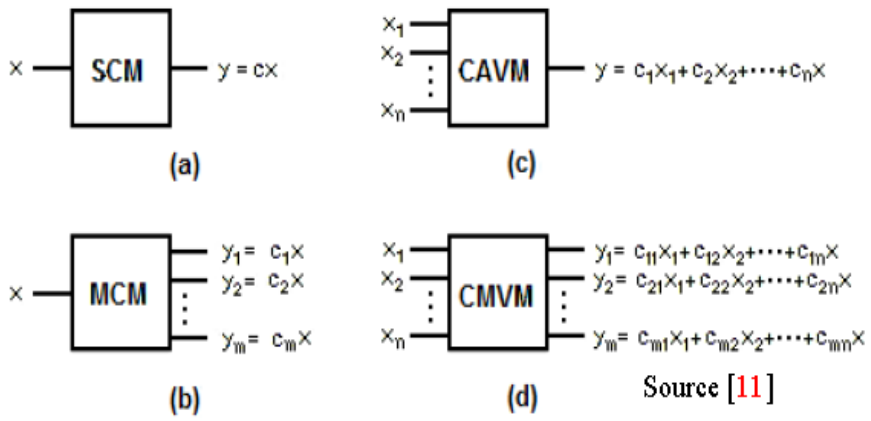


Figure II.1.Types de multiplications par des constantes [11].

Dans ce travail, nous nous basons sur les deux premiers types de multiplication par constantes : la multiplication par des simples constantes (SCM) et la multiplication par multiples constantes (MCM).

II.3.1. La multiplication par une simple constante SCM

La multiplication SCM traite le problème de la multiplication d'une seule constante C*ij* par une variable X*ij*. Pour être implémentée matériellement d'une manière efficace, la multiplication de C*ij* × X*ij* doit être sans multiplieurs (multiplierless), c'est-à-dire en utilisant seulement des additions, des soustractions, et des décalages. Dans une implémentation matérielle, l'opération de décalage est sans coût, parce qu'elle peut être réalisée en utilisant simplement le câblage (c'est-à-dire sans aucune porte). Nous supposons que l'addition et soustraction ont le même coût en termes de surface et de vitesse.

La multiplication SCM est l'opération qui consiste à chercher la décomposition en additions, soustractions, et décalages de la constante C*ij* avec un nombre minimum d'opération d'addition/soustraction. Une bonne décomposition conduit à des avantages considérables en termes de temps d'exécution, de surface, et de consommation de puissance.

Exemple illustratif :

L'exemple est 45 × X $_j$, de nombreuses solutions possibles, seulement quatre des solutions sont présentées :

$$45 \times X_j = (101101_2) \times X_j = X_j \times 2^5 + X_j \times 2^3 + X_j \times 2^2 + X_j \quad \text{II.1}$$

$45 \times X_j = (63\text{-}18) \times X_j = [\,(111111_2) - (010010_2)] \times X_j = (X_j \times 2^6 - X_j) - X_j \times 2^4 - X_j \times 2$ II.2

$$45 \times X_j = (10\bar{1}0\bar{1}01_2) \times X_j = X_j \times 2^6 - X_j \times 2^4 - X_j \times 2^2 + X_j \quad \text{II.3}$$

$$45 \times X_j = U \times 2^2 + U \quad \text{avec } U = X_j \times 2^3 + X_j \quad \text{II.4}$$

- EQ. II.1 est la méthode la plus simple pour la multiplication par une simple constante, elle transforme la constante *C ij* en représentation binaire, convertit les 1 en décalage basés sur leurs positions, et additionne les valeurs décalées. Pour *C ij*=45, EQ II.1 nécessite trois additions et trois opérations de décalage. Le nombre des additions à l'aide de la représentation binaire est inférieur au nombre d'instances « 1 ».

- Une autre méthode (EQ. II.2) utilise les décalages et les soustractions en traduisant les valeurs « 0 » en opération de décalage tout en les soustrayant de la constante de la même longueur composée seulement des 1. La constante de 45 requiert six bits et la constante correspondante de six bits de tous les 1 (111111_2) qui est égale à (63_{10}). Le terme $X_j \times 2^6$ représente le nombre de 63 et les deux termes suivants $X_j \times 2^4$ et $X_j \times 2$ représentent les termes 16 et 2 respectivement (16+2=18).

- La représentation de la CSD (EQ.II.3) encode un nombre constant en utilisant le nombre minimal de chiffres différent de zéro. Par conséquent, lors de la transformation d'une multiplication constante en une séquence de décalage et additions, la représentation de la CSD donne le nombre minimum d'addition. Mais dans ce cas spécial ($C_{ij} = 45$), ne fournit aucun avantage sur EQ.II.1 et II.2.

- EQ. II.4 permet d'obtenir le nombre minimal d'addition pour $Cij = 45$. La réduction des additions vient du partage du terme $U = 9j \times Xj$. Ceci est bien illustré par la figure II.2.b, où les nœuds gris signalent une opération de décalage, et les rouges indiquent une addition. Le nombre total d'opérations est deux additions. Notez que plusieurs solutions avec 2 additions peuvent exister (Figure II.2.a, II.2.b, et de II.2.c).

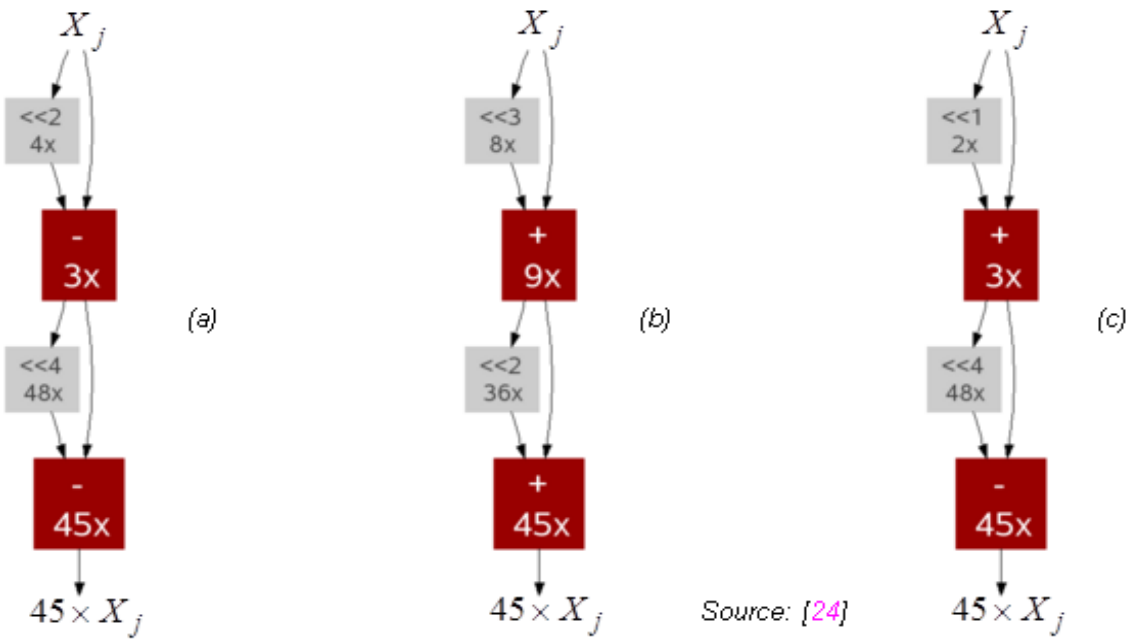


Figure II.2. Le nombre minimal d'additions pour 45 j × X.[15]

Les solutions sont données par le site web de spirale (www.spiral.net). Le signe «<<a» signifie un décalage d'une position ($\times 2^a$*).*

L'objectif de l'heuristique SCM est de fournir des solutions optimales dans un temps de calcul raisonnable. La prévisibilité est une autre caractéristique importante de l'heuristique SCM. Selon la taille des bits de la constante, elle permet de connaître à l'avance (avant implémentation), le nombre maximal d'additions et le coût formant le chemin critique (vitesse).

II.3.2 Multiplication par de multiples constantes (MCM)

MCM est une extension du SCM est le où est la seule variable X $_j$ multiplié par l'ensemble de constantes cibles $\{C_{1\,j},\ C_{2\,j},\ C_{3\,j}, C_{m\,j}\}$ (la colonne entière *j*). L'implémentation de la multiplication de plusieurs constantes par la même variable en utilisant un nombre minimal d'opérations d'addition/soustractions est connue sous le nom de problème de multiplications par de multiple constantes MCM. Ce dernier réduit le nombre d'additionneurs (surface sue silicium) pat le partage de termes partiels. Puisque le problème MCM est la généralisation du problème SCM, il est également NP-difficile.

MCM est utilisé dans de nombreuses applications telles que les filtres numériques FIR, les transformations de signaux linéaires, le traitement d'image et l'arithmétique des ordinateurs.

Exemple illustratif

Nous effectuons les deux multiplications suivantes : 81 × X j et 23 × X j. Fig. II.3.a et II.3.b représentent la meilleure optimisation pour chaque cas individuellement. Les nœuds indiquent plus, alors que le spectacle des bords du montant du transfert. Fig. II.3.a et II.3.b nécessitent deux additions, entraînant quatre additions à effectuer deux multiplications. Fig. II.3.c montre l'optimisation simultanée des deux variables. Le variables peuvent partager une multiplication commune 9 x, par conséquent, le nombre total des additions pour les deux variables est réduit d'une unité (soit un total de trois additions).

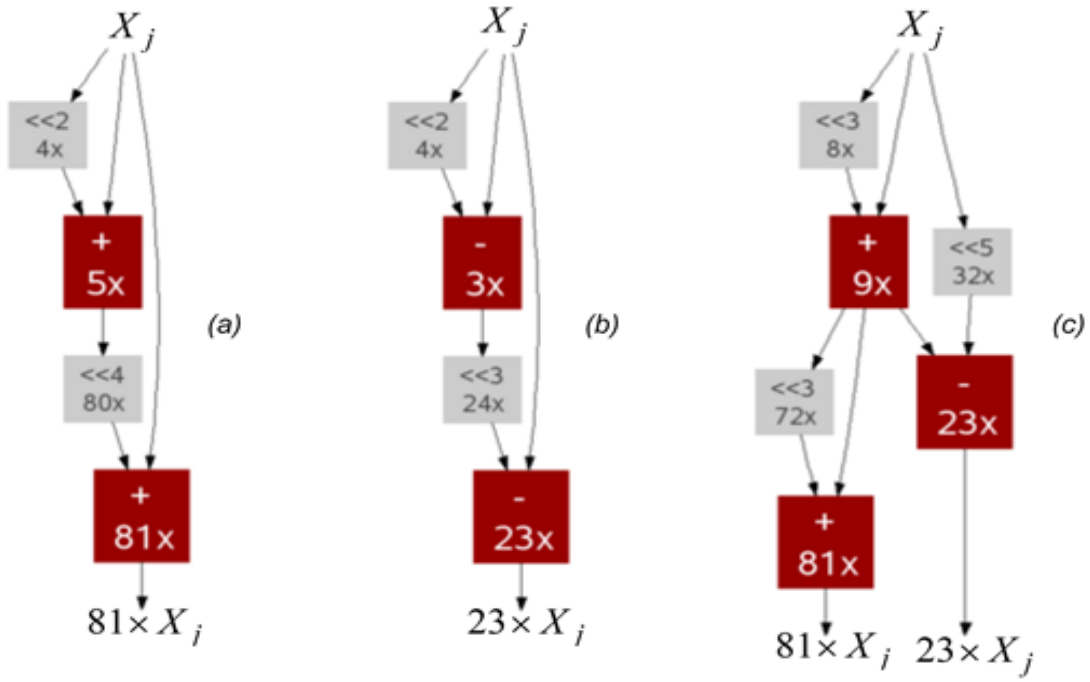


Figure II.3. Multiplication des constantes 81 et 23.

(a), (b) l'optimisation de chaque variable indépendamment ; Cela nécessite deux additions par constante pour un total de quatre additions. (c) l'optimisation simultanée de ces deux variables. Les variables peuvent partager une addition entraînant trois additions pour la multiplication de ces deux constantes.

II.3.3 : définition formelle du problème SCM

Avant de formaliser le problème de la SCM, nous devons définir clairement :

- Le type de la constante: positive, négative, impaire ou paire. Presque toutes les heuristiques de proposées SCM ne gèrent que les constantes impaires/positives, car les heuristiques paires/négatives peuvent être dérivées simplement en utilisant al négation et le décalage. On note que dans ce cas un traitement extra des constantes est nécessaire.

- Les opérations autorisées : addition, soustraction, décalage vers la gauche, décalage vers la droite, et opération « ou ». Limiter le nombre d'opérations rend le problème plus difficile à résoudre. La Plupart des projets des heuristiques permettent seulement l'addition, soustraction et décalage vers la gauche. Décalage à droite permet à quelques heuristiques [] puisque il a des inconvénients, tout en apportant peu de valeur : par exemple, il ne travaille pas avec l'arithmétique modulo 2^K.

La définition formelle suivante s'applique à tous les types de constantes, c'est-à-dire positive/négative et paire/impaire, autorisant l'addition, la soustraction et le décalage vers la gauche uniquement [4].

Soit *C* notre constante, *C* Z. Une suite finie d'entiers u_0, u_1, u_2,…,$u_q$est acceptable pour C si elle satisfait les propriétés suivantes :

- Valeur initiale : $u_0 = 1$;
- Pour tout i>0, $u_i = s_i \times u_j \times 2^{a_i} + r_i \times u_k \times 2^{b_i}$; avec j, k < i ; s_i, $r_i \in \{-1,0,1\}$; et $a_i, b_i \in N$;
- Valeur finale : $u_q \times 2^{c_q} = c$, avec $c_q \in N$.

Le problème est de trouver un algorithme qui prend le nombre *C* et qui génère une séquence acceptable $(u_i)_{0 \le i \le q}$ aussi courte que possible. q est appelé la qualité, ou la longueur. On note que pour retarder les déplacements, nous pouvons limiter b_i à 0 (cela casse la symétrie, mais fait moins variable). Ainsi, un nombre arbitraire X étant donné la solution correspondante itérativement calcule $u_i \times X$ déjà calculé les valeurs, $u_j \times X$ *et* $u_k \times X$ jusqu'à

obtenir C × X. Note qu'avec cette formulation, on a calculé la valeur permise de réutiliser le tout déjà. C'est pourquoi certaines solutions générées peuvent besoin de stocker des résultats temporaires.

SCM/MCM est un problème fondamental dans le contrôle, DSP et les télécommunications.

II.3.4 : Les algorithmes SCM/MCM existants

Les algorithmes MCM existants peuvent être divisés en quatre grandes classes :

- Algorithmes de recodage des Chiffres tels que la représentation canonique chiffres signés (CSD) [6,8], stand recodage [4,5], et Dimitrov DBNS recodage [5 ,8] ;
- Algorithme d'élimination des sous-expressions communes (CSE) à l'aide des critères spéciaux effectués après un premier recodage. Des exemples typiques sont Hartley [7,8], [6,8] de Lefèvre et Boullis [7,8] ;
- Algorithme de réalisation des graphes acycliques (DAG) cette catégorie comprend les Bernstein [7,8], MAG [20,8] et Hcub [5,8].
- Algorithmes hybrides combinant CSE et DAG tel que l'algorithme optimal récent BIGE [14,8].

II.3.5. Définition des métriques pour SCM/MCM

L'objectif des algorithmes proposés pour la multiplication SCM/MCM est de produire une implémentation matérielle efficace dans un délai raisonnable, c'est-à-dire rapide (haut vitesse), compacte et à faible consommation de puissance.

Il existe un certain nombre de métrique pour la multiplication par une simple constante SCM et de multiple constante MCM, mais le plus couramment utilisés sont :

- **Limite supérieure (*Upb*)** : pour chaque constante Ci à N-bit corresponds à Ai additions, pour l'implémentation de $Ci \times X$. $Upb = \max Ai$.
- **Profondeur d'adder (Ath)** : soit Di le nombre d'adder qui traversent l'entrée à une des sorties du circuit logique de la multiplication par une constante (le chemin le plus long). $A_{th} = \max (Di)$.

- **Moyenne (Avg) :** pour chaque constante *Ci* à N-bit corresponds à *Ai* additions, pour la mise en œuvre de *Ci* × *X*, $Avg=\sum_{i=1}^{m} A_i/m$, où m est le nombre total des constantes.

En fait, bien que formellement, la profondeur de l'additionneur est définie par une estimation de chemin le plus long à travers le circuit logique (chemin critique)[9,8].

II.3.6: Les principales Limitations des algorithmes SCM/MCM existants

L'optimisation des systèmes linéaires est un sujet essentiel qui a été au centre des efforts continus au cours des dernières années, ce qui entraîne un nombre impressionnant d'algorithmes SCM/MCM. Parmi les principales limitations de ces algorithmes existants on trouve :

- La Prévisibilité
- La durée de stockage en mémoire
- Le risque de débordement
- La facilité d'utilisation et la mise en œuvre

II.4. Le nouvel algorithme de recodage (RADIX-2^r)

Il repose sur l'arithmétique radix-2r, traite l'heuristique SCM/MCM susmentionnée, facile à utiliser et plus sécurisé en cas d'un dépassement de capacité.

Principe :

Une constante C à N bits est exprimée en radix-2^rcomme suit :

$$C = \sum_{j=0}^{(n/r)-1} C_{rj-1} + 2^0 C_{rj} + 2^1 C_{rj+1} + 2^2 C_{rj+2} + \cdots + 2^{r-2} C_{rj+r-2} - 2^{r-1} C_{rj+r-1}$$

$$= \sum_{j=0}^{(n/r)-1} Q_j \, 2^{rj} \qquad \text{II.5}$$

Avec C-1=0 et r∈ N^*

En cas d'un nombre signé, il s'agit de la formule suivante :

$$-2^{r-1}C_{rj+r-1} \times 2^{rj} + C_{rj+r-1} \times 2^{r(j+1)} = C_{rj+r-1} \times 2^{rj+r-1}$$

La fonctionnalité d'algorithme RADIX-2r a une prévisibilité permet la génération des contrôleurs LTI entièrement capables de satisfaire les différentes exigences, telles que :

- Générer un contrôleur comprenant un nombre minimal d'additions (contrôleur plus compact);
- Générer un contrôleur avec un chemin critique plus court (contrôleur plus rapide) ;
- Activer un compromis entre le nombre d'additionneurs et le nombre d'étapes additionnelles, c'est-à-dire entre la surface et la vitesse ;
- Génère un contrôleur satisfaisant la contrainte de retard de sorte que le nombre d'additionneurs /soustracteurs soit minimisé ; etc.

L'arithmétique radix-2r est un outil mathématique simple et puissant qui pourrait être exploré pour la résolution de problème de la multiplication par une constante, les chapitres qui suivent montrer l'efficacité de cet algorithme.

II.5. RADIX-2^r pour la multiplication par une constante SCM/MCM

L'arithmétique RADIX-2r est une heuristique développée pour minimiser le nombre des additions dans la multiplication par une constante. Sa complexité est sous linéaire [8]. Elle permet de générer un recodage d'une constante C afin d'avoir un produit C x X qui utilise un nombre d'additions optimisé.

La représentation RADIX-2r est développée par SAM en 1990 [8]. Dans cette technique un nombre représenté en complément à 2 de N-bits est écrit comme suit:

$$C = \sum_{j=0}^{(N+1)/r-1)} C_{rj} + 2^0 C_{rj} + 2^1 C_{rj+1} + 2^2 C_{rj+2} + \cdots + 2^{r-2} C_{rj+r-2} - 2^{r-1} C_{rj+r-1} \times 2^{rj}$$

$$= \sum_{j=0}^{(N+1)/r-1} Q_j \, 2^{rj} \qquad \text{II.6}$$

Ou X-1=0 et r∈ N^*

A des fins de simplicité et sans perte de généralité, on considère que r est un diviseur de N eq (II.6), la représentation complément à 2 de x est divisée en n+1/r complément a 2 segments (Q_j), représentés sur r bits car il va de 2^0 jusqu'à 2^{r-1} . Toutefois Q_j a besoin d'un bit supplémentaire (C_{rj-1}) égal au bit de poids fort du segment précèdent (Q_{j-1}), et donc chaque 2 segments adjacents ont un bit en commun. L'ensemble des chiffres DS (2^r) correspond à (II.1) tel que :

$$Q_j \in DS(2^r) = \{-2^{r-1}, -2^{r-1}+1, \cdots, 2^{r-1}-1, 2^{r-1}\}$$

Le produit devient :

$$C \times X = \sum_{j=0}^{(N+1)/r\,-1} X \times Q_j \times 2^{rj} \qquad \text{II.7}$$

Le signe des termes $Q_j$correspond au bit $C_{rj+r-1} et \left|Q_j\right| = 2^{kj} \times m_j avec\ kj \in 1,2,3,\cdots,r-1\ et\ mj \in OM\{2^r\} \cup \{0,1\} ou\ OM(2^r) = \{3,5,7,\cdots,2^{r-1}-1\}$,
dans RADIX. $2r\ OM(2^r)$ est l'ensemble des nombres positifs impairs avec $|OM(2^r)| = 2^{r-2}-1$

Chaque Q_j comprend r+1 bits. Le nombre total des différentes combinaisons des bits est 2^{r-1}, à partir de (II.1) seulement deux combinaisons produisent Q_j=0, en cas où tous les r+1 bits sont à '0' ou '1'
Finalement on peut exprimer le produit comme suit :

$$C \times X = \sum_{j=0}^{(N+1)/r\,-1} (-1)^{c_{rj+r-1}} \times (m_j \times X) \times Q_j \times 2^{rj+kj} \qquad \text{II.8}$$

Contrairement à la multiplication par une variable (Y x X) où tous les produits partiels (mj x X) doivent être pré calculés, dans la multiplication par une constante (C x X) seulement les sous-ensembles sont nécessaires. En réalité, le nombre des produits partiels est le nombre de différentes valeurs de $m_j$générées par le processus de décodage des (N+1)/r segments (les termesQ_j). Donc la génération des produits partiels (PP) consiste premièrement si m_j≠0, lors du calcul des PP m_jx X s'il n'est pas pré calculer avant, ensuite il est soumis à un décalage à

gauche matériel de $r_j + k_j$ positions. Finalement la négation $(-1)^{c_{rj+r-1}}$ de dépend du bitc_{rj+r-1}.

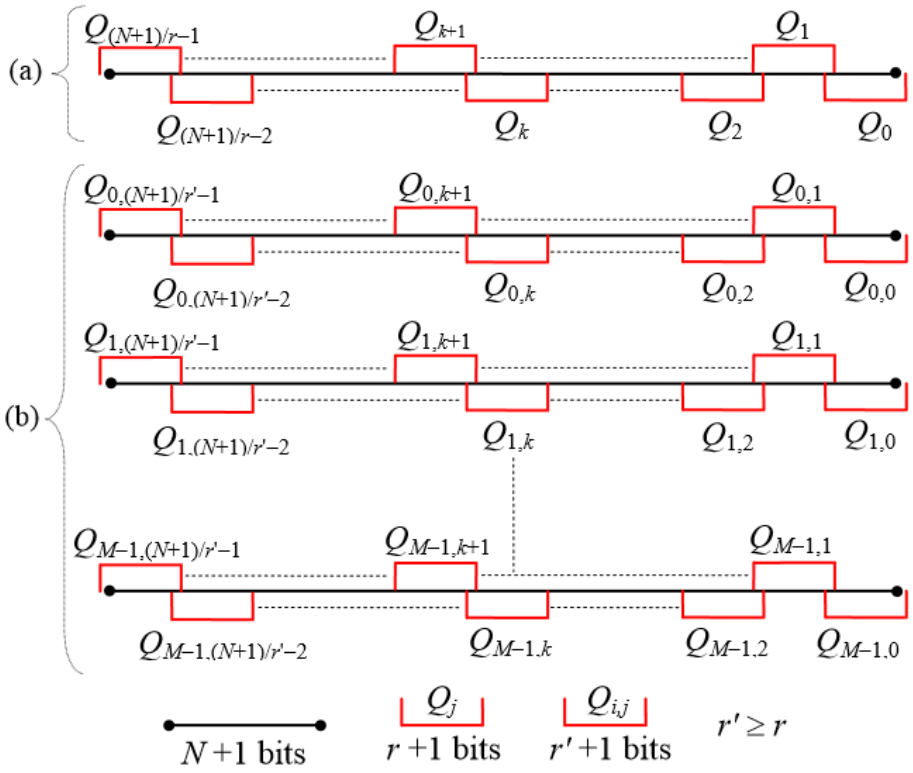


Figure II.4. Partitionnement des segments d'une constante de N bits sous RADIX-2r (a) Radix-2r SCM, (b) Radix-2r MCM

II.5.1. Nombre maximal d'additions pour une constante de N bits

D'un côté il y a (N+1)/r itérations dans (II.8), chaque itération génère un produit partiel, donc le nombre maximal de PP est (N+1)/r qui requiert au max $N_{PP} = \frac{N+1}{r} - 1$ additions. D'un autre côté, un maximum de $2^{r-2} - 1$ PP non triviaux {3.X, 5.X,…,($2^{r-2} - 1$). X} peut être invoqué lors du processus de génération des PP. Ils sont construits en utilisant la méthode binaire, du bit de poids le plus faible au le bit de poids le plus fort, et pour cela, les éléments de m_j3,5,..., $2^{r-2} - 1$ sont construits un par l'autre, chaque fois en utilisant une seul addition entre les éléments qui sont déjà construits avec une puissance de 2.

Le processus de construction des PP non triviaux est illustré par l'exemple suivant ou en prend RADIX-2^6 :

OM(26) = { 1, 3, 5, 7, 9,11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31}

$$=\{1\} \cup \{2^1+1=3\} \cup \{2^2+1=5, 2^2+3=7\} \cup \{2^3+1=9, 2^3+3=11,$$

$$2^3+5=13, 2^3+7=15\} \cup \{2^4+1=17, 2^4+3=19; 2^4+5=2^1,$$

$$2^4+7=2^3, 2^4+9=2^5, 2^4+11=2^7, 2^4+13=2^9, 2^4+15=31\}.$$

Donc il est clair que l'EQ (II.8) contient que l'opération d'addition, de soustraction, et de décalage à gauche. La figure II.5 représente tous les détails nécessaires à l'implémentation hardware.

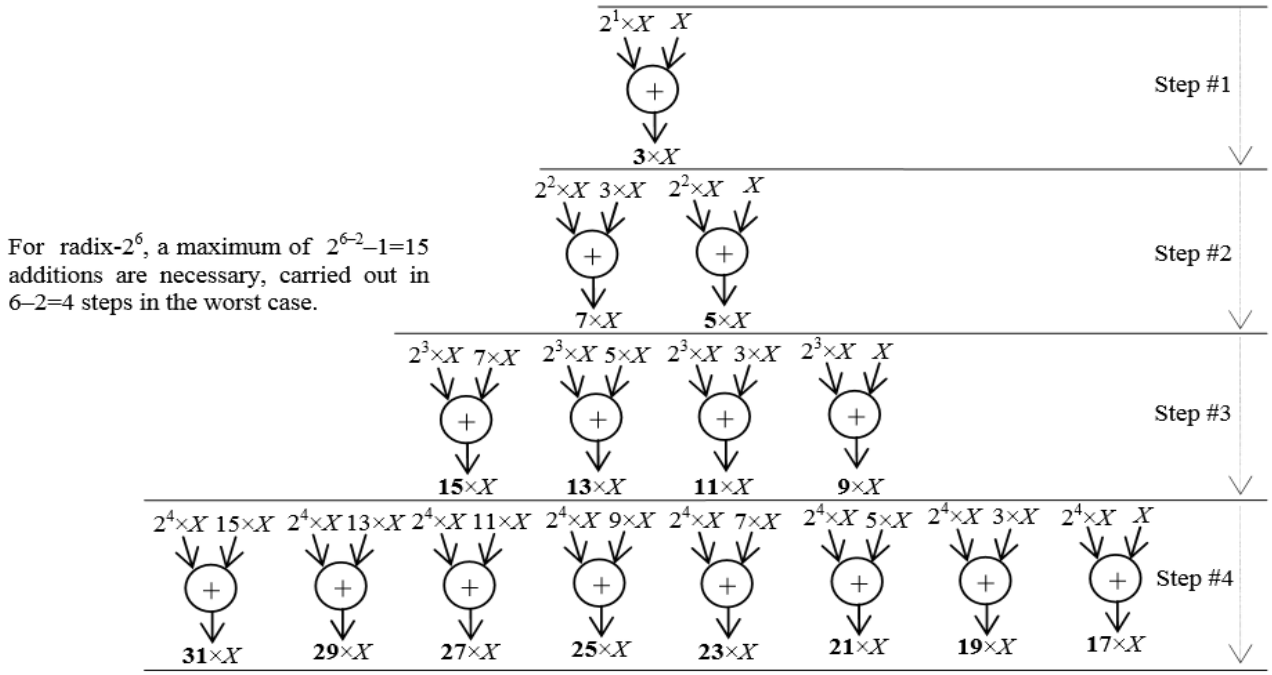


Figure II.5. Ordonnancement séquentiel de l'ensemble des produits partiels nécessaire pour RADIX-2r [8].

Par conséquent, le nombre total d'additions requis par Radix-2r est égale à :

$$U_{pb}(r) = M_{om} + M_{pp} = \left\lceil \frac{N+1}{r} + 2^{r-2} - 2 \right\rceil \qquad \text{II.9}$$

$U_{pb}(r)$ est minimal pour $r = 2.W(\sqrt{(N+1).\log(2)})/\log(2)$ ou W est la fonction de Lambert qu'est la réciproque de la fonction de variable complexe f définie par f(w)= we^w, c'est-à-dire que pour tous nombres complexe z et w, nous avons :

$$z = e^w \Leftrightarrow w = W(z)$$

Le minimum est obtenu pour un des deux nombres entiers les plus proches de r et les deux nombres doivent être testés. La figure II.3 représente les limites supérieures "*Upper-bounds*" des nombres d'additions pour CSD, DBNS, et RADIX-2r.

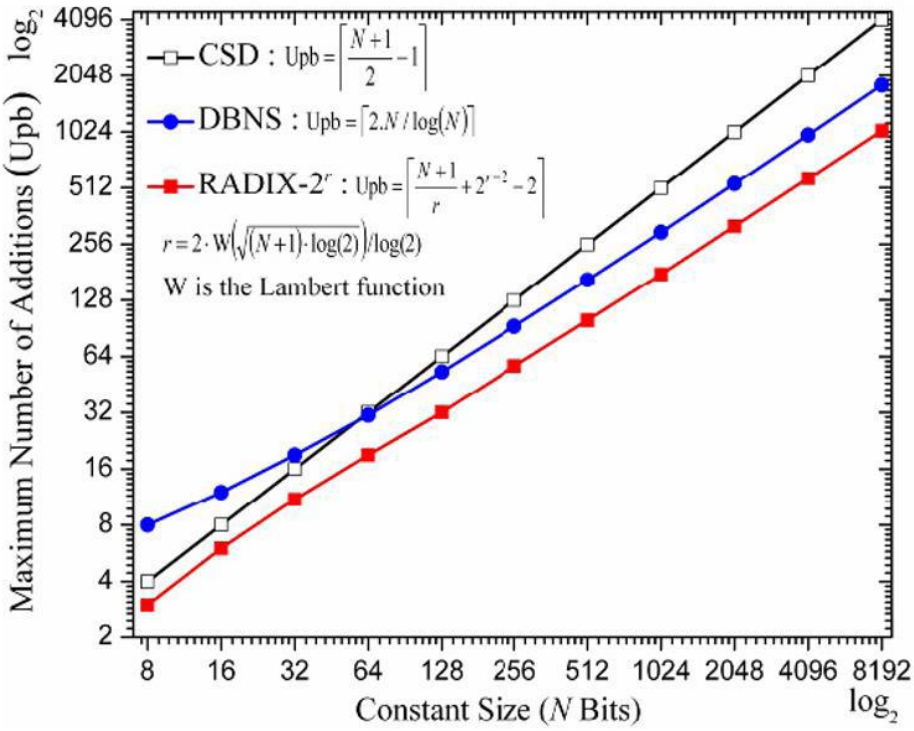


Figure II.6**.** Comparaison de limites supérieures pour une constante de N bits

En ce qui concerne le nombre moyen d'additions, il a été calculé de manière exhaustive pour des valeurs de C variant de 0 à 2^N-1, pour N = 8, 16, 24 et 32. Mais pour N = 64, nous avons calculé la moyenne avec10^5,10^6,10^9et 10^{10} valeurs aléatoires de C uniformément distribuées. Alors que la différence entre les quatre résultats obtenus sont insignifiants (<10-3), la valeur Moyenne est autour de 15.7165 additions [8]. Les résultats sont présentés dans le tableau II.1.

Pour N = 64, RADIX-2r utilise 23,12% en moyennes moins d'additions que CSD. Ce gain semble progresser linéairement pour des valeurs faibles de N.

Tableau II.1. Nombre moyen d'additions de RADIX-2r et CSD[8]

Longueur de bit N de constante	CSD		RADIX-2r		Economie (Avg,%)
	Avg	Upb	Avg	Upb	
8	1.7882	4	1.8645	3	-4.2668^{+}
16	4.4445	8	4.5127	6	-1.5344^{+}
24	7.1111	12	6.7994	9	4.3832
32	9.7777	16	8.9627	11	8.3352
64	20.4444	32	15.7165^{*}	19	23.1256

La figure II.7 représente les valeurs moyennes des différentes heuristiques

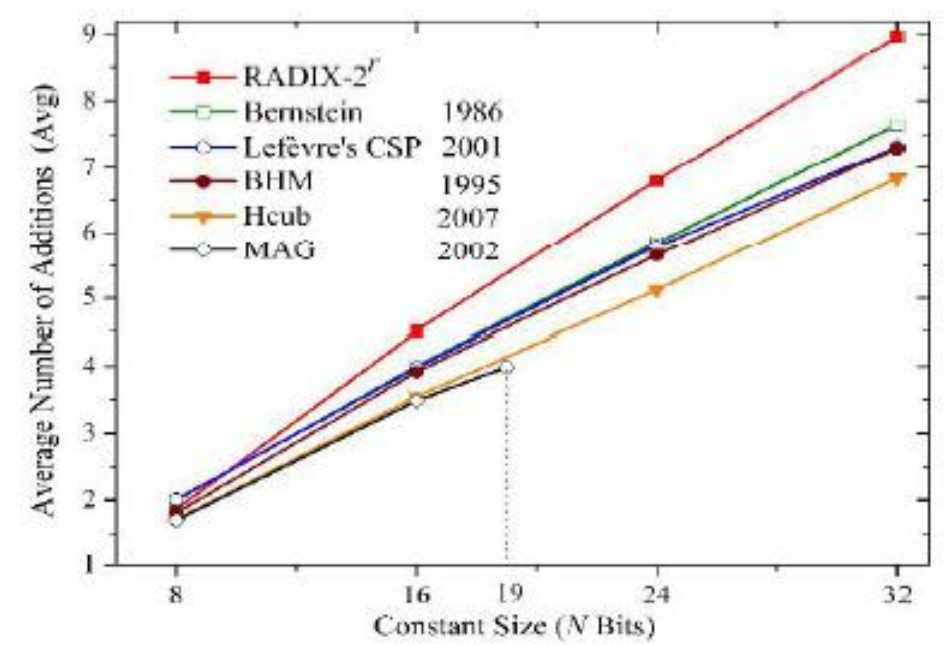


Figure II.7. Longueur de bits N d'une constante [8]

Le tableau II.2 contient les différentes relations de RADIX-2r pour un nombre M de constantes (MCM) à longueur de bits variable.

Tableau II.2. Equations de RADIX-2r pour des constantes non négatives de différentes longueurs de bit [16]

Métriques	Equations
Coût en additionneurs	$Upb(r') = M \times \left\lceil \frac{N+1}{r'} \right\rceil + 2^{r'-2} - 1 - M$ avec $r'=r_1$ ou $r'=r_2$
Profondeur en additionneurs	$Ath(r') = \left\lceil \frac{N+1}{r'} \right\rceil + r' - 3$ avec $r'=r_1$ ou $r'=r_2$
Moyenne en additionneurs	$-M + Avg_{pp} + Avg_{om} \le Avg(r') \le -M - 1 + Avg_{pp} + 2^{r'-2}$ avec $Avg_{pp} = \left(1 - 2^{-r'}\right) \times M \times \left\lceil \frac{N+1}{r'} \right\rceil$, $Avg_{om} = \sum_{j=0}^{M \times \left\lceil \frac{N+1}{r'} \right\rceil - 1} \left\{ \sum_{k=1}^{2^{r'-2}-1} P(m_{jk}) \times \left[1 - P(m_{jk})\right]^j \right\}$, $P(m_{jk}) = \frac{\log_2 \left\lceil \frac{2^{r'-1}}{2 \times k + 1} \right\rceil}{2^{r'-1}}$, et $r'=r_1$ ou $r'=r_2$
$r_1 = 2 \cdot W\left[\sqrt{M \cdot (N+1) \cdot \log(2)}\right] / \log(2)$	$r_2 = W\left[4.M \cdot (N+1) \cdot \log(2)\right] / \log(2)$

Contrairement aux autres algorithmes MCM, chaque constante dans RADIX 2r est implémentée à part, indépendamment des autres, mais toutes les autres constantes utilisent les même PP.

II.5.2. Les caractéristiques de RADIX-2r

II.5.2.1. Totalement prévisible

La limite supérieure (Upb) en nombre d'additions, de profondeur de l'additionneur (Ath) et la moyenne (Avg), sont connus par des formules analytiques exactes. Cette caractéristique non seulement permet aux concepteurs d'avoir une image avant la mise en œuvre sur la surface, la vitesse, et la consommation de puissance, mais elle permet également aux outils de synthèse de satisfaire rapidement les contraintes d'utilisateur et d'effectuer les compromis nécessaires sans les rétroactions « sans fin » à la recherche de la solution appropriée. Notez qu'aucun des algorithmes de MCM existants n'est prévisible en Upb, Ath ou Avg [11].

II.5.2.2. Sous-linéaire

Pour un nombre de constantes non négatives donné M avec une longueur de bit N, RADIX- 2r présente une complexité sous-linéaire O (M × N / r), où r est une fonction de (M, N). Cela signifie qu'il n'a aucune limitation concernant le couple (M, N). Pour les problèmes de grande complexité (M × N >>), RADIX-2r est très probablement le seul qui soit à même capable de fonctionner. Notez que les meilleures heuristiques MCM ont une complexité polynomiale ou exponentielle [11].

II.5.2.3. Solution haute vitesse et faible consommation de puissance

Profondeur d'additionneur(Ath) n'est pas seulement une mesure de chemin critique (vitesse), mais aussi un bon indicateur de la consommation d'énergie. Les algorithmes de MCM existants ne peuvent pas rivaliser avec Radix-2r, car il permet une réduction logarithmique de Ath, alors que c'est impossible dans les autres algorithmes en raison des additions partagés [11].

II.6. Les spécifications d'entrée du système par l'application RADIX-2r

La solution RADIX-2r est une application, développée au CDTA, et est basée sur l'arithmétique de multiplication par multiple constantes de *Ms A.K Oudjida*. Elle prend une matrice de constantes stockées dans un fichier texte comme entré puis elle génère dans un fichier le recodage des constantes du système avec ses spécifications comme sortie.

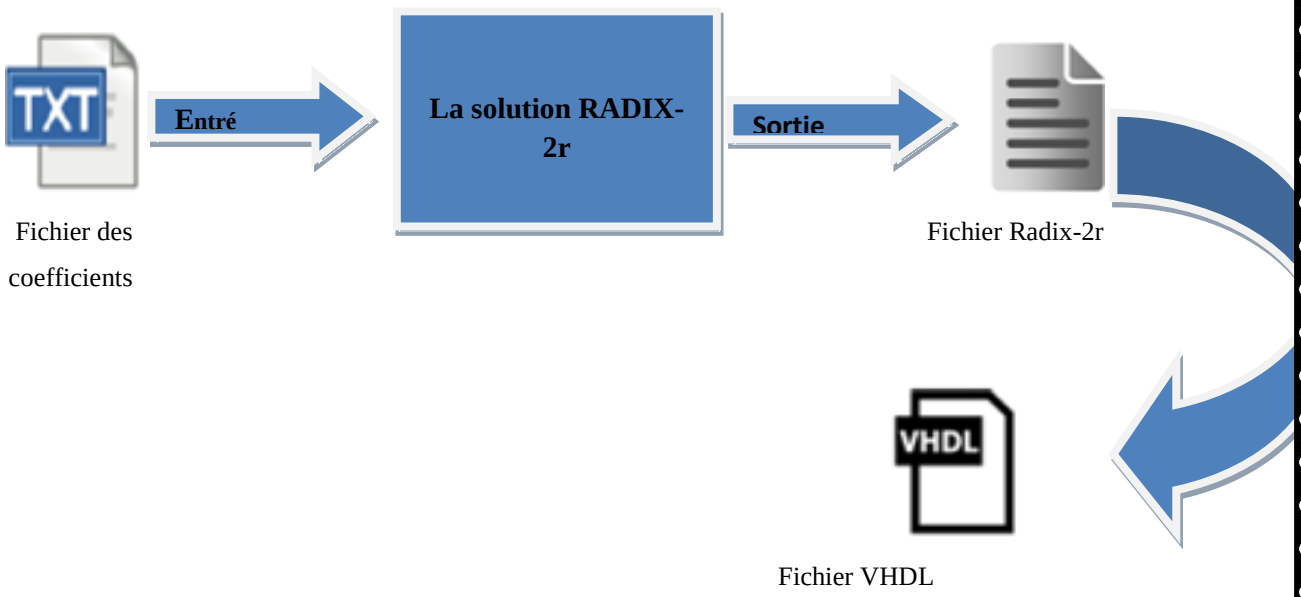


Figure II.8. Le programme RADIX-2r et VHDL construit

Cette application donne les résultats suivants :

- Les multiples impairs utilisés ainsi que le recodage des constantes selon l'arithmétique RADIX-2r.
- La limite inferieure en nombre d'additionneurs.
- La limite supérieure en cas ou les additionneurs est connectés en série ou en structure d'arbre.
- La longueur de bits maximale et minimale des constantes.
- Le nombre minimal de constantes (Hmin Set) utilisées pour la représentation du SCM/MCM (les odd multiples + les constantes).
- La valeur optimale de r pour avoir une limite superieure minimale.
- La valeur de r pour avoir un nombre minimal d'additionneurs.

Le fichier généré par le programme RADIX-2r qui contient le recodage ainsi que les spécifications d'un exemple ou on a C = 10599 est illustré par la figure II.6.

```
10599

*** Radix-2^r MCM solution of the reduced set (Hmin set) including positiv

    3 = +1<<1 +1<<0
    7 = +1<<3 -1<<0
10599 = +7<<0 +3<<5 -7<<8 +3<<12

*** Radix-2^r MCM Solution for the user given constants (H set) ***

10599 = +7<<0 +3<<5 -7<<8 +3<<12

With:

    7 = +1<<3 -1<<0
    3 = +1<<1 +1<<0


*** Radix-2^r solution summary ***

Lower-bound in adder-cost   = 3
```

Figure II.9 : Le recodage de 10599 et sa spécification par l'algorithme RADIX-2r

II.7. Le principe de RADIX-2r SCM/MCM au niveau bit

La multiplication par une constante SCM/MCM sous RADIX-2r est basée sur l'addition, la soustraction, et le décalage à gauche. Au niveau bloc d'additionneurs (Adder block) l'addition et la soustraction sont faite directement, bien qu'au niveau bit on sait que chaque deux Qj adjacent (Qj,Qj+1) sont décalés un par rapport à l'autre par un décalage à gauche Kj. Le résultat de l'addition/soustraction donc soit l'addition de tous les bits de Qj avec celle de Qj+1, alors qu'on peut bénéficier un nombre très important des additionneurs complets par l'élimination des addition/soustraction qui sont pas nécessaire pour l'opération. On ce qui concerne le bit ou il y a le décalage à gauche, dans la majorité des cas on fait le déplacement de ces bits vers le poids faible du résultat plus une addition/soustraction des bits restants.

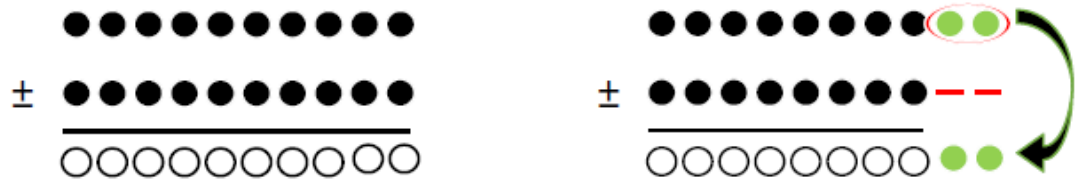


Figure II.10. Le principe d'optimisation au niveau d'additionneurs de 1 bit

II.7.1. Extensions de signe

Nous considérons le cas général de MCM ($C_0, C_1, C_2, \ldots, C_{M-1}$) × X Où Ci est une constante non négative et X représenté au format complément à deux. Cependant, en arithmétique complément à deux, le signe de tous les opérandes doit être étendu à la longueur en bits du résultat avant toute opération.

Deux produits partiels PPs ($\ldots+2^{r(j+1)} \times Q_{j+1} \times X + 2^{rj} \times Q_j \times X + \cdots$) exigent une extension de signe SE variant de 1 à 2r-1 bits. Quand $Q_{j+1} < Q_j$ l'extension de signe SE va être minimale et vise versa. Un exemple d'une extension de signe est donné dans la figure II.11 ou la longueur de bits de X égal a 8, r=3, j=0 [24].

$$\begin{cases} Q_j = \mp 2^2 => 2^{rj} \times Q_j \times X = 2^0 \times 1 \times X = X \\ Q_{j+1} = \mp 2^2 => 2^{r(j+1)} \times X = 2^3 \times 2^2 \times X = 2^5 \times X \end{cases}$$

$$\begin{cases} Q_j = \mp 2^2 => 2^{rj} \times Q_j \times X = 2^0 \times 2^2 \times X = 2^2 \times X \\ Q_{j+1} = \mp 2^2 => 2^{r(j+1)} \times Q_{j+1} \times X = 2^3 \times 1 \times X = 2^3 \times X \end{cases}$$

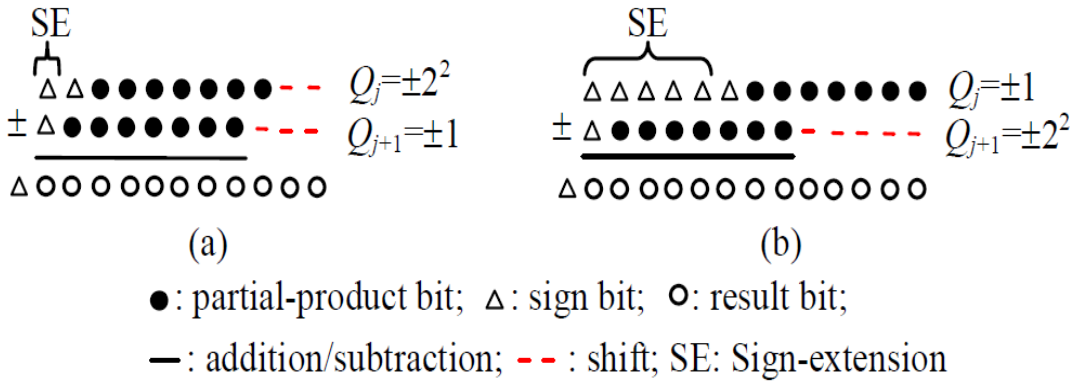


Figure II.11. La méthode d'extension de signe appliqué dans RADIX-2r [17]

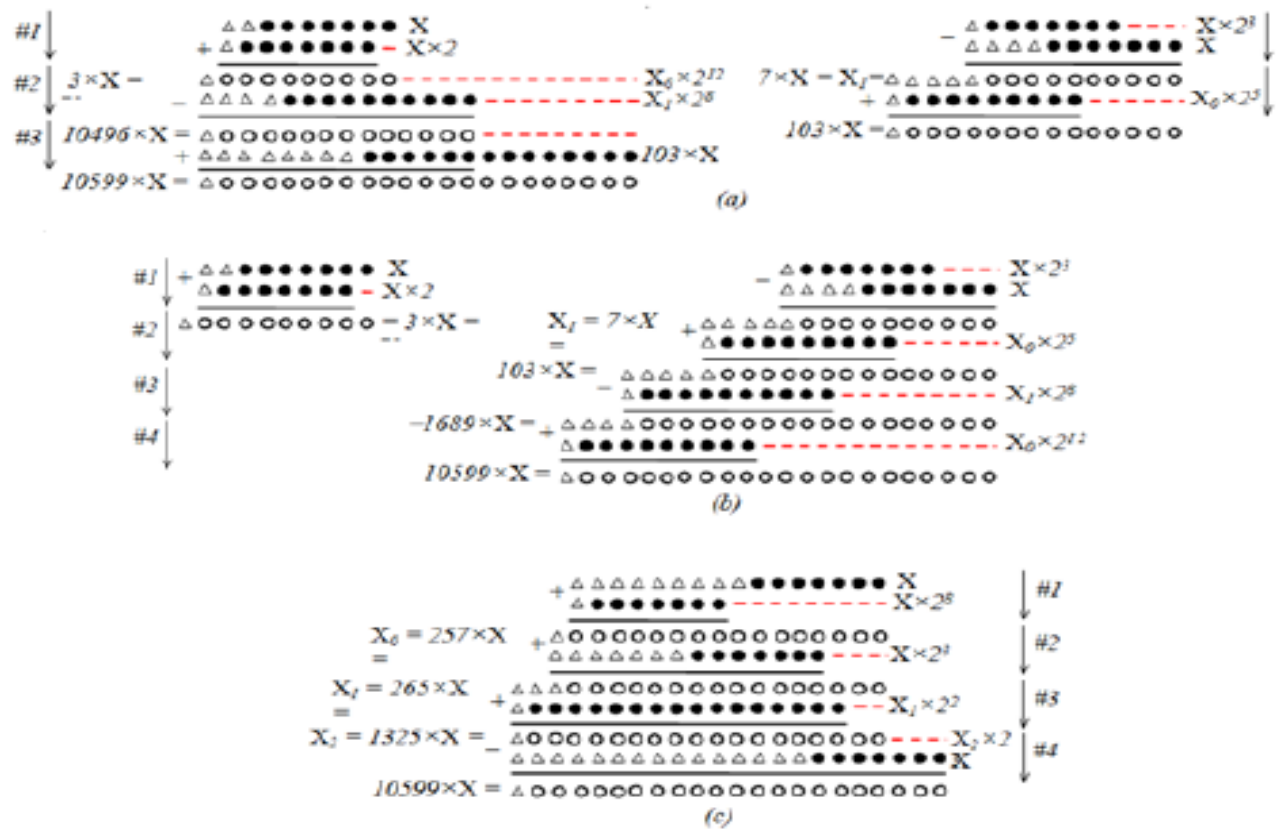


Figure II.12. Implémentation au niveau de bits de 10955 x *X* en utilisant radix-2r parallèle (a), radix-2r sériel (b), Hcub (c) [17]

II.7.2. L'avantage de RADIX-2r au niveau bit

RADIX-2r est un algorithme efficace quelque soit la taille en bit (N) de la constante. C'est le seul algorithme qui offre un temps d'exécution sous-linéaire ($OM \times N / r$ ') [16] par rapport à la complexité du problème. Par contre les autres algorithmes offrent des complexités polynomiales ou carrément exponentielles. Quand la constante est très grande, le temps d'exécution sera très important. Le tableau II.1 représente la complexité des algorithmes MCM.

Hcub	$O(M^4 \times N^5 \times \log(M \times N) + M^3 \times N^6)$
BHM	$O(M^3 \times N^4)$
Lefèvre CSP	$O(M^3 \times N^3)$
RAG$_n$	$O(M^2 \times N^3 \times \log(M \times N))$
MAD$_c$	$O(M \times 2^N)$
NAIAD	$O(M \times 2^N)$
SIREN	$O(M \times 2^N)$
CSD	$O(M \times N)$
R3	$O(M \times N)$
RADIX-2^r	$O(M \times N / r')$

II.8. Le développement d'une solution SCM/MCM au niveau bit

Le principe de la solution au niveau bit est le même qu'au niveau bloc d'additionneurs où on utilise les spécifications des constantes du système SCM/MCM générées par RADIX-2r, puis extraire les coefficients m_j et k_j, ainsi que les caractéristiques particulières de chaque constante pour construire une description matérielle en VHDL équivalente aux spécifications utilisées. Le nombre d'additionneurs 1 bit (FA) nécessaires ne dépasse jamais le nombre calculé par la formule. La procédure de détection de la zone des multiples impaires, la conversion des données extraites, ainsi que le stockage des coefficients mj et kj sont les mêmes. Le calcul du nombre d'additionneurs 1 bits, du système est fait par l'utilisation des coefficients kj et mj et est stocké dans la liste chainée des données.

II.8.1. Stratégie de calcul du nombre des additionneurs 1 bits

Afin d'obtenir de meilleures performances du model en termes de surface occupée, vitesse d'exécution, ainsi que la consommation d'énergie, l'utilisation d'un nombre d'additionneurs exact est nécessaire pour avoir les résultats désirés. C'est pourquoi la procédure de calcul des additionneurs 1 bit utilisé prend une importance majeure lors du développement de la solution.

Le calcul du nombre d'additionneurs est basé sur le calcul de la longueur en bits de chaque couple d'opérandes successifs par la relation :

$$bit\_lenght = m_j + log_2 C$$

Puis la comparaison de ces deux longueurs et l'élimination des additionneurs qui font l'addition avec des zéros (les additionneurs ou il y a un décalage). Le seul cas ou l'utilisation d'additionneurs complet sans élimination c'est lorsque $kj[i] > kj[i+1]$ et le deuxième opérande est négatif ($mj[i+1]<0$) , comme le montre la figure II.3(d), les autre cas (a), (b), (c) permettent l'annulation des additionneurs 1 bit égaux au coefficient de décalage k_j.

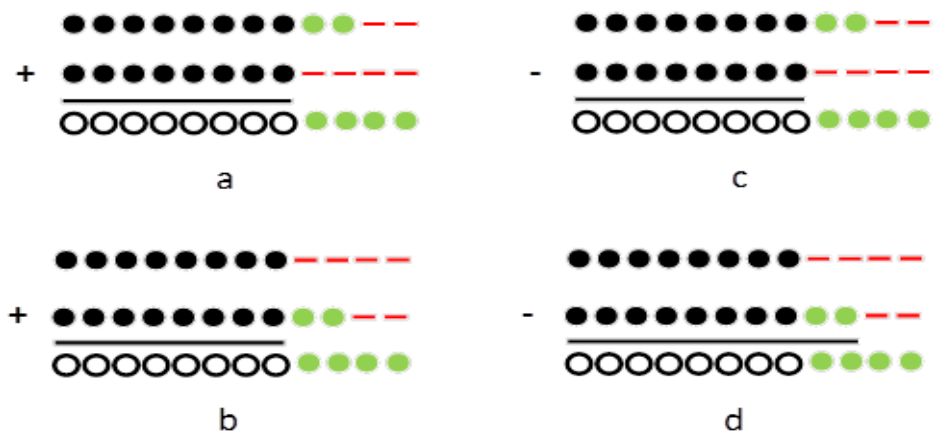


Figure II.13. Les possibilités d'une addition/soustraction en RADIX-2r

II.8.2. La génération du code VHDL

VHDL est un langage de description matériel utilisé pour décrire le comportement et la structure des systèmes numérique, l'acronyme VHDL signifie VHSIC Hardware Description Langage, et VHSIC lui-même signifie Very High Speed Integrated Circuit, toutefois, VHDL est un langage de description matériel à usage général qui peut être utilisé pour décrire et simulé une grande variété de systèmes numériques.

La manipulation au niveau bit consiste à manipuler les données à l'aide des opérations booléennes ET (AND), OU (OR), OU exclusif (XOR) et NON (NOT), ainsi que les décalages logiques et arithmétiques. La génération du code VHDL au niveau bit passe par plusieurs étapes d'analyse et de traitement des cellules de la liste chainée des données stockées dans le but d'avoir des lignes d'instruction bien précises qui décrivent le système avec une utilisation exact des ressources. On s'appuyant sur une description structurelle où on utilise une description d'un composant (additionneur complet 1 bit) puis instancier ce composant afin d'obtenir un bloc d'additionneur optimisé avec une bon fonctionnement de chaque composant élémentaire qui représente une multiplication C x X.

II.9. La description VHDL du SCM/MCM

II.9.1. L'additionneur complet 1 bits (FA)

C'est un circuit qui fait l'addition en bit, il a 3 entrées pour le premier bit, le deuxième bit et le retenu précédent comme un troisième bit. La sortie est constitué de la somme et de la retenu.

Le schéma ci-dessus ainsi que le tableau représente le circuit d'un additionneur complet 1 bit et sa table de vérité respectivement

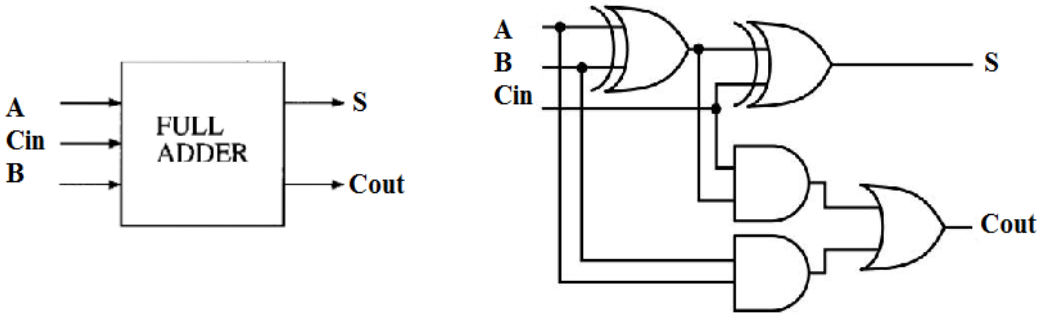


Figure II.14 .Un additionneur complet bit

Tableau II.3. Table de vérité d'un additionneur complet 1 bit

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

II.9.2. L'additionneur soustracteur

L'additionneur/soustracteur est un circuit qui fait les deux opérations (l'addition, la soustraction). La soustraction des deux nombres A et B donnée par $S = A+B'+1$, ou B' est le complément à 1 de B, l'implémentation hardware de cette formule fait par une porte XOR ou l'ensemble des bits de B sera inversé puis à l'aide de l'entrée de la retenu on ajout un 1 pour obtenir une conversion complément à 2. La figure II.7 représente un exemple d'un additionneur soustracteur de 4 bits. La table de vérité (Tableau II.3) représente l'inversement des bits lorsque on fait une soustraction (Add/sus =1).

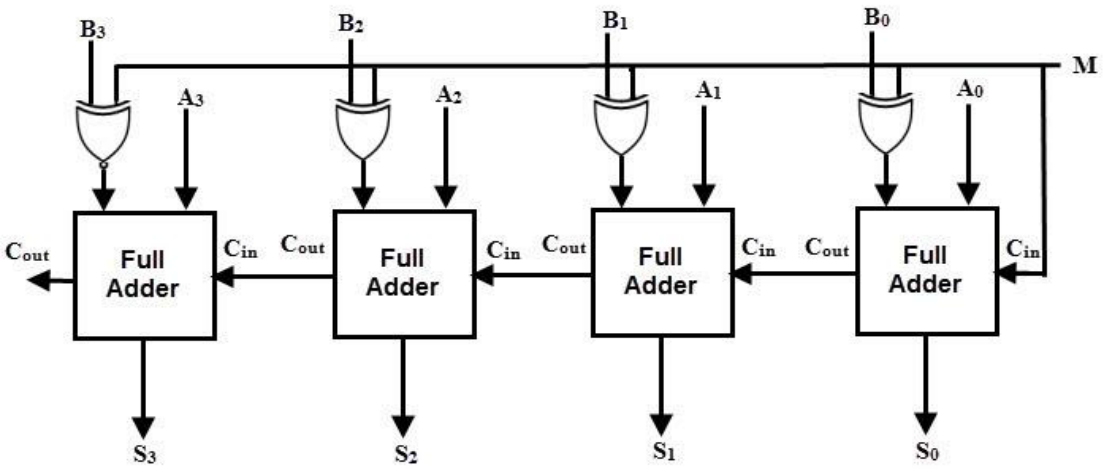


Figure II.15. Le circuit d'un additionneur soustracteur de 4 bits [11]

Tableau II.4. Table de vérité d'un inversement par XOR[19]

Add/Sub	B	S
0	0	0
0	1	1
1	0	1
1	1	0

II.9.3. L'architecture d'un SCM/MCM

Au niveau bit on élimine tous les additionneurs de 1 bit dont nous n'avons pas besoin. Ces additionneurs font la somme des bits avec des zéros, alors qu'il suffit juste de l'assignée à la sortie du signal de résultat. Cette opération ne consomme aucune ressource hardware. La figure II.13 représente l'architecture d'un exemple d'un multiple impaire au niveau bit par l'utilisation de cette solution.

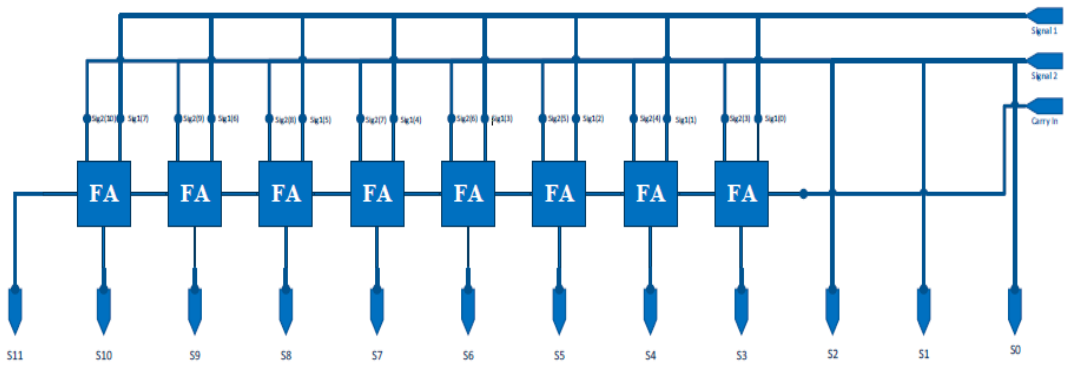


Figure II.16. L'architecture d'un SCM au niveau bit

On associe les circuits des multiples impairs afin de construire les circuits SCM/MCM motionnées dans les spécifications. La figure II.16 représente un exemple d'une architecture SCM/MCM.

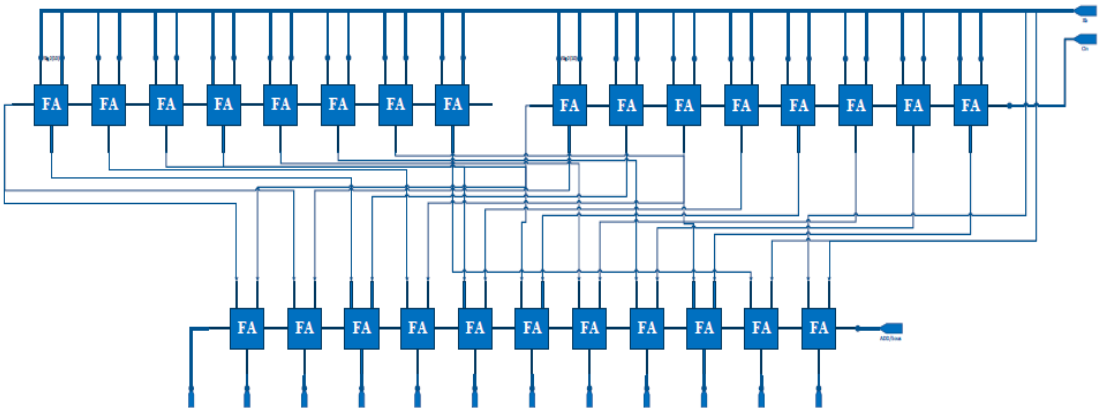


Figure II.17. L'architecture d'un exemple SCM/MCM

II.10. Conclusion

Dans ce chapitre nous avons introduit le problème de la multiplication par une constante simple et multiple SCM/MCM et nous avons présenté brièvement la nouvelle solution algorithmique radix-2r pour la résolution de ce problème.

Nous avons constaté que la solution au niveau bit offre une optimisation très importante en termes de ressources hardware qui influence directement la surface, le temps d'exécution, ainsi que l'énergie consommée.

Le chapitre qui suit sera dédié à l'utilisation de cette solution pour construire des filtres numériques RIF optimisé.

Chapitre III :

Application du SCM/MCM aux filtres RIF

III.1. Introduction

Comme exemple de LTI, nous présentons dans cette section le filtre numérique RIF qui est implémenté généralement dans des circuits de type ASIC en raison des exigences de conception de systèmes embarqués (vitesse et puissance).

Dans ce chapitre on utilise la solution SCM/MCM pour générer une description matérielle pour un filtre RIF optimisée au niveau de la surface, vitesse, et consommation de puissance.

III.2. Les plateformes de réalisation des filtres RIF

- Logique câblée (portes logiques, mémoires, …) par l'utilisation des FPGA/ASIC.
- Logique programmée (processeur de traitement du signal "DSP : Digital Signal Processing", microprocesseur "ordinateur") [10].

III.2.1. FPGA/ASIC

FPGA (Field Programmable Gate Arrays) est un circuit intégré personnalisable pour une application spécifique, il peut être programmé sue le terrain, c'est-à-dire configurable par l'utilisateur après la fabrication.

FPGA contient un ensemble de cellules programmables, chaque cellule peut exécuter une fonction sélectionnée parmi plusieurs possibles. L'interconnexion est également programmable.

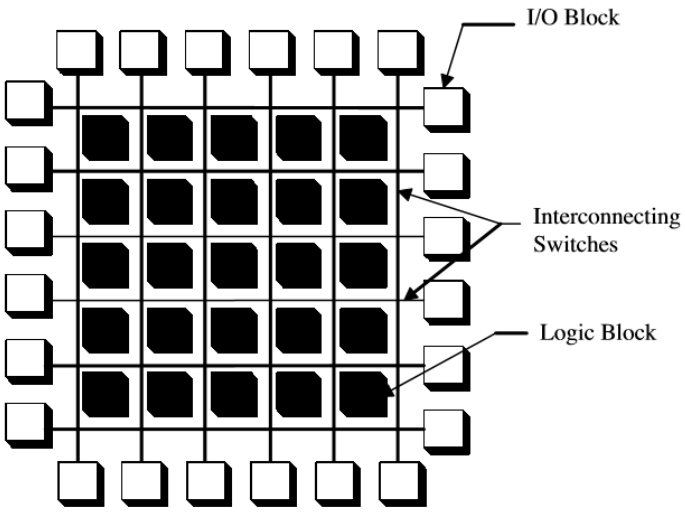


Figure III.1. La structure interne d'une FPGA[18]

ASIC *(Application Specific Integrated Circuits) sont des circuits intégrés personnalisés utilisés pour des tache de conception spécifique*.

III.2.1.1. Caractéristiques des FPGA/ASIC

- Temps de réponse très rapide dans les entrées/sorties
- Des capacités de calcul parallèles massives qui peuvent être exploitées en optimisant l'architecture
- Les multiples cellules MAC peuvent être implémentées sur une seule puce [].

III.2.2. DSP

Un DSP (Digital Signal Processor) est un processeur dont l'architecture est optimisée pour effectuer des calculs complexes en parallèle à chaque cycle d'horloge, mais aussi pour accéder très facilement à un grand nombre d'entrée/sortie (numérique ou analogique).

III.2.2.1. Caractéristiques des DSP

- Peu couteux
- Basés sur l'architecture RISC (Reduced Instruction Set Computer)

- ont des multiplieurs accumulateurs rapides
- Architecture de pipeline multi-stage

Afin de faire une comparaison de entre ces deux types de plateformes, et pour le même traitement d'image effectuée, le tableau suivant illustre les résultats de ce traitement :

Tableau III.1: le traitement d'image dans une FPGA et un DSP[18]

	FPGA	DSP
Le nombre des cycles d'horloge	164354cycles	10770432cycles
Temps d'exécution (fréquence d'horloge est 50 Mhz pour FPGA, 600 Mhz pour DSP)	3.2 ms	17.9 ms
Temps d'exécution/pixel	0.19 µs	1 µs
Utilisation matériels HW	742 éléments logiques	67.2 KB
Nombre de ligne de Code	132	429
L'énergie consommée	79.97 mW	540 mW

Il est clair que le traitement du signal à basse fréquence, à haute vitesse, à faible consommation d'énergie et à code court peut être mis en ouvre sur FPGA. Contrairement au DSP, le DSP est relativement élevé en termes d'énergie et de performances. Donc l'utilisation de FPGA/ASIC peut garantir des performances élevées à des coûts énergétiques inférieurs.

III.3. la structure d'un filtre RIF

La structure d'un filtre RIF nécessite un nombre de N multiplieurs de SCM (multiplication single constant) et de N-1 additionneurs et registres. Cependant, une observation attentive révèle deux formes principales :

- La taille de registre (*Sr*) dans la forme directe est fixe et égale à la longueur de bits l'entrée du filtre b*wi*. Dans la forme transposée, la taille de registre dépend de N, b*wi*, et de la longueur en bit des coefficients. Le terme *Sr* est calculé comme suit :

$$Sr = bwi + \log_2 \sum_{i=0}^{N-1} |h_i| \qquad \text{III.1}$$

- Le chemin critique de la forme transposée (un multiplieur et un additionneur) est plus court que celui de la forme directe (n multiplieur et [$\log_2$ N] additionneurs, en supposant que les additionneurs sont structurés en un arbre binaire), la figure III.2 l'architecture d'un filtre RIF à la forme transposée.

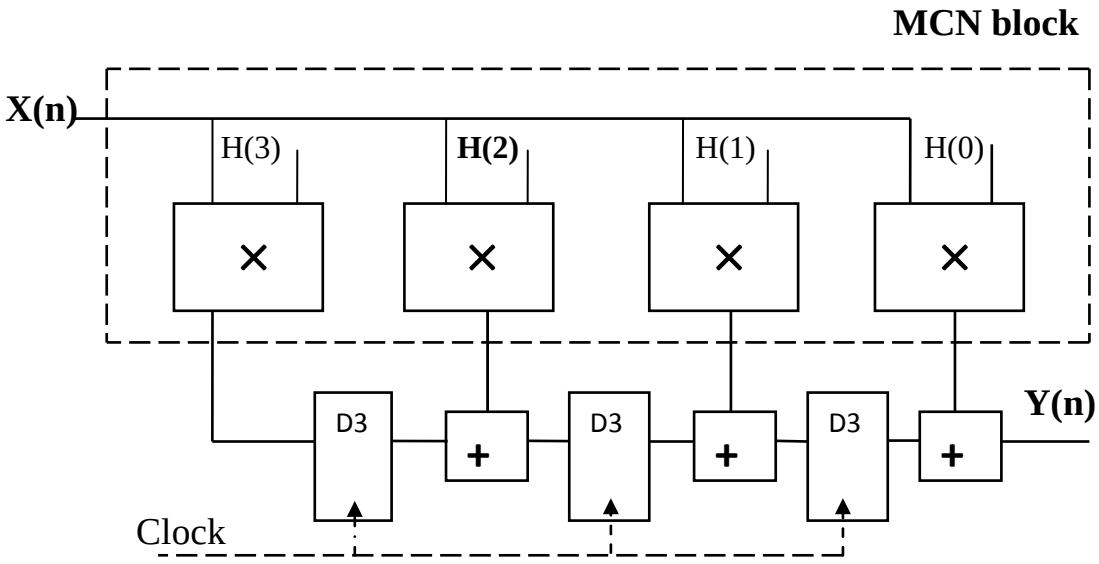


Figure III.2. L'architecture d'un filtre RIF

La conception matérielle de cette architecture est basée sur la stratégie d'utilisation de blocs MCM et d'additionneurs pour assurer la somme des produits. Nous utilisons ce que l'on appelle le pipelining pour réduire le chemin critique, augmenter la vitesse d'échantillonnage et réduire la consommation d'énergie.

III.4. Résultat et discussions

```
* Run for an

*** User giv

-710
327
505
582
398
-35
-499
-662
-266
699
1943
2987
3395
2987
1943
699
-266
-662
-499
-35
398
582
505
327
-710
```

Recodage

```
*** Radix-2^r MCM solution of the reduce

   3 = +1<<1 +1<<0
   5 = +1<<2 +1<<0
   7 = +1<<3 -1<<0
   9 = +1<<3 +1<<0
  11 = +9<<0 +1<<1
  13 = +9<<0 +1<<2
  35 = +3<<0 +1<<5
 133 = +5<<0 +1<<7
 199 = +7<<0 +3<<6
 291 = +3<<0 +9<<5
 327 = +7<<0 +5<<6
 331 = +11<<0 +5<<6
 355 = +3<<0 +11<<5
 499 = -13<<0 +1<<9
 505 = -7<<0 +1<<9
 699 = -5<<0 +11<<6
1943 = -9<<0 -3<<5 +1<<11
2987 = +11<<0 -3<<5 +3<<10
3395 = +3<<0 +5<<6 +3<<10
```

Figure III.3. Les coefficients de filtres ainsi que leur recodage en RADIX-2r[20]

Apres l'exécution de notre programme, il génère une description VHDL optimisé équivalant à partir de recodage précèdent (voire la figure III.4)

```
675     MAC_0 <= MAC_1 - SCM_0;
676     MAC_1 <= MAC_2 + SCM_1;
677     MAC_2 <= MAC_3 - SCM_2;
678     MAC_3 <= MAC_4 + SCM_3;
679     MAC_4 <= MAC_5 + SCM_4;
680     MAC_5 <= MAC_6 - SCM_5;
681     MAC_6 <= MAC_7 + SCM_6;
682     MAC_7 <= MAC_8 - SCM_7;
683     MAC_8 <= MAC_9 - SCM_8;
684     MAC_9 <= MAC_10 - SCM_9;
685     MAC_10 <= MAC_11 - SCM_10;
686     MAC_11 <= MAC_12 + SCM_11;
687     MAC_12 <= MAC_13 + SCM_12;
688     MAC_13 <= MAC_14 + SCM_13;
689     MAC_14 <= MAC_15 - SCM_14;
690     MAC_15 <= MAC_16 - SCM_15;
691     MAC_16 <= MAC_17 - SCM_16;
692     MAC_17 <= MAC_18 - SCM_17;
693     MAC_18 <= MAC_19 + SCM_18;
694     MAC_19 <= MAC_20 - SCM_19;
695     MAC_20 <= MAC_21 + SCM_20;
696     MAC_21 <= MAC_22 + SCM_21;
697     MAC_22 <= MAC_23 - SCM_22;
698     MAC_23 <= MAC_24 + SCM_23;
699     MAC_24 <= - SCM_24;
700     MCM_out <= MAC_0;
```

Figure III.4. Une partie de la description VHDL de filtre

Pour la simulation de ce filtre sur modelsim, on utilise un fichier (.do), qui génère des stimulis d'entrée du filtre, le signal de reset, ainsi que le signal d'horloge, comme montrée à la figure III.5.

```
force xb 00000001 0
force reset 0 0,1 10,0 20
force clk 1 0,0 10 -repeat 20
force cin 0 0
run 300ns
```

Figure III.5. La génération des signaux d'entré pour la simulation

Comme il est apparaît à la figure III.6, les résultats de simulation du filtre RIF apparaissent après un certain nombre d'oscillation d'horloge, la sortie du filtre sera égale à la

somme des produit Ci x X, à cause du pipeline du système qui joue le rôle de délai. La partie inconnue dans ce signal (en rouge) est fait parce que les registres n'étaient pas encore remplis par des valeurs, donc on applique une impulsion de reset qui permet la réinitialisation des registres à 0.

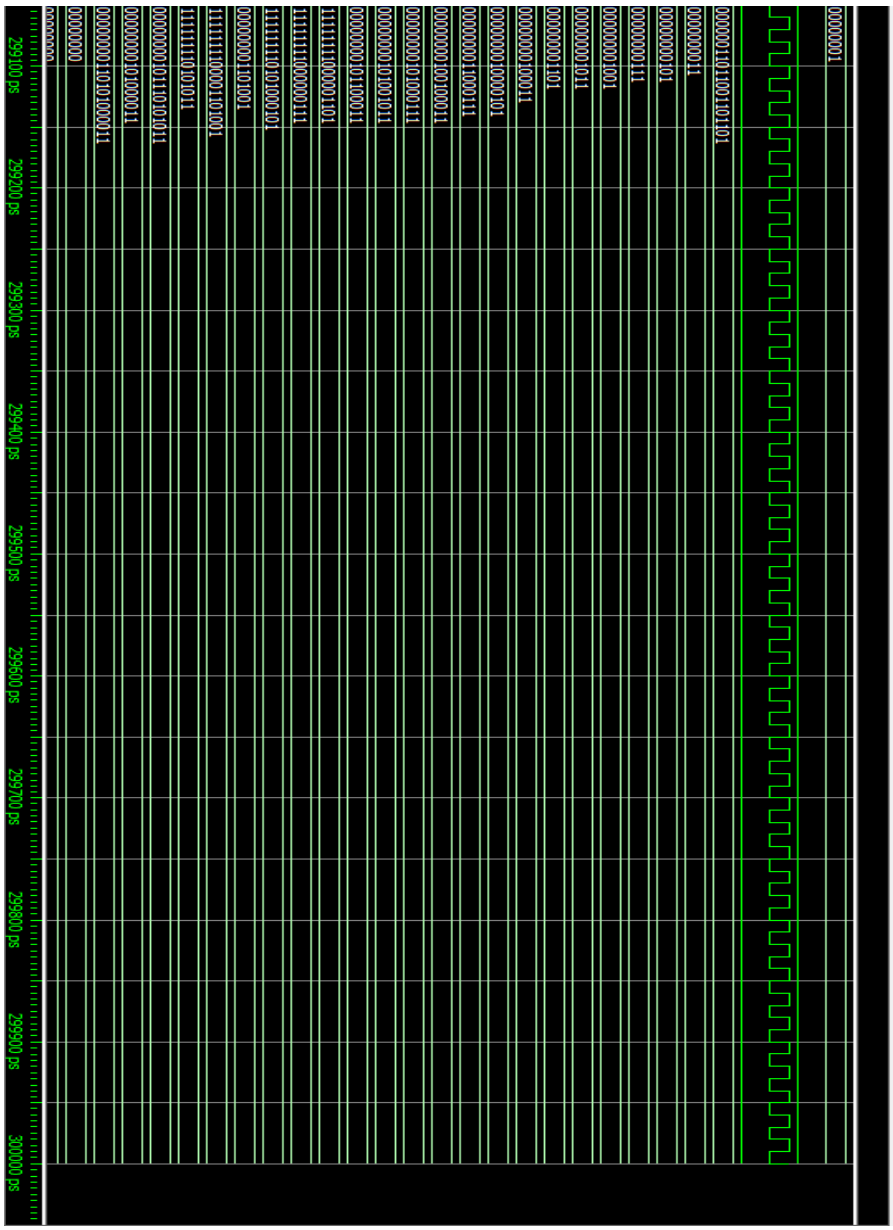


Figure III.5. Les résultats de simulation de filtre FIR

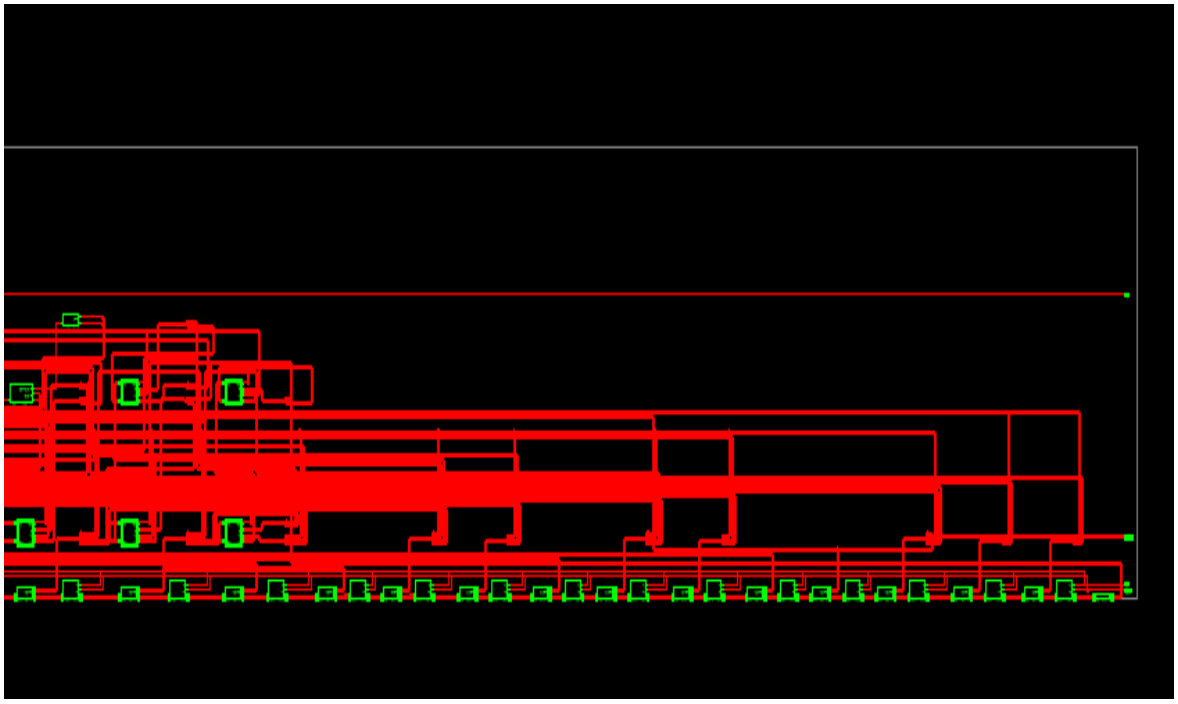


Figure III.6. L'architecture de filtre RIF optimisé

III.5. Conclusion

Dans ce chapitre on a appliqué la solution de résolution des problèmes SCM/MCM au niveau d'additionneur de 1 bit, sur les filtres à réponse impulsionnelle finie RIF. Cette application permet de générer une description VHDL très optimisée bit par bit pour tous les filtres RIF. Elle offre une optimisation importante au niveau des ressources consommées dans les blocs de multiplication, avec une architecture interne optimisée basée sur des d'additionneur de 1 bit.

Conclusion générale

Conclusion générale

L'objectif fixé au lancement de ce travail était de trouver une implémentation hardware optimale des systèmes linéaires invariants dans le temps (LTI) qui permet de satisfaire les spécifications des systèmes embarqués en termes de vitesse, surface, et de consommation de puissance. Une telle implémentation passe par le développement d'heuristiques efficace de la multiplication par simple/multiple constantes SCM/MCM en utilisant le calcul exact, et aussi de choisir les techniques d'optimisation de l'implémentation les plus appropriées, selon que la cible finale soit ASIC ou FPGA.

Dans ce travail, une heuristique MCM (RADIX-2r) basée sur l'arithmétique Radix-2r a été développée aussi bien au niveau bloc d'additionneurs qu'un au niveau bit (Full Adder). Cette heuristique présente l'avantage par rapport à l'état de l'art d'être totalement prédictible en nombre maximum d'additionneurs. Elle requiert une complexité de calcul sous-linéaire et une profondeur en additionneurs (chemin critique) presque optimale. Ces limites sont prouvées d'une façon exacte, y compris pour le nombre moyen d'additionneurs. Elles constituent les seules limites analytiques exactes connues jusqu'à présent pour le problème du MCM. Outre les limites analytiques uniques connues jusqu'ici pour le MCM au niveau bloc d'additionneurs. Une nouvelle limite au niveau bit d'additionneurs a été introduite.

Nous avons prouvé en circuit ASIC de filtre de type RIF que Radix-2r possède les meilleurs résultats en vitesse, puissance, en particulier dans le bloc MCM de haute complexité. De plus, la complexité d'exécution sous-linéaire de Radix-2r signifie qu'il n'y a pas de limitation à la taille de problème a résoudre.

Enfin, la nouvelle arithmétique Radix-2r peut être avantageusement appliquée à d'autres domaines numériques, tels qu'en traitement numérique du signal (DSP), traitement d'image, télécommunications et le cryptage. Une idée consiste à appliquer la nouvelle heuristique SMC/MCM aux algorithmes de chiffrement RSA pour cibler des longues clés de chiffrement (plus de 4096 bits).

Bibliographie

[1] S, olivier, “traitement numérique de signal,”. Electronique et informatique industriels, vol. EC-10, No. 3, pp. 389–400, Septembre 2014.

[2] F, BELMAHDI, application de filtre de kalman pour le debrouitage des signaux, UMMTO, 14/12/2015,p 10.

[3] Y. Voronenko and M. Püschel, “Multiplierless Multiple Constant Multiplication,” ACM Trans. on Algorithms (TALG), vol. 3, No. 2, article 11, pp. 1-38, May 2007.

[4] A. K. Oudjida, N. Chaillet, M. L. Berrandjia «RADIX-2r Arithmetic for Multiplication by a Constant: Further Results and Improvements » *IEEE Trans. on Circuits and Systems II: Express Brief,* vol. 62, pp. 372-376, Avril 2015.

[5] A. K. Oudjida, N. Chaillet «RADIX-2r Arithmetic for Multiplication by a Constant » *IEEE Trans. on Circuits and Systems II: Express Brief,* vol. 61, pp. 349-353, May 2014.

[6] K.C.Chang, "Digital system design with VHDL and synthesis : An integrated appraoch, Los Alamitos", California: IEEE Computer society, 1999, 551p, ISBN 0-7695-0023-4.

[7] [7] j. Charles H. Roth, "Digital system design using VHDL, Boston: PWS Publishing" company, 1997, 476p, ISBN 0-534-95099-X.

[8] J. Thong and N. Nicolici, “An optimal and practical approach to single constant multiplication,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 9, pp. 1373–1386, September 2011.

[9] A. K. Oudjida, «Binary Arithmetic for Finite-Word-Length Linear Controllers: MEMS Applications » Université de Franche-Comté, Besançon, 2015.

[10] J. Thong, "New Algorithm for Constant Coefficient Multiplication in Custom Hardware," Master Thesis, No 4292, McMaster University, Hamilton, Ontario, Canada, October 2009.

[11]M. Labandji, Développement d’une Solution au Niveau "Bit" au Problème de la Multiplication par une Constante. université AMOB, 2017, p 25.

[12]A. K. Oudjida, A. Liacha, M. Bakiri, N. Chaillet,” Multiple Constant Multiplication Algorithm for High-Speed and Low-Power Design” *IEEE Trans on Circuits and Systems II: Express Briefs,* vol. 63, n° %12, pp. 176 - 180, Fevrier 2016.

[13] H. Sam, and A. Gupta, “A Generalized Multibit Recoding of Two’s Complement BinaryNumbers and its Proof with Application in Multiplier Implementation,” IEEE Trans. on Computers, vol. 39, N° 8, August 1990.

[14] J. Thong and N. Nicolici, “An optimal and practical approach to single constant multiplication,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 9, pp. 1373–1386, September 2011.

[15] A.Liacha, A. K. Oudjida, F. Ferguene, M. Bakiri, M. L. Berrandjia «Design of High-Speed, Low-Power, and Area-Efficient FIR Filters» *Circuits, Devices and Systems*, n° %15, DOI 10.1049/iet-cds.2017.0058 , 2017.

[16] «The leading operating system for PCs, IoT devices, servers and the cloud _ Ubuntu,» Canonical Ltd. Ubuntu, 2017. [En ligne]. Available: https://www.ubuntu.com/. [Accès le 6 septembre 2017].

[17] C. f. d. d'Ubuntu, «gedit - Documentation Ubuntu Francophone,» Communauté francophone d'utilisateurs d'Ubuntu, 2017. [En ligne]. Available: https://doc.ubuntufr. org/gedit. [Accès le 6 septembre 2017].

[18] C. f. d. d'Ubuntu, «gcc - Documentation Ubuntu Francophone,» Communauté francophone d'utilisateurs d'Ubuntu, 2017. [En ligne]. Available: https://doc.ubuntufr. org/gcc. [Accès le 6 septembre 2017].

[19] «Binary Adder and Subtractor,» ELECTRONICS HUB, 29 juin 2015. [En ligne]. Available: http://www.electronicshub.org/binary-adder-and-subtractor/. [Accès le 7 septembre 2017].

[20] CDTA « Multiple Constant Multiplication Algorithm », © 2015. [En ligne]. Available : http://www.cdta.dz/products/mcm/. [Accés le 16 septembre 2017].