الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière : Télécommunication
Spécialité : Systèmes des télécommunications

Présenté par

Abdi Abderraouf

&

Tigrine Amir

Étude et implémentation FPGA du filtrage FIR avec réduction simultanée de l'utilisation logique de DSP et LUT

Promoteur : Mountassar MAAMOUN

Co-promoteur : Abdelhalim BENBELKACEM

Année Universitaire 2023-2024

Remerciements

Tout d'abord, nous tenons à remercier Dieu le tout-puissant et miséricordieux, qui nous a accordé la force et la patience nécessaires pour mener à bien ce modeste travail.

Je tiens à exprimer ma sincère reconnaissance à mon encadreur Mr Mountassar MAAMOUN pour sa précieuse guidance, son expertise et son soutien tout au long de la réalisation de mon mémoire de fin d'études.

Nous adressons nos sincères remerciements aux membres du jury pour l'honneur et l'intérêt qu'ils ont portés à notre travail en acceptant d'examiner et d'évaluer notre mémoire.

Un grand merci à tous les enseignants du département d'électronique, en particulier aux professeurs de l'option "Télécommunication", ainsi qu'à tous les autres professeurs qui nous ont guidés tout au long de notre parcours universitaire, ainsi qu'à toutes les personnes qui, par leurs paroles, leurs écrits, leurs conseils et leurs critiques, ont contribué à enrichir notre réflexion.

Enfin, nous exprimons notre profonde gratitude envers nos parents pour leur soutien indéfectible, leur patience et leur contribution tout au long de cette aventure. Nous remercions également tous nos proches et amis pour leur encouragement constant tout au long de ce processus.

Merci à tous, à nos parents, à nos frères et sœurs, à toute notre famille et à nos amis pour leur soutien inestimable.

Dédicace

Les années ont passé très vite, avec leur douceur et leur amertume, avec leur fatigue et joie. Je suis très fier d'où je suis et où je suis maintenant. Ce n'était pas facile, mais je l'ai fait Après 18 ans de labeur, me voilà aujourd'hui, je le conclus par cette dédicace. Vraiment, il est dommage que nous soyons arrivés au bout, mais tout commencement a une fin.

Avant tout, je dédie cette note à mes parents bien-aimés "brahim" et "Leila". Je vous remercie pour votre soutien constant et votre patience, vous m'avez accompagné tout au long de ces années. Il n'y a rien qui puisse exprimer ma gratitude et mon appréciation envers vous. Je ferai de mon mieux pour toujours vous rendre fiers et la tête haute. Je prie Dieu Tout-Puissant pour qu'Il vous protège et vous accorde une santé et un bonheur éternels.

Je tiens également à remercier mes frères "Soheib", "Iman", "Mohammed" et "Abderrahim". Je vous souhaite le bonheur, le succès et la réalisation de vos objectifs, si Dieu le veut.

Ma reconnaissance va également à toute ma famille, oncles, tantes, cousins et cousines, qui attendent mon succès. Je vous souhaite le bonheur.

Enfin, je remercie tous mes amis, en particulier Tigrine Amir, Bekhouche Abdallah, Mesbah Zoheir et Abdelhakim. Je vous souhaite une vie heureuse et une amitié éternelle, si Dieu le veut.

ABDI ABDERRAOUF

Dédicace

À ma mère "Salima" qui a toujours été là pour moi, qui m'a soutenu et encouragé dans mes rêves, et qui a été mon refuge dans les moments de difficulté. Je vous remercie pour tout, mère, et je vous dédie ce mémoire avec amour et gratitude.

À mes frères "Hamza", "Riadh" et "Nacereddine", qui ont partagé mes joies et mes peines, qui m'ont inspiré et motivé, et qui ont été mes meilleurs amis. Je vous remercie pour votre présence dans ma vie et pour les souvenirs inoubliables que nous avons partagés.

À mes amis "Benkiar Mehdi", "Bekhouche Abdallah", "Mesbah Zoheir" et "Abdi Abderraouf", qui ont été mes compagnons de route, mes confidents et mes soutiens. Je vous remercie pour vos encouragements, vos conseils et vos rires. Vous avez été une source de joie et de motivation pour moi, et je vous dédie ce mémoire avec amour et reconnaissance. Ce texte est un exemple de dédicace qui exprime votre gratitude et votre amour envers vos proches et vos amis. Vous pouvez l'adapter à votre style et à votre ton pour qu'il soit plus personnel et plus authentique.

TIGRINE AMIR

ملخص: يقدم هذا العمل تصميم مرشح FIR عالي المستوى تم تحسينه لمنصات FPGA، مما يقلل من استخدام موارد DSP و LUT. يسمح بتحديث معاملات المرشح في الوقت الفعلي من خلال الاستفادة من سرعة وبنية FPGA. يدير التصميم اختلافات تردد أخذ العينات للحصول على وقت حسابي إضافي ويستغل هياكل LUT-SR في اختيار عينة الأنابيب والمدخلات. تستخدم BRAM FPGA لتخزين وتحديث المعاملات وشرائح DSP لمعالجة المخرجات. وحدة التحكم تتزامن مع تشغيل BRAM و LUT multiplexers.

كلمات المفاتيح:FPGA على BRAM ; FPGA على LUT ; FPGA على DSP

Résumé :

Ce travail présente une conception de filtre FIR d'ordre élevé optimisée pour les plateformes FPGA, réduisant l'utilisation des ressources DSP et LUT. Il permet la mise à jour en temps réel des coefficients du filtre en tirant parti de la vitesse et de l'architecture du FPGA. La conception gère les différences de fréquence d'échantillonnage pour obtenir du temps de calcul supplémentaire et exploite les structures LUT-SR pour le pipelining et la sélection des échantillons d'entrée. Les BRAM FPGA sont utilisées pour stocker et mettre à jour les coefficients, et les tranches DSP pour traiter les sorties. Une unité de contrôle synchronise le fonctionnement des BRAM et des multiplexeurs LUT.

Mots clés :DSP sur FPGA; LUT sur FPGA; BRAM sur FPGA.

Abstract :

This work presents an optimized high-order FIR filter design for FPGA platforms, reducing DSP and LUT resource usage and enabling real-time coefficient updates. It leverages FPGA speed and architecture to manage input sampling rates, gaining extra computation time. The design fully utilizes FPGA's LUT-SR structures for pipelining and input sample selection, and uses block RAMs to store and update coefficients. DSP slices process BRAM outputs and multiplexers. A single control unit synchronizes BRAM addressing with LUT multiplexer selection, ensuring simultaneous operation

.**Keywords :**DSP on FPGA ; LUT on FPGA ; BRAM on FPGA.

Listes des acronyms et abréviations

FPGA: Field Programmable Gate Array.

FIR: Finite Impulse Response.

RAM: Random Access Memory.

BRAM: Block RAM.

LUT: Look-Up Table.

DSP: Digital Signal Processor.

LUT-SR: Look-Up Table - Shift Register.

FIFO: First In, First Out.

FSM: Finite State Machine.

MUX: Multiplexer.

RMU: Remote Monitoring Unit.

CTR: Counter.

MAC: Multiplication-Accumulation.

ASICs: Application-Specific Integrated Circuits.

VHSIC: Very High Speed Integrated Circuits.

HDL: Hardware Description Language.

VHDL : Very High Speed Integrated Circuits Hardware Description Language.

MATLAB : Matrix Laboratory .

CLK : Clock .

CE : Clock Enable .

CLR : Clear .

Table des matières

Liste des figures

Introduction générale

Les filtres à réponse impulsionnelle finie (FIR) sont des composants essentiels dans les systèmes de traitement numérique du signal (DSP) , utilisés dans une multitude d'applications telles que les communications, l'audio, et le traitement d'image. Que ce soit avec des coefficients fixes ou reconfigurables, les filtres FIR jouent un rôle crucial dans la réalisation de performances optimales en termes de filtrage de signaux. Avec les progrès considérables des applications multimédias et mobiles, la demande de systèmes DSP à haute vitesse et à complexité réduite n'a cessé de croître. Par conséquent, il est impératif de développer des architectures de filtres FIR d'ordre élevé, capables de reprogrammation en temps réel, pour répondre aux exigences des systèmes mobiles et de communication de nouvelle génération.

Ce mémoire propose une structure efficace de filtre FIR d'ordre élevé reconfigurable en temps réel, avec une utilisation extrêmement réduite de la logique FPGA. Pour atteindre cet objectif, une approche innovante utilisant la structure spéciale de la LUT-SR des FPGA est mise en œuvre. Cette structure, composée d'un registre à décalage et d'un multiplexeur, est entièrement exploitée pour le pipelining et la sélection des échantillons d'entrée. Deux fréquences d'horloge distinctes sont utilisées pour alimenter les registres à décalage et le sélecteur de multiplexeur, optimisant ainsi le processus de filtrage.

Différentes structures de filtre FIR à virgule fixe basées sur FPGA peuvent être citées dans la littérature . Les structures symétriques basées sur FPGA offrent une fréquence d'échantillonnage d'entrée élevée avec une utilisation insignifiante des LUT, à côté d'une utilisation importante des tranches DSP du FPGA. Une réduction

supplémentaire de l'utilisation des LUT a été présentée pour les architectures basées sur FPGA qui prennent en charge la technique d'extension de signal symétrique . Cette technique exploite la longueur variable de la table de correspondance FPGA Shift-Register (LUT-SR) pour atteindre la réduction de l'utilisation des LUT, sans inclure aucune optimisation de tranche DSP.

Les BRAMs (Block RAMs) des FPGA sont employées pour stocker et mettre à jour les coefficients reconfigurables du filtre, assurant une flexibilité et une adaptabilité maximales. Les slices DSP de la FPGA sont associées pour calculer les données de sortie des BRAMs et des multiplexeurs. Grâce à cette approche, nous pouvons réduire l'utilisation des slices DSP à un seul, réalisant ainsi une implémentation extrêmement efficace en termes de ressources.

Cette étude est structurée en trois chapitres principaux : Le premier chapitre présente une revue détaillée des architectures utilisées pour l'implémentation des filtres FIR, en mettant l'accent sur les avantages et les inconvénients de chaque méthode. Le deuxième chapitre se concentre sur l'approche proposée, détaillant la structure de la LUT-SR et l'utilisation des BRAMs et des slices DSP pour optimiser l'implémentation des filtres FIR sur FPGA. Enfin, le troisième chapitre décrit les étapes de la mise en œuvre pratique, y compris les tests et les simulations, et présente les résultats obtenus, permettant de valider l'efficacité de la solution proposée.

Chapitre 1 : Architectures utilisées pour l'implémentation des filtres FIR

1.1 Introduction

Les filtres à réponse impulsionnelle finie (FIR) jouent un rôle crucial dans le traitement du signal numérique.

L'implantation efficace d'un filtre FIR est essentielle pour optimiser ses performances. Plusieurs architectures existent, chacune présentant des avantages et des inconvénients spécifiques. Ce chapitre offre une introduction à ces architectures, permettant de comprendre leurs principes de fonctionnement et leurs implications pour la conception de filtres FIR.

En plus des tranches de LUT, les FPGA récents incluent des BRAMs et des tranches DSP. Les opérations d'écriture et de lecture des BRAMs sont synchrones. Les deux ports sont symétriques et totalement indépendants. Pendant une opération d'écriture, les données de sortie de la mémoire peuvent être configurées pour rester inchangées. La tranche DSP inclut principalement un pré-additionneur sélectionnable suivi d'un multiplicateur à deux entrées, de multiplexeurs et d'un additionneur à deux entrées. Dans cette section, nous discutons des limites de réduction de l'utilisation de la logique des structures basées sur FPGA. De nombreuses implémentations de structures de filtre FIR peuvent être observées dans la littérature. Cependant, elles peuvent être organisées en quatre structures principales : les structures directes et transposées, les structures symétriques et les structures basées sur l'arithmétique distribuée.

1.2 Définition FPGA

Les FPGA (Field Programmable Gate Arrays) sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court. Le progrès de ces technologies permet de faire des composants toujours plus rapides et à plus haute intégration, ce qui permet de programmer des applications importantes. Cette technologie permet d'implanter un grand nombre d'applications et offre une solution d'implantation matérielle à faible coût pour des compagnies de taille modeste pour qui, le coût de développement d'un circuit intégré spécifique implique un trop lourd investissement.[1]

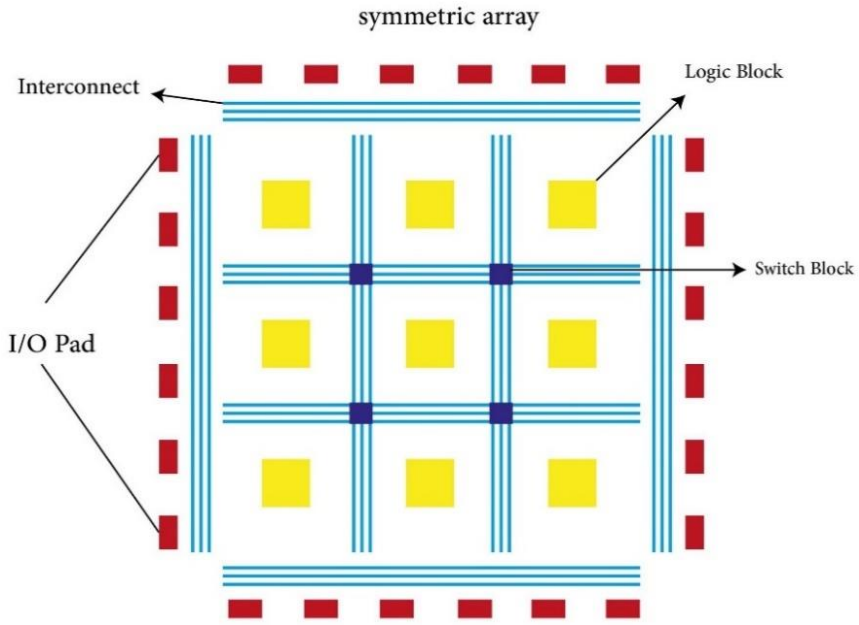

Figure 1.1 Structure Générale d'un FPGA

1.3 Les filtres numériques

Il existe deux catégories des filtres numériques linéaires filtre FIR et filtre IIR. Selon la durée de la réponse impulsionnelle et l'architecture que nous utilisons, que nous devons utiliser le filtre FIR.[3]

1.4 Filtre à réponse impulsionnelle finie (fIR)

Les filtres numériques à réponse impulsionnelle finie (FIR), également connus sous le nom de filtres non récursifs, se distinguent par leur réponse impulsionnelle basée sur un nombre fini d'échantillons d'un signal. Malgré cette limitation, ils occupent une place prépondérante dans le traitement du signal en raison de leurs propriétés uniques et de leur simplicité de mise en œuvre.[3]ils sont cependant très largement utilisés car ils possèdent des propriétés uniques (phase, linéarité, stabilité, flexibilité), ces filtres sont aussi connus par nom filtres non récursif.[8]

1.5 Structures directes

Les structures directes dans les FPGA sont des architectures qui permettent une interconnexion directe entre les différents blocs logiques du circuit, offrant ainsi une programmation efficace et rapide des filtres FIR. Ces structures exploitent un haut niveau de parallélisme, ce qui permet d'implémenter chaque étage du filtre dans des blocs logiques distincts.[6]

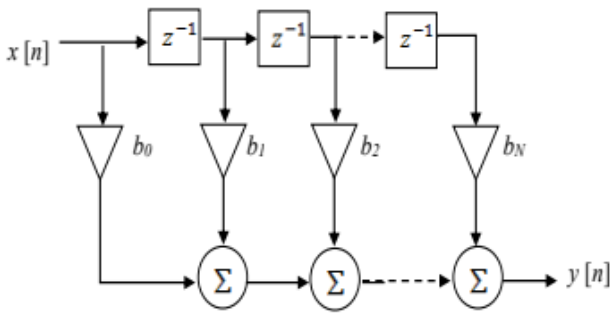

Figure 1.2 Structure directd'un filtre FIR.[6]

De plus, les structures directes assurent une utilisation optimale des ressources logiques du FPGA en évitant le gaspillage lié à un routage complexe. Grâce à la flexibilité

de conception offerte par la programmabilité des FPGA, les structures directes peuvent être reconfigurées facilement pour implémenter différents filtres FIR, les rendant ainsi très adaptées à cette tâche.

1.5.1 Principe de fonctionnement

L'équation de filtrage d'un filtre FIR d'ordre N s'exprime comme suit :

$$y(n) = \sum_{i=0}^{N-1} \left(hi \times x(n - i) \right) \quad (1.1)$$

où :

- $y(n)$est la sortie du filtre à l'instant n
- $x(n)$est l'entrée du filtre à l'instant n
- hiest le coefficient du filtre (i de 0 à N)

La structure directe traduit directement cette équation en une architecture matérielle composée de :

- **N multiplicateurs:** un pour chaque coefficient du filtre
- **N retards:** pour stocker les N derniers échantillons d'entrée

1.5.2 Avantages des structures directes

Les structures directes présentent plusieurs avantages :

A. **Interconnexions directes :** Les blocs logiques du FPGA sont reliés directement entre eux, sans passer par un réseau d'interconnexions programmable complexe. Cela permet une implémentation plus efficace et rapide des filtres FIR.

B. **Parallélisme élevé :** La structure directe permet d'exploiter le parallélisme inhérent aux filtres FIR, en implémentant chaque étage du filtre dans des blocs logiques distincts. Cela permet d'obtenir de hauts débits de données.

C. **Utilisation optimale des ressources :** Les structures directes permettent d'utiliser de manière optimale les ressources logiques du FPGA, en évitant le gaspillage lié à un routage complexe.

D. **Flexibilité de conception :** La programmabilité des FPGA permet de reconfigurer facilement les structures directes pour implémenter différents filtres FIR.

1.5.3 Limites de l'implémentation FPGA des structures directes

L'implémentation FPGA, utilisant la forme directe de (1), requiert des multiplicateurs, ce qui signifie des tranches DSP FPGA. Par conséquent, l'ordre maximal possible du filtre FIR est uniquement contraint par les tranches DSP FPGA disponibles, en plus de la non-utilisation des BRAMs et de l'utilisation insignifiante des tranches LUT. De plus, les FPGA à faible coût incluent un nombre relativement limité de tranches DSP et de BRAMs. Cette contrainte, ainsi que le déséquilibre significatif de l'utilisation de la logique, indiquent le chômage de ces structures pour l'implémentation FPGA à faible coût de filtres FIR d'ordre élevé.[7,8]

1.6 Structures transposées

Les structures transposées constituent une alternative aux structures directes pour l'implantation des filtres FIR. Elles reposent sur une transformation mathématique de l'équation de filtrage, permettant de réduire le nombre de multiplications nécessaires tout en conservant les mêmes propriétés de filtrage.[6]

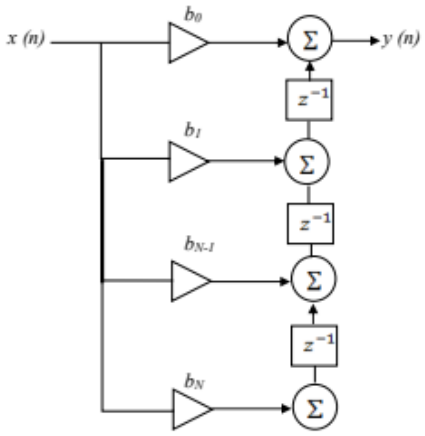

Figure 1.3 Structure transposée d'un filtre FIR.[6]

1.6.1 Principe de fonctionnement

L'équation de filtrage d'un filtre FIR d'ordre$N$ s'exprime comme suit :

$$y(n) = \sum_{i=0}^{N+1} (h'(i) \times x'(n + i)) \qquad (1.2)$$

où :

- $h'(i)$est le coefficient transposé du filtre à l'instant k .
- $x'(n)$est l'entrée transposée du filtre à l'instant n .

Les structures transposées peuvent être plus efficaces que les structures directes en termes de nombre de multiplications, mais elles nécessitent un prétraitement et un post-traitement des données.

1.6.2 Avantages des structures transposées

Les structures transposées présentent plusieurs avantages par rapport aux structures directes :

a) **Meilleure efficacité en termes de précision :** Moins sensibles aux erreurs d'arrondi dues aux multiplications répétées
b) **Optimisation des calculs :** Les structures transposées peuvent offrir une optimisation des calculs en réorganisant les opérations de manière plus efficace, ce qui peut conduire à des performances améliorées en termes de vitesse de traitement2.
c) **Gestion de la mémoire :** En modifiant la disposition des calculs, les structures transposées peuvent permettre une gestion plus efficace de la mémoire, réduisant ainsi les besoins en termes de ressources et améliorant potentiellement l'efficacité énergétique2.
d) **Flexibilité de conception :** Les structures transposées offrent une alternative à la structure directe, ce qui peut permettre aux concepteurs de choisir l'approche la mieux adaptée à leurs besoins spécifiques en termes de performances, de ressources et de contraintes de conception.

1.6.3 Inconvénients des structures transposées

Malgré leurs avantages, les structures transposées présentent également des inconvénients :

a) **Complexité accrue de la conception :** La transformation mathématique et la réorganisation des calculs rendent la conception plus complexe
b) **Augmentation de la latence :** Le traitement des données peut être légèrement plus lent en raison de la réorganisation des opérations

1.6.4 Implémentation FPGA des structures transposées

L'implémentation FPGA, utilisant la forme transposèe requiert des multiplicateurs, ce qui signifie des tranches DSP FPGA. Par conséquent, l'ordre maximal possible du filtre FIR est uniquement contraint par les tranches DSP FPGA disponibles, en plus de la non-utilisation des BRAMs et de l'utilisation insignifiante des tranches LUT. De plus, les FPGA à faible coût incluent un nombre relativement limité de tranches DSP et de BRAMs. Cette contrainte, ainsi que le déséquilibre significatif de l'utilisation de la logique, indiquent le chômage de ces structures pour l'implémentation FPGA à faible coût de filtres FIR d'ordre élevé.[7,8]

1.7 Structures symétriques

Les structures symétriques constituent une optimisation particulière des structures directes applicable uniquement aux filtres FIR dont les coefficients présentent une symétrie paire ou impaire autour d'un point central. Elles permettent d'exploiter cette symétrie pour réduire davantage le nombre d'opérations arithmétiques nécessaires à l'implantation du filtre.[9,6]

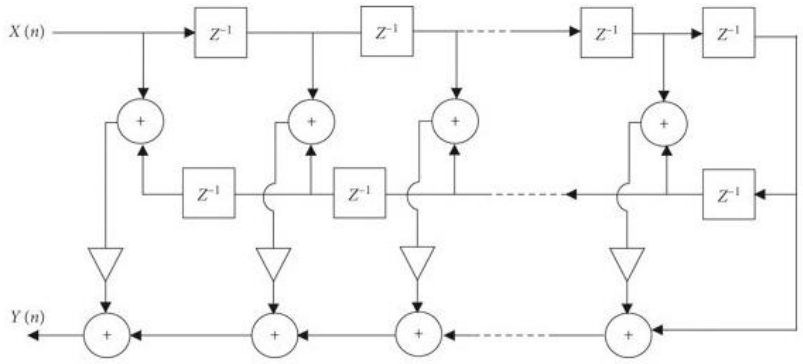

Figure 1.4 Structure symétrique d'un filtre FIR.

1.7.1 Principe de fonctionnement

Les filtres FIR symétriques possèdent des coefficients qui satisfont l'une des conditions suivantes :

Symétrie paire : $hi = h(N-i)$ pour tout i de 0 à $N/2$.

Symétrie impaire : $hi = -h(N-i)$ pour tout i de 0 à $N/2$.

En exploitant cette symétrie, il est possible de réduire le nombre de multiplications et d'additions nécessaires au filtrage. Pour les filtres à symétrie paire, on peut calculer uniquement la moitié des produits et effectuer des additions ou des soustractions pour obtenir la sortie complète. Pour les filtres à symétrie impaire, les produits impliquent des multiplications par -1, ce qui peut être optimisé en utilisant des techniques spécifiques.

1.7.2 Avantages des structures symétriques

Les structures symétriques présentent des avantages considérables par rapport aux structures directes :

a) **Réduction significative du nombre de multiplications et d'additions :** Permet d'améliorer considérablement l'efficacité du filtrage en termes de ressources matérielles et d'énergie.
b) **Gain important en termes de performance et de précision :** Moins sensibles aux erreurs d'arrondi dues à la réduction du nombre d'opérations.

c) **Simplicité de codage :** Les structures symétriques nécessitent moins d'informations spécifiques pour être codées génétiquement , elles sont donc plus susceptibles d'apparaître sous forme de variations phénotypiques par le biais de mutations aléatoires , cela explique en partie la prévalence des formes symétriques dans la nature.

1.7.3 Inconvénients des structures symétriques

Les structures symétriques présentent un inconvénient majeur :

Applicabilité restreinte : Ne sont applicables qu'aux filtres FIR dont les coefficients présentent une symétrie paire ou impaire.

1.7.4 Implémentation FPGA des structures symétriques

Pour l'implémentation d'un filtre FIR d'ordre élevé sur des tranches DSP FPGA, le meilleur choix est la conception de filtres symétriques. [10]

- **Coefficients FIR symétriques**

Par conséquent, les coefficients FIR de la structure à taps doivent respecter la forme symétrique, où f(2M – N) avec N pair. Ainsi, h(i) peut être réorganisé en utilisant l'expression suivante :

$$h(i) = h(2M - 1 - i) \qquad (1.3)$$

Calcul de la sortie du filtre FIR symétrique.

Par la suite, l'expression de la sortie y(n) du filtre FIR symétrique peut être donnée par:

$$y(n) = \sum_{i=0}^{M-1} h(i)[x(n-i) + x(n-2M+1+i)] \qquad (1.4)$$

- **Avantages de l'implémentation symétrique**

En utilisant l'équation (1.5), on peut conclure que le nombre de multiplications est fondamentalement divisé par deux . Cependant, le nombre d'additions n'est pas réduit. Selon la configuration des tranches DSP FPGA, les additionneurs peuvent être partitionnés en deux parties : la partie pré-additionneur et la partie additionneur. Par conséquent, l'utilisation des tranches DSP est rationnellement divisée par deux. Cette

implémentation représente un choix supérieur par rapport aux structures directes et transposées. Néanmoins, l'ordre de filtre maximal possible de la structure basée sur FPGA reste limité par les tranches DSP FPGA disponibles, en plus de la non-utilisation des BRAMs.[10]

1.8 Structures basées sur l'arithmétique distribuée

Les structures basées sur l'arithmétique distribuée s'attaquent aux limitations des structures directes et transposées en exploitant le parallélisme au niveau des bits pour accélérer les calculs de multiplication et d'addition. Elles décomposent les opérations arithmétiques sur des mots entiers en opérations sur des bits individuels, permettant une exécution simultanée sur plusieurs unités de traitement.[11]

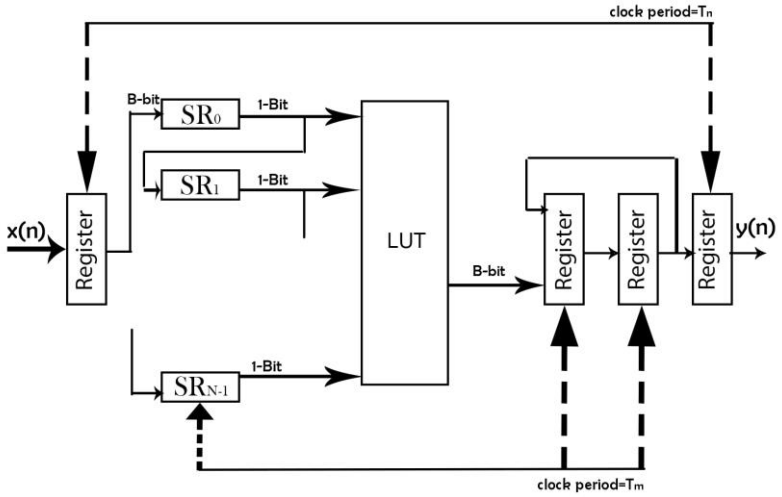

Figure 1.5 Structure basées sur l'arithmétique distribuée d'un filtre FIR.

1.8.1 Principe de fonctionnement

L'arithmétique distribuée repose sur la décomposition des opérations arithmétiques en opérations sur des bits individuels. Par exemple, une multiplication sur des mots entiers de 32 bits peut être décomposée en 32 multiplications de bits individuels. Ces opérations sur des bits peuvent ensuite être exécutées simultanément sur plusieurs unités de traitement spécialisées, telles que des additionneurs-porteurs ou des multiplicateurs bit à bit.

1.8.2 Avantages des structures basées sur l'arithmétique distribuée

Les structures basées sur l'arithmétique distribuée présentent des avantages considérables en termes de performance :

a) **Gains de performance significatifs :** Permettent d'atteindre des vitesses d'exécution bien supérieures aux structures directes et transposées.
b) **Réduction de la latence :** Le traitement des données est plus rapide en raison de l'exécution simultanée des opérations sur des bits.
c) **Consommation d'énergie optimisée :** L'exploitation du parallélisme permet d'améliorer l'efficacité énergétique du filtrage.
d) **Facilité de mise en œuvre** : Les algorithmes de traitement numérique du signal basés sur l'arithmétique distribuée sont plus faciles à mettre en œuvre que les filtres analogiques . Ils offrent une meilleure précision et une meilleure reproductibilité des résultats.

1.8.3 Inconvénients des structures basées sur l'arithmétique distribuée

Les structures basées sur l'arithmétique distribuée présentent des inconvénients majeurs :

a) **Complexité accrue de la conception et de l'implantation :** Nécessitent des unités de traitement spécialisées et une conception plus sophistiquée.
b) **Augmentation du coût matériel :** Les unités de traitement spécialisées peuvent être plus coûteuses que les composants standard.
c) **Moins adaptables à divers types de filtres :** Leur efficacité est optimale pour les filtres FIR avec des coefficients symétriques ou répétitifs.

Sensibilité aux erreurs de quantification : La répartition des calculs sur plusieurs éléments peut amplifier les effets des erreurs de quantification, pouvant dégrader la précision du filtre .Une attention particulière doit être portée à la gestion des erreurs de calcul dans ces structures.

1.8.4 Implémentation FPGA des structures basées sur l'arithmétique distribuée

La technique DA est une structure de calcul efficace dans laquelle les multiplicateurs sont remplacés par une table de recherche. La structure du filtre FIR basé sur DA est composée d'une unité de registre à décalage avec registres à décalage, une unité LUT de bits, une unité de décalage-additionneur et un registre. Comme illustré dans la Figure 1.5, la configuration FIR basée sur DA avec une longueur d'entrée de B bits et une période d'horloge d'échantillonnage , la période d'horloge de décalage serait inférieure ou égale à , ce qui est actuellement facilement réalisable. Cependant, la taille de l'unité LUT DA augmente de manière exponentielle avec l'ordre du filtre.[11,12]

- **Réduction de la taille de l'unité LUT**

Pour contourner cet inconvénient et réduire la taille de l'unité LUT de la structure basée sur DA, plusieurs approches modifiées sont suggérées pour l'implémentation du filtrage FIR et du calcul de produit interne. De plus, la structure basée sur DA reste l'option inévitable pour l'implémentation de filtres adaptatifs FIR d'ordre élevé dans les FPGA à faible coût.

- **Implémentation FPGA de l'unité de registre à décalage DA**

Notre investigation approfondie de l'implémentation FPGA de l'unité de registre à décalage DA révèle que chaque partie de B bits est routée en utilisant une tranche LUT de SR bits. SR représente la taille maximale des bits de la LUT. Ainsi, l'approche DA entraîne le chômage de (SR - B) bits du LUT-SR. Par exemple, l'implémentation d'une profondeur d'échantillonnage de 16 bits sur un FPGA Spartan-6, qui est un FPGA LUT-SR de 32 bits, entraîne un taux de chômage allant jusqu'à 50%. Notre structure proposée est conçue pour atteindre une implémentation efficace et équilibrée en évitant cet inconvénient mentionné.[13,14]

1.9 Conclusion

Le choix de l'architecture pour implémenter un filtre FIR dépend de plusieurs facteurs, tels que les performances requises, les ressources disponibles et les contraintes de conception. Les structures directes sont simples à implémenter, mais elles peuvent être inefficaces pour les filtres de grand ordre. Les structures transposées et symétriques peuvent offrir de meilleures performances, mais elles nécessitent un

prétraitement et un post-traitement des données. Les structures basées sur l'arithmétique distribuée sont les plus performantes, mais elles sont également les plus complexes à implémenter.

Chapitre 2 : Etude de la méthode pour l'implémentation du filtrage FIR sur FPGA

2.1 Introduction

Dans ce chapitre , nous présentons une nouvelle architecture matérielle de filtre FIR d'ordre élevé, pour les applications en temps réel basées sur FPGA, afin d'atteindre une réduction équilibrée de l'utilisation de la logique FPGA. Notre architecture proposée est configurée pour offrir la capacité de mettre à jour les coefficients du filtre en temps réel.

2.2 Structure principale de l'architecture

2.2.1 Système de traitement de filtre FIR proposé

Le système de traitement proposé est illustré à la Figure 2.1. Il comprend un convertisseur série-parallèle synchrone, un convertisseur parallèle-série asynchrone, un compteur binaire contrôlé, une mémoire et un accumulateur multiplicateur. Le convertisseur série-parallèle synchrone est implémenté à l'aide d'un registre à décalage série-parallèle. Le convertisseur parallèle-série asynchrone est implémenté à l'aide d'un multiplexeur.

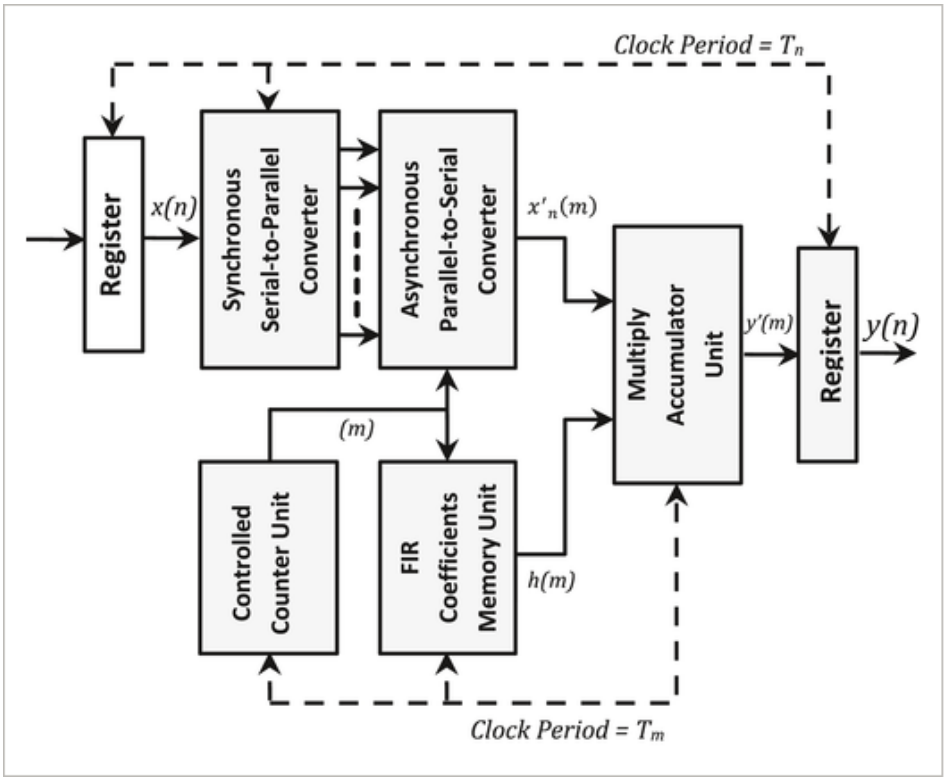

Figure 2.1 Système de traitement de filtre FIR proposé.[10]

$x(n)$: le signal d'entrée du système.

$x'n(m)$: le signal intermédiaire au sein du système.

$y'(m)$: Un autre signal intermédiaire dérivé de $x'n(m)$.

$y(n)$: le signal de sortie du système.

$h(m)$: la réponse impulsionnelle du filtre.

La sortie m du compteur contrôlé gère à la fois l'adressage de la mémoire et le contrôle du multiplexeur. Par conséquent, les échantillons $x'_n(m)$ de sortie doivent être égaux aux échantillons d'entrée $x(n)$, en prenant en compte le temps de basculement.

$$x'_n(m) = x(n-m) \qquad (2.1)$$

a) **Convertisseur série-parallèle synchrone** : C'est un composant qui convertit des données d'un format série à un format parallèle de manière synchrone, c'est-à-

dire que la conversion est synchronisée avec une horloge. Il est implémenté à l'aide d'un registre à décalage série-parallèle, ce qui suggère que les données sont d'abord reçues en série puis décalées dans le registre pour être converties en format parallèle.

b) **Convertisseur parallèle-série asynchrone :** C'est un composant qui convertit des données d'un format parallèle à un format série de manière asynchrone, ce qui signifie que la conversion n'est pas synchronisée par une horloge centrale. Il est implémenté à l'aide d'un multiplexeur, qui est un dispositif permettant de sélectionner l'une des entrées et de la transmettre en sortie.

c) **Compteur binaire contrôlé** : C'est un composant qui compte les impulsions binaires et qui peut être contrôlé ou configuré selon les besoins du système.

d) **Mémoire** : C'est un composant qui stocke des données et des instructions pour le traitement ultérieur. Il peut s'agir de mémoire volatile ou non volatile, selon que les données sont conservées lorsque l'alimentation est coupée ou non.

e) **Accumulateur multiplicateur :** C'est probablement un composant qui accumule les résultats de multiples opérations de multiplication. Cela peut être utilisé dans des calculs répétitifs ou itératifs où le résultat de chaque multiplication est ajouté à un total cumulatif.

2.2.1.1 Avantages du système de traitement proposé

• Réduction de l'utilisation simultanée des ressources DSP et LUT grâce à une répartition efficace des tâches de traitement.

• Structure flexible et évolutive pour s'adapter à une large gamme de longueurs et de complexités de filtre FIR.

• Prise en charge efficace des coefficients de filtre reconfigurables pour une adaptabilité accrue.

• Conception simple et modulaire pour une implémentation et une maintenance faciles.

2.2.2 Transformation des échantillons pour un filtre FIR

La transformation des échantillons pour un filtre FIR (Finite Impulse Response) est une étape cruciale dans le processus de filtrage numérique. Elle consiste à convertir un signal temporel discret en une représentation fréquentielle, puis à appliquer la réponse fréquentielle du filtre pour modifier le spectre du signal. Le signal filtré est ensuite obtenu en transformant la sortie fréquentielle du filtre dans le domaine temporel.[21]

La Figure 2.2 montre un exemple simplifié de notre transformation d'échantillon d'entrée x en échantillon de sortie x' avec $(N = 4)$. Pour chaque nouvelle valeur de n , m augmente de 0 à $(N - 1)$ (=3 dans la Figure 3) avec une période d'horlogeT_m.

À titre d'illustration, les valeurs de$x'_n(m)$ et $x'_{n+1}(m)$peuvent être obtenues comme suit :

$$\begin{cases} x'_n(0) = x(n) \\ x'_n(1) = x(n-1) \\ x'_n(2) = x(n-2) \\ x'_n(3) = x(n-3) \end{cases} \qquad \begin{cases} x'_{n+1}(0) = x(n+1) \\ x'_{n+1}(1) = x(n) \\ x'_{n+1}(2) = x(n-1) \\ x'_{n+1}(3) = x(n-2) \end{cases} \qquad (2.2)$$

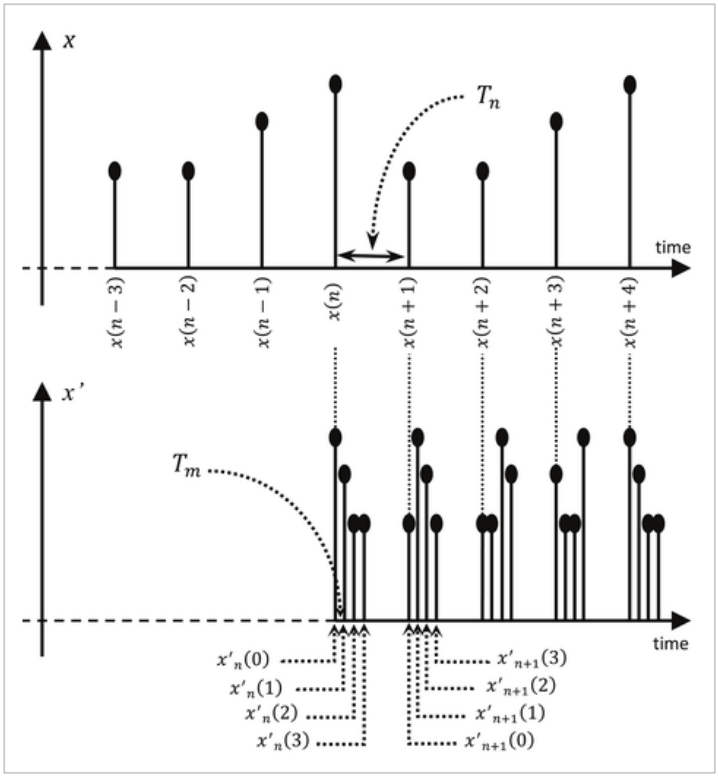

Figure 2.2 Transformation des échantillonspour un filtre FIR à 4 coefficients $(N = 4)$**.**

La sortie du convertisseur asynchrone parallèle-série produit, après chaque nouvel échantillon d'entrée, les $(N-1)$ échantillons d'entrée précédents. Le temps correspondant de cette séquence doit être inférieur à la période des échantillons In. Pour réaliser une conversion complète asynchrone parallèle-série entre deux échantillons d'entrée, la période Im doit être inférieure ou égale à (Tn/N) pour les filtres non symétriques. Ainsi, le calcul de multiplication-accumulation peut être effectué plusieurs fois entre deux échantillons d'entrée.[10]

Similaire aux structures basées sur DA, la période de calcul de multiplication-accumulation de notre nouveau système de traitement proposé sera égale à la période Im. Par conséquent, une tranche DSP FPGA peut être réutilisée plusieurs fois.[10]

Les coefficients FIR$h$sont stockés et générés à l'aide d'une mémoire, dont la taille doit au moins correspondre à la taille N × B-bit. Les coefficients et les échantillons x'n (m) doivent être sélectionnés en parallèle. Pour une valeur m, $x'n\ (m)$ et $h\ (m)$ sont générés simultanément. Les valeurs y'n (m) sont les résultats du calcul de multiplication-accumulation de $h\ (m)$ et $x'n\ (m)$. En conséquence, $Y\ n\ (m)$ peut être exprimé par :

$$y'_n(m) = y'_n(m-1) + h(m)\,x'_n(m) \qquad (2.3)$$

où les valeurs de y'_n pour $\{m = 0,1,\dots,N-1\}$ sont données comme :

$$\begin{cases} y'_n(0) = y'_n(-1) + h(0)\,x'_n(0) \\ y'_n(1) = y'_n(-1) + h(0)\,x'_n(0) + h(1)\,x'_n(1) \\ \vdots\vdots \\ \vdots\vdots \\ y'_n(N-1) = y'_n(-1) + \sum_{m=0}^{N-1} h(m)\,x'_n(m) \end{cases} \qquad (2.4)$$

Pour clarifier, supposons que les données stockées par le convertisseur synchrone série-parallèle sont $\{x\,(n)$, a $(n-1),\dots,I\,(n-(N-1))\}$. Ensuite, $x'_n(m)$ peut être remplacé en utilisant (4) et le dernier terme de (3) peut être réécrit comme suit :

$$y'_n(N-1) = y'_n(-1) + \sum_{m=0}^{N-1} h(m)\,x(n-m) \qquad (2.5)$$

Nous pouvons affirmer que $y'_n(N-1)$ fournit la somme de l'expression du filtre FIR et de la valeur initiale de l'accumulateur de multiplication. Par conséquent, le système de traitement de sortie y (n) sera équivalent à la sortie du filtre FIR, si seuls $y'n\,(N-1)$ sont stockés dans le registre de sortie et : $y'_n(-1) = 0$. Cela signifie que le registre de sortie doit être alimenté par le relai d'échantillonnage In et l'unité d'accumulateur de multiplication doit être effacée à chaque période d'In. Ensuite, l'expression de $y'n\,(N-1)$ devient :

$$y'_n(N-1) = \sum_{m=0}^{N-1} h(m)\,x(n-m) \qquad (2.6)$$

Par conséquent, notre sortie y(n) sera exprimée par :

$$y(n) = \sum_{m=0}^{N-1} h(m)\,x(n-m) \qquad (2.7)$$

Ce qui est un équivalent parfait de :

$$y(n) = \sum_{i=0}^{N-1} h(i)\,x(n-i) \qquad (2.8)$$

2.2.3 DSP (Digital Signal Processor)

Le DSP est un microprocesseur spécialisé généralement programmé en C, peut-être avec un code d'assemblage pour les performances. Il est bien adapté aux tâches mathématiques extrêmement complexes, avec traitement conditionnel. Ses performances sont limitées par la fréquence d'horloge et le nombre d'opérations utiles qu'il peut effectuer par horloge. Pour une conception de calcul élevée et des performances améliorées, les FPGA modernes disposent de ressources dédiées telles que des multiplicateurs et des blocs DSP. Les principales applications DSP sont le filtrage, la compression, la FFT, la DFT, l'encodage et le décodage des flux d'entrée. Ces opérations nécessitent des ressources dédiées qui peuvent prendre en charge le pipelining et l'exécution dans un délai plus court. Pour cela, les blocs DSP sont utilisés comme ressource dédiée dans les FPGA modernes. La différence fondamentale entre un processeur DSP et un processeur générique est le bloc matériel de multiplication-accumulation (MAC) du processeur DSP et les structures de mémoire et de bus spécialisées pour faciliter l'accès fréquent aux données que l'on trouve couramment dans les applications DSP.[15]

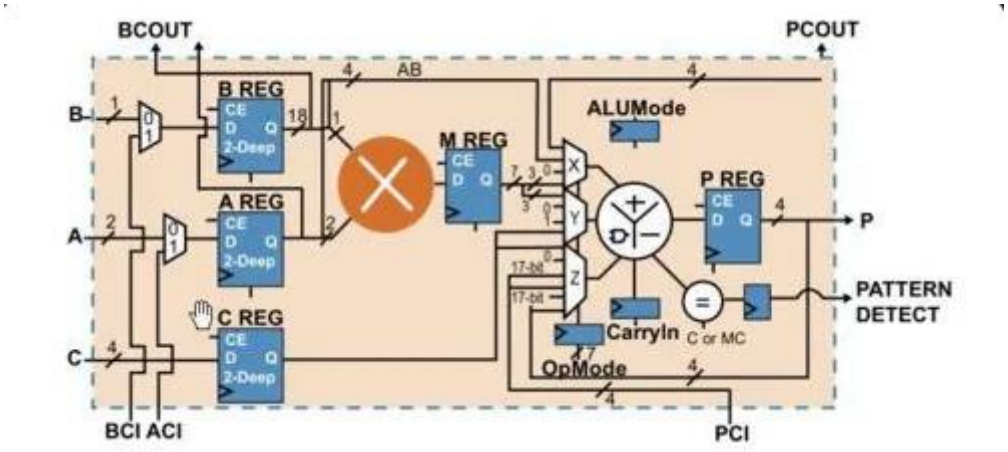

Figure 2.3 Structure DSP[15]

2.2.4 Block RAM (BRAM)

BRAM est une mémoire intégrée et les FPGA sont constitués de BRAM à port unique et à deux ports. Selon l'architecture du dispositif FPGA, chaque BRAM se compose de plusieurs cellules de RAM statiques. Parmi eux, quelques cellules sont

utilisées pour la configuration de la mémoire, et le reste est utilisé pour le stockage des données. BRAM est utilisé pour le stockage de données interne, utilisé pour concevoir des FIFO, des tampons, des piles et peut être utilisé pour stocker les données requises par le FSM. Chaque BRAM a une horloge et une activation d'horloge, lecture et écriture, et chaque BRAM est synchronisée. Si nous considérons la BRAM à double port, les deux ports peuvent être utilisés de manière interchangeable et peuvent contrôler les opérations de lecture et d'écriture simultanées.[16]

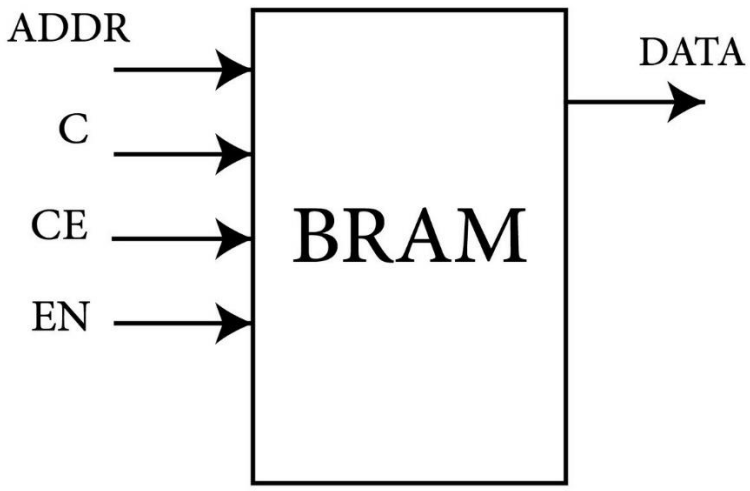

Figure 2.4 Structure de BRAM

2.2.5 LUT (Look-Up Table)

Le concept LUT peut être facilement compris en utilisant les conceptions basées sur MUX. En effet, la fonction logique est réalisée à l'aide de la LUT. LUT fournit le retard uniforme, quel que soit le nombre d'entrées pour la même conception.[20]

Les FPGAs modernes permettent de configurer les LUTs de différentes manières, telles que des ROMs de base, des RAMs et des registres de décalage. Configurer les LUTs en tant que registres de décalage offre un moyen attractif d'ajouter plus de bits de stockage à un générateur linéaire binaire : par exemple, ajouter un registre de décalage SRL32 Xilinx à un générateur r-bit optimisé pour les LUT permet d'augmenter la taille de l'état à n = r + 32.[17]

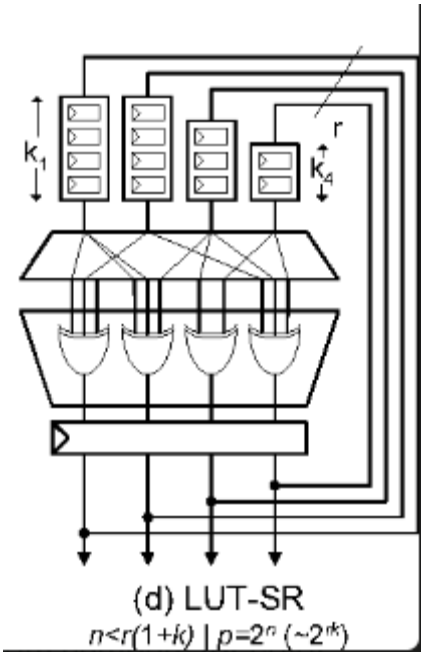

Figure 2.5 Structure de LUT. [17]

2.2.6 structure principale du filtre FIR d'ordre élevé reconfigurable pour une implémentation sur FPGA.

Les filtres à réponse impulsionnelle finie (FIR) sont essentiels dans le traitement du signal numérique (DSP), utilisés pour le filtrage, le lissage et le débruitage des signaux. Les filtres FIR d'ordre élevé offrent une grande flexibilité et précision dans la conception de la réponse fréquentielle.

La figure 2.6 illustre la structure principale proposée du filtre FIR reconfigurable basé sur FPGA. Cette structure se compose d'une unité de registre-multiplexeur, d'une unité DSP, d'une unité de contrôle et d'une unité de mémoire RAM des coefficients. L'unité de mémoire RAM des coefficients utilise la BRAM FPGA, afin de gérer les coefficients de filtre reconfigurables. L'adoption de la BRAM FPGA et de la tranche DSP est motivée par la structure logique FPGA. La hauteur d'une tranche DSP FPGA est la même que cinq tranches de blocs logiques configurables et correspond également à la hauteur d'une BRAM FPGA de 18.[19,20]

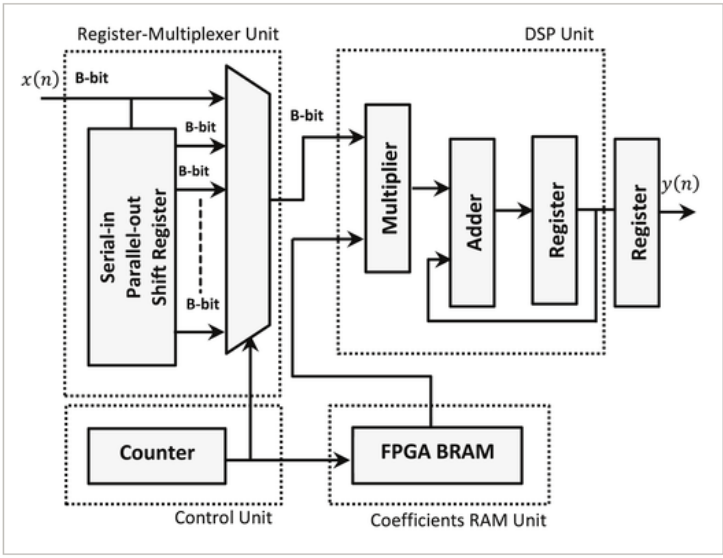

Figure 2.6 Structure principale du filtre FIR d'ordre élevé reconfigurable pour une implémentation sur FPGA.[10]

- **Les avantage de structure principale du filtre FIR d'ordre élevé reconfigurable pour une implémentation sur FPGA.**

-L'avantage supplémentaire de la structure proposée est l'association, en tant qu'unité unique, du registre à décalage série-parallèle avec le multiplexeur asynchrone, exploitant les connexions internes de la tranche et du LUT. Cette configuration permet un registre à décalage de 128 bits et un multiplexage n'utilisant qu'une seule tranche FPGA.

2.2.7 La structure de l'unité registre-multiplexeur de B bits.

Le maintien des échantillons d'entrée de B bits de notre filtrage FIR à 128 taps nécessite B tranches FPGA et seulement B LUTs pour le 32 taps. Les tranches B sont connectées en parallèle pour accomplir le calcul basé sur B bits. Les connexions détaillées sont présentées dans la figure 2.7.

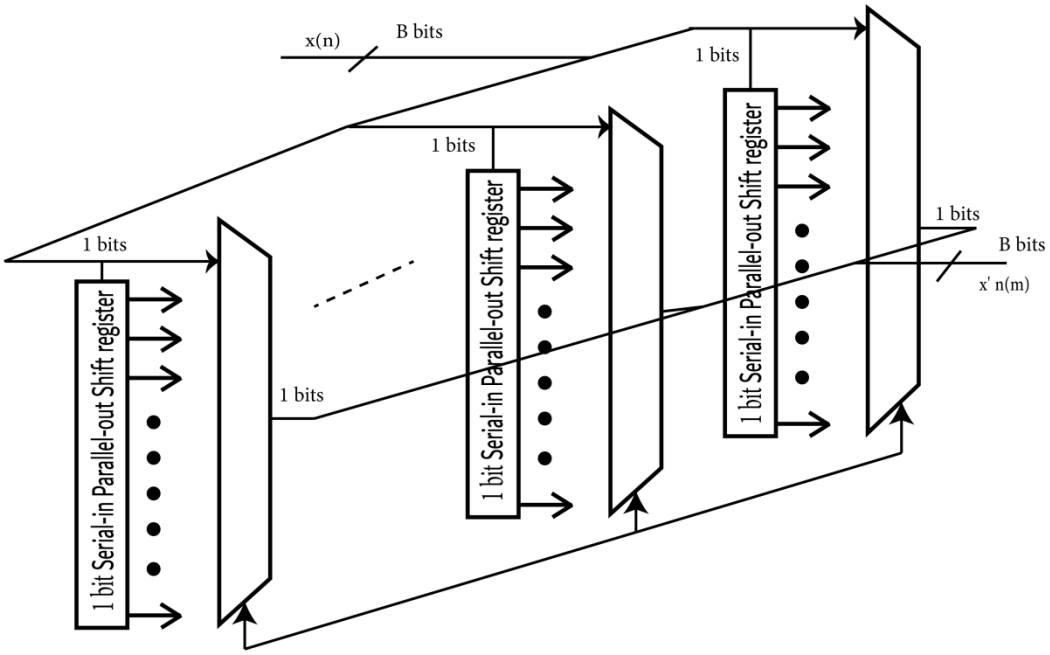

Figure 2.7 Structure de l'unité registre-multiplexeur de B bits.

a) les composant La structure de l'unité registre-multiplexeur de B bits :

- **Registre:** Le registre est une mémoire qui peut stocker B bits de données. Les données sont écrites dans le registre via une entrée de données et lues via une sortie de données.
- **Multiplexeur:** Le multiplexeur sélectionne une source de données parmi plusieurs entrées. Il dispose d'une entrée de sélection qui détermine quelle entrée est connectée à la sortie du multiplexeur.
- **Commutateur:** Le commutateur contrôle la direction du flux de données. Il peut connecter la sortie du registre à l'entrée du multiplexeur ou à la sortie de l'unité.

b) Avantages de la structure de l'unité registre-multiplexeur de B bits :

La structure de l'unité registre-multiplexeur de B bits présente plusieurs avantages :

- **Parallélisme:** Les tranches FPGA sont connectées en parallèle, ce qui permet d'effectuer le calcul du produit scalaire en parallèle. Cela permet d'améliorer les performances du filtre FIR.

- **Flexibilité:** La structure de l'unité registre-multiplexeur de B bits est flexible et peut être facilement adaptée à des filtres FIR de différentes longueurs de taps.
- **Efficacité:** La structure de l'unité registre-multiplexeur de B bits est efficace en termes de ressources matérielles. Le nombre de tranches FPGA et de LUTs nécessaires est proportionnel à la longueur du filtre FIR.

2.2.8 Unitè DSP (Digital Signal Processor)

L'unité DSP effectue les multiplications et les additions nécessaires au calcul des produits intermédiaires et de la sortie finale du filtre. Elle peut être implémentée à l'aide de blocs DSP dédiés ou de blocs logiques programmables (LUTs) du FPGA.[15]

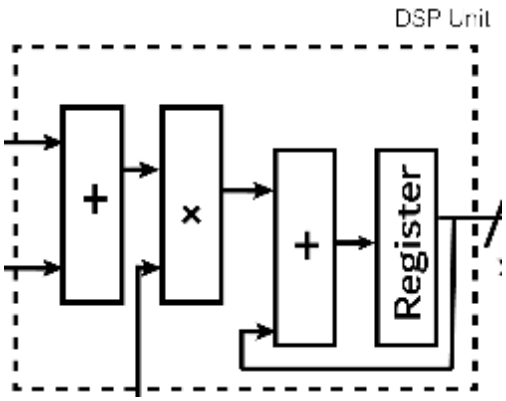

Figure 2.8 Structure DSP unit

2.2.9 unitè de controle

L'unité de contrôle séquence les opérations de la RMU et de l'unité DSP. Elle gère également la configuration du filtre, telle que le choix des coefficients de filtre et le mode de fonctionnement (filtrage direct, filtrage par transposition, etc.).

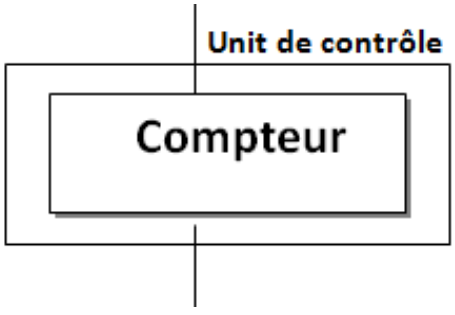

Figure 2.9 Structure compteur.

2.2.10 Unitè de coefficient RAM

Les BRAMs jouent un rôle essentiel pour stocker efficacement les coefficients reconfigurables des filtres FIR sur FPGA, avec une utilisation qui dépend de la taille du filtre et de la fréquence d'horloge. L'utilisation des DSPs est plus liée à la complexité des coefficients.

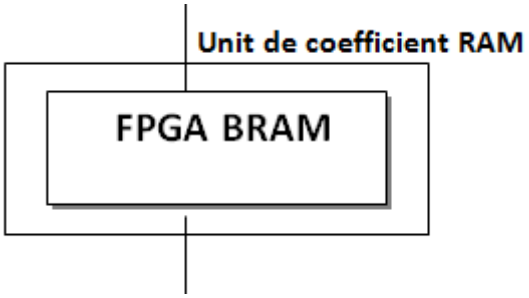

Figure 2.10 Structure Block RAM

2.2.11 Interfaces d'entrée/sortie

Les interfaces d'entrée/sortie permettent au filtre FIR de recevoir des échantillons d'entrée et de produire des échantillons de sortie. Les interfaces peuvent être implémentées à l'aide de broches d'E/S du FPGA ou de blocs IP spécifiques.

2.3 Architecture de filtre FIR symétrique d'ordre élevé

La structure principale du filtre FIR reconfigurable proposée peut être appliquée au cas symétrique.

Une réduction supplémentaire de l'utilisation de la logique peut être obtenue. En exploitant la nouvelle structure proposée, l'ordre du filtre peut passer de simple à double, sans aucune utilisation de logique FPGA supplémentaire significative pour les trois unités : l'unité de contrôle, l'unité de mémoire RAM des coefficients et l'unité DSP. La figure 2.11 montre la structure proposée du filtre FIR symétrique reconfigurable de haut ordre pour la mise en œuvre FPGA. L'unité de registre-multiplexeur est divisée en deux parties. Seule la taille du registre à décalage doit être agrandie pour le filtre d'ordre faible. Les filtres FIR dont l'ordre est un multiple de s R bits ne nécessitent pas d'utilisation de logique supplémentaire. De plus, l'unité de contrôle nécessite moins d'utilisation de logique. Ces avantages sont principalement obtenus grâce à la

configuration structurelle des LUT de registre à décalage FPGA et à la composition récente des tranches DSP FPGA, qui comportent un pré-additionneur avant le multiplicateur en plus de l'additionneur principal.

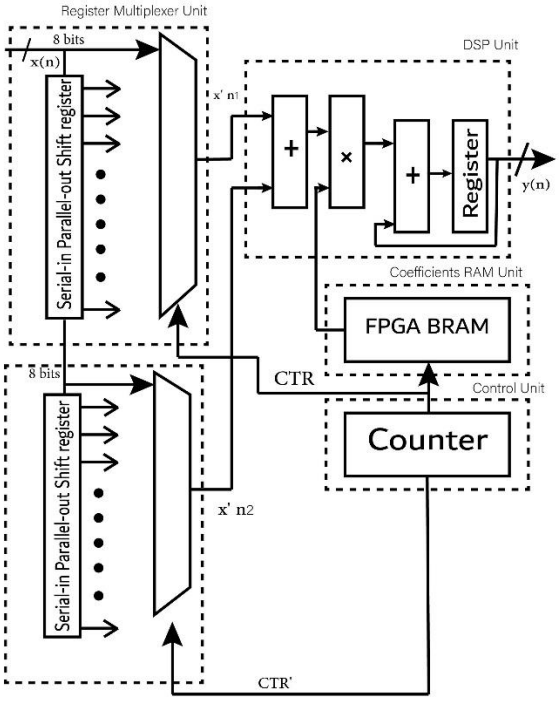

Figure 2.11 Structure proposée du filtre FIR symétrique d'ordre élevé reconfigurable pour une implémentation sur FPGA

$X(n)$: le signal entrer de regitre 8 bits

$X'(n)$: le signal sortie de regitre 256 bits

$Y(n)$: le signal de sortie

En général, la fréquence de fonctionnement maximale des tranches DSP FPGA fosp est légèrement supérieure à la fréquence de fonctionnement maximale des BRAM FPGA Bram. En conséquence, la fréquence d'échantillonnage d'entrée maximale fis de le filtre FIR symétrique proposé sera déterminée par Bram. Elle peut être doublée par rapport à la structure de base et est estimée à l'aide de l'expression suivante :

$$f_{IS} = \frac{f_{BRAM} \times 2}{N} \qquad (2.9)$$

2.4 Architecture symétrique modifiée pour implémentation FPGA

pour les filtres symétriques, Im doit être inférieur à (2Tn/N) et la valeur maximale de la sortie du compteur contrôlé doit être équivalente à (N/2 - 1), où N est l'ordre du filtre FIR symétrique. Les deux multiplexeurs doivent être contrôlés de manière inverse à l'aide des signaux CTR et CTR' (figure 2.12). Si la première entrée du premier multiplexeur est sélectionnée, la dernière entrée du deuxième multiplexeur doit être sélectionnée. Ainsi, le CTR fournit un incrément de 0 à (N/2 - 1) et le CTR' fournit un décrément de (N/2-1) à 0**.**

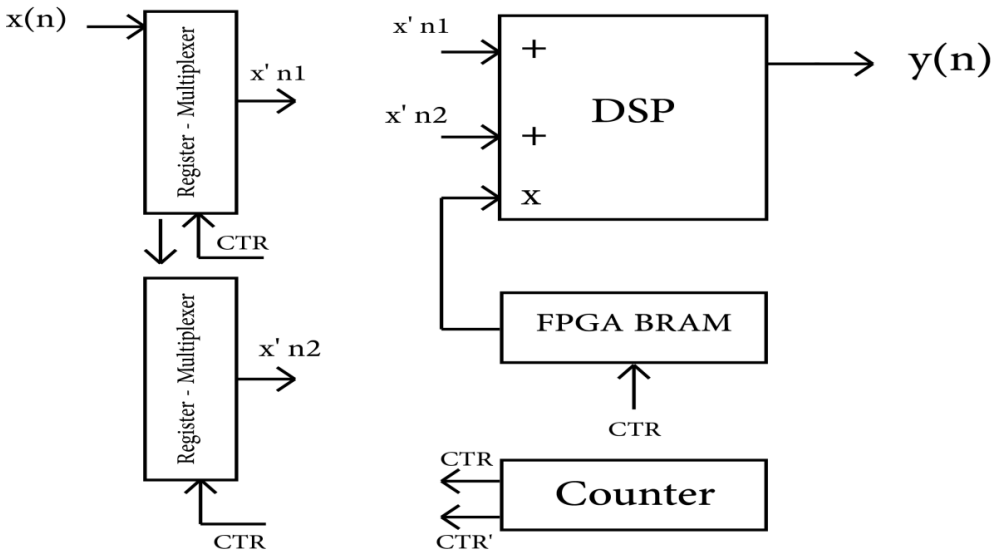

Figure 2.12 Structure l'architecture symétrique modifiée pour implémentation FPGA

2.5 Conclusion

La méthode proposée d'utilisation simultanée réduite des blocs DSP et des LUT pour la mise en œuvre du filtrage FIR sur les FPGA offre une approche prometteuse pour équilibrer l'utilisation des ressources et les performances. La méthode utilise efficacement les points forts des blocs DSP et des LUT pour réaliser un filtrage FIR efficace sur les FPGA.

Chapitre 3 : Implémentation et résultats

3.1 **Introduction**

Dans ce chapitre, nous présentons les résultat de implémentation effectuées a l'aide des logiciel Matlab et xilinx ISE pour implémentation FPGA et comparant sur l'architecture de résultat de l'implémentation.

3.2 **Matlab**

MATLAB est un langage de programmation et une plateforme unique pour le calcul numérique, la simulation et la programmation. Il permet d'accéder à un environnement interactif tout-en-un pour traiter des matrices, tracer des données et concevoir diverses interfaces utilisateur.

MATLAB offre plusieurs options pour travailler avec les FPGA :

HDL Coder génère du code Verilog et VHDL synthétisable et portable à partir de fonctions MATLAB, de modèles Simulink et de diagrammes Stateflow2. Le code HDL généré peut être utilisé pour la programmation FPGA ou la conception et le prototypage ASIC.MATLAB offre des solutions intégrées pour concevoir des systèmes sur FPGA en partant de modèles Simulink et de code MATLAB.

Bien que payantes, ces solutions permettent une conception productive et une exploration rapide des architectures[21].

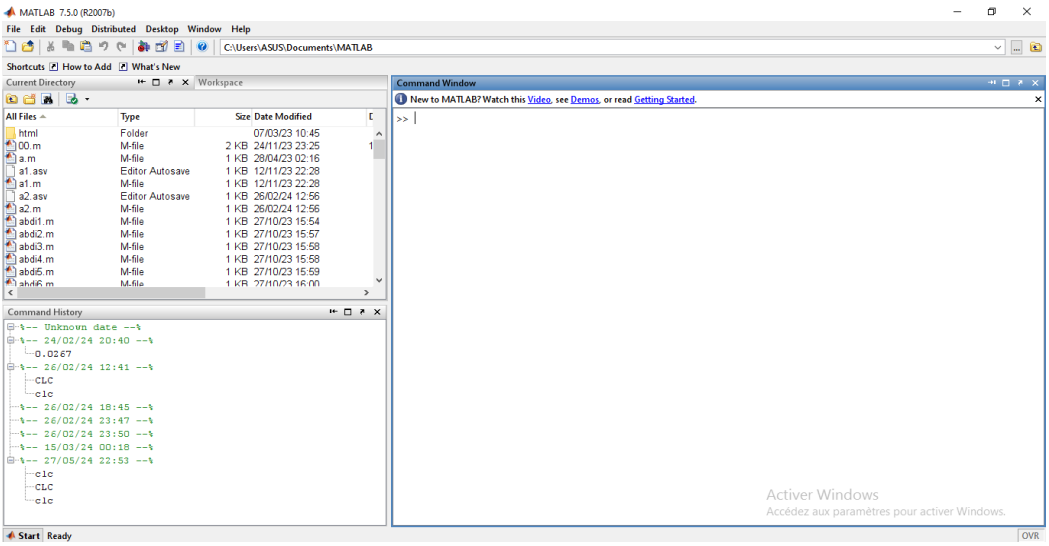

Figure 3.1 Interface graphique de logiciel Matlab

3.3 **Xilinx ISE**

Xilinx ISE, abrégé de Integrated Synthesis Environment, est un outil logiciel abandonné de Xilinx pour la synthèse et l'analyse des conceptions HDL, qui cible principalement le développement de micrologiciels embarqués pour les familles de produits de circuits intégrés Xilinx FPGA et CPLD

. Il a été remplacé par Xilinx Vivado, offrant des fonctionnalités améliorées pour le développement de systèmes sur puce. La dernière version, 14.7, a été publiée en octobre 2013, marquant la fin des versions planifiées d'ISE

. La suite de conception ISE comprend différentes éditions comme Suscription et Web Edition, répondant à différents besoins des utilisateurs et niveaux de support de périphériques[18].

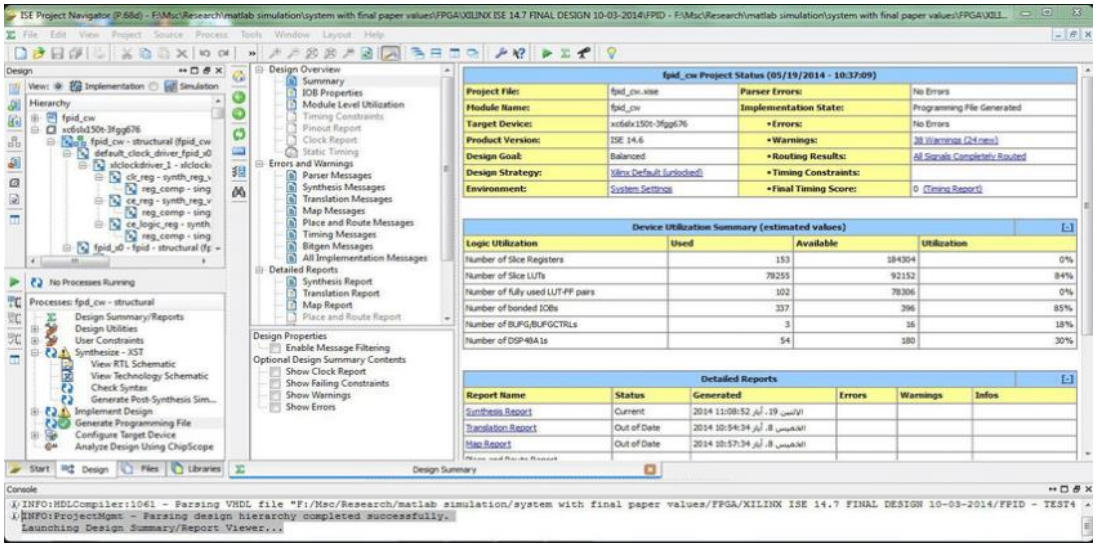

Figure 3.2 Interface graphique de logiciel Xilinx ISE

3.4 XIlinx vivado

Xilinx Vivado est une suite de développement logiciel de conception matérielle développée par Xilinx. Il remplace l'ancien environnement de développement ISE (Integrated Synthesis Environment) et offre des outils avancés pour la conception, la simulation et la vérification des circuits logiques programmables, tels que les FPGA (Field-Programmable Gate Arrays) et les SoC (System on Chips). Vivado inclut des fonctionnalités pour la synthèse, l'implémentation, l'analyse temporelle, la vérification de conception, ainsi que pour la programmation et la configuration des dispositifs Xilinx.

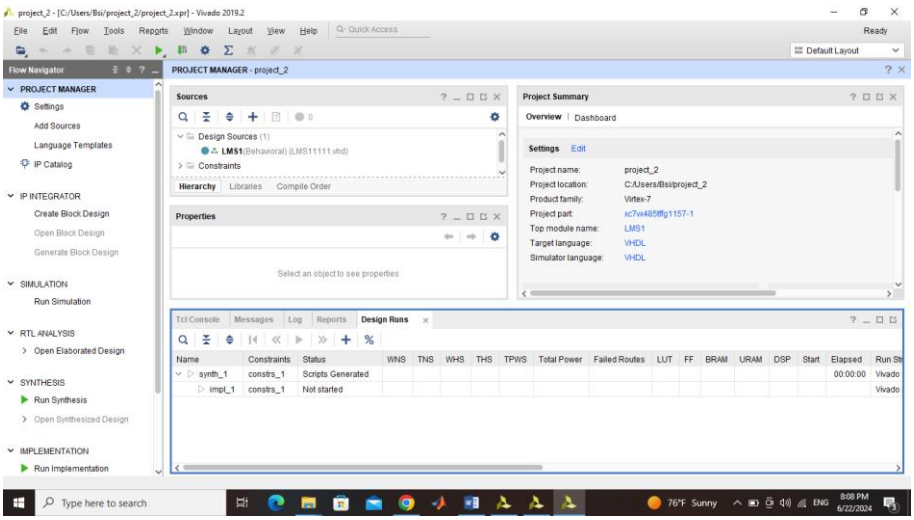

Figure 3.3 Interface de xilinx vivado

3.5 Implémentation sur FPGA

3.5.1 Implémentation DSP

Cette figure 3.4 représente un schéma d'un DSP (Digital Signal Processing) , souvent utilisé dans du filtre FIR symétrique d'ordre élevé reconfigurable pour une implémentation sur FPGA, ce diagramme illustre Les entrées passent par les registres et sont ensuite envoyées à l'additionneur et au multiplicateur. Les sorties de l'additionneur et du multiplicateur sont ensuite envoyées vers un autre registre avant de produire la sortie finale. Il y a un signal de réinitialisation ("reset_n") connecté via une porte inverseuse , qui semble être utilisé pour réinitialiser les registres. Les lignes rouges indiquent le chemin des données à travers les registres, l'additionneur, et le multiplicateur.

Le module DSP qui prend des entrées, effectue des opérations d'addition et de multiplication à l'aide d'un additionneur et d'un multiplicateur combinatoire, et produit une sortie, avec des registres utilisés pour synchroniser et stocker les données.

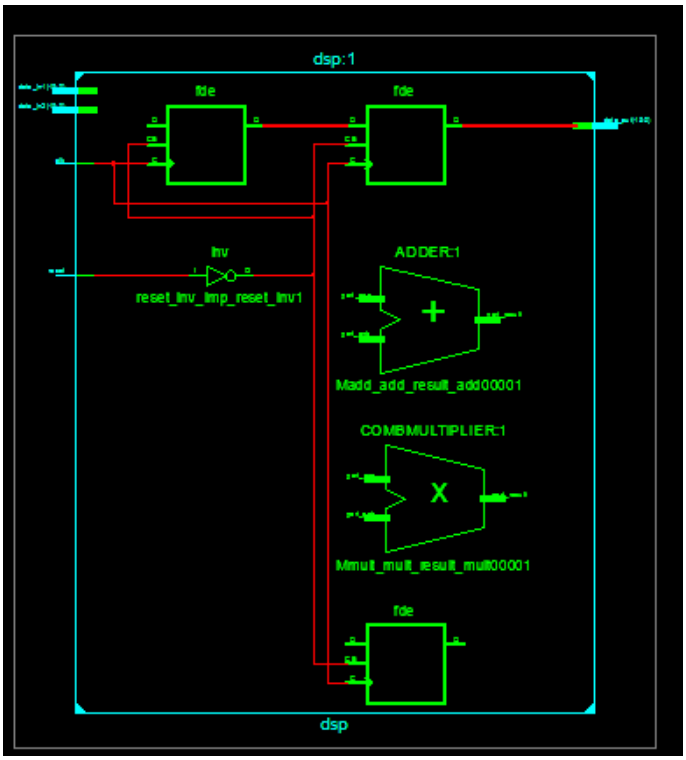

Figure 3.4 Résultat de synthèse DSP.

3.5.2 **Component DSP :**

Cette figure 3.5 représente un schéma d'un component DSP (Digital Signal Processing), ce diagramme représente un component DSP qui reçoit deux entrées de données de 16 bits, un signal d'horloge et un signal de réinitialisation, traite ces entrées à l'intérieur du bloc DSP, et produit une sortie de données de 16 bits

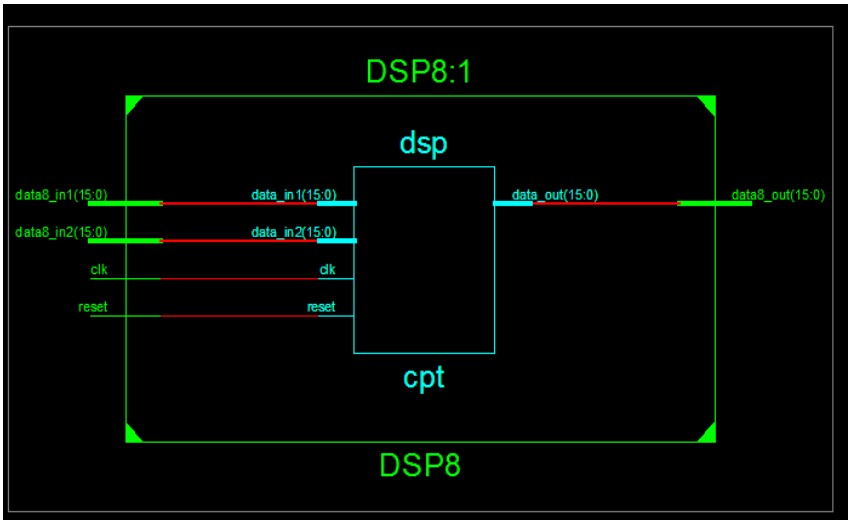

Figure 3.5 Résultat de synthèse component DSP.

3.5.3 **Implémentation BRAM :**

Cette figure 3.6 représente l'implémentation d'une mémoire BRAM (Block RAM), ce diagramme représentePorte OR (or2b2) : Combine les signaux CE et EN pour produire une sortie nommée DATA_or0000_imp_DATA_or00001. Mémoire ROM (Mrom_rdata1) : Utilise l'adresse (ADDR[7:0]) pour lire les données de la mémoire ROM. Inverseur (inv) : Inverse le signal de contrôle pour produire DATA_not0001_imp_DATA_not00011. Registre (fde) : Stocke les données lues de la mémoire ROM et produit la sortie finale DATA(15:0).Les signaux CE et EN sont combinés par la porte OR et le résultat est envoyé à la mémoire ROM. L'adresse ADDR[7:0] est utilisée pour accéder à la mémoire ROM. Les données lues de la mémoire ROM (Mrom_rdata1) sont envoyées au registre fde. L'inverseur (inv) est utilisé pour inverser le signal de contrôle avant de l'envoyer au registre. La sortie du registre est connectée à la sortie finale DATA(15:0).

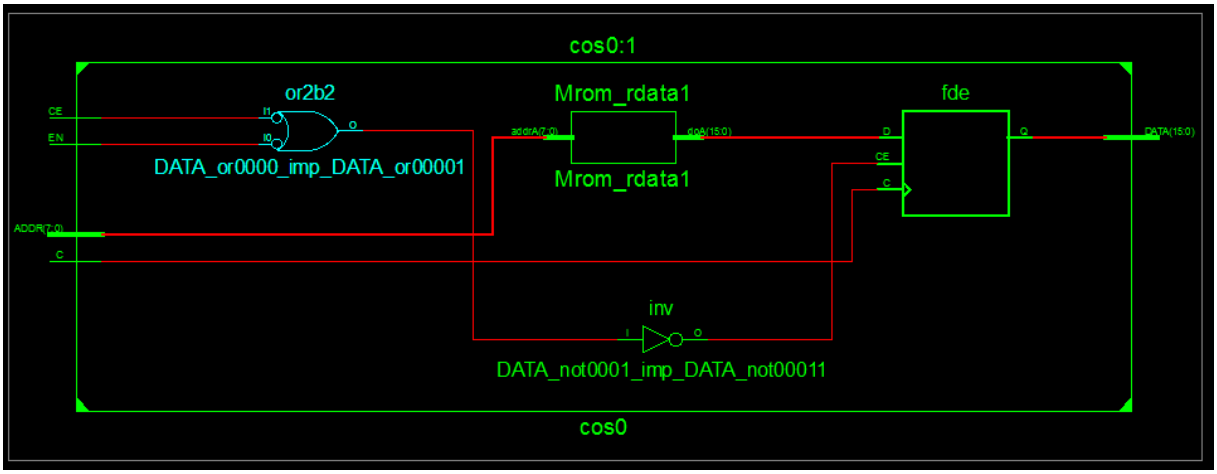

Figure 3.6 Résultat de synthèse BRAM

3.5.4 **Component BRAM :**

Cette figure 3.7 représente un schéma d'un component BRAM , ce schéma montre un bloc de mémoire RAM avec des bus d'adresse et de données, ainsi que des signaux de contrôle typiques (horloge, activation, sélection de puce). Ces composants sont couramment utilisés dans les conceptions FPGA pour stocker des données temporairement pendant le traitement. Le module central, étiqueté "cos0" dans ce cas, semble être connecté à ces bus et signaux pour gérer les opérations de mémoire.

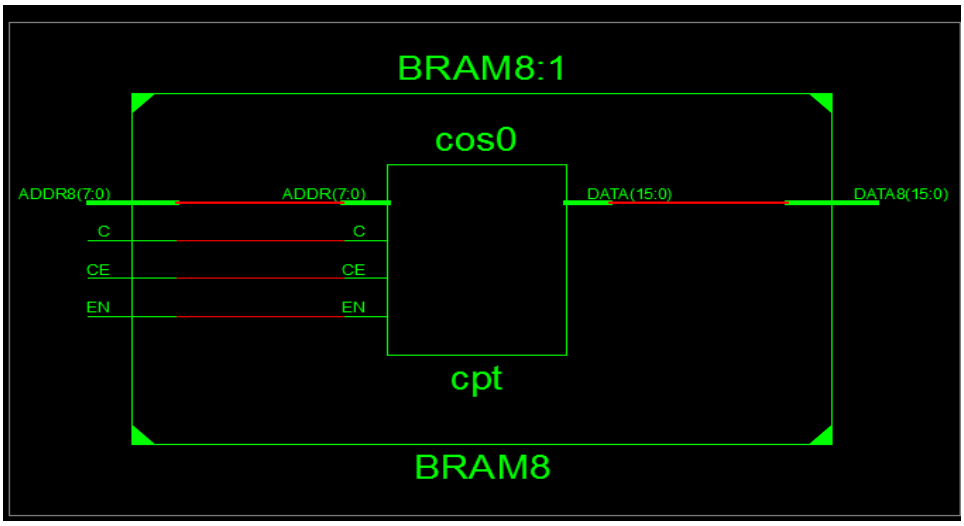

Figure 3.7 Résultat de synthèse component BRAM

3.5.5 **Implémentation compteur**

Ce diagramme montre un compteur numérique implémenté dans un FPGA. Le fonctionnement du compteur repose sur les éléments suivants : les entrées CE (Clock Enable), CLK (Clock), et CLR (Clear). CE permet d'activer ou de désactiver le comptage; lorsque CE est activé, le compteur peut incrémenter sa valeur. CLK est l'horloge du système, et à chaque front montant de l'horloge, si CE est activé, le compteur incrémente sa valeur. CLR réinitialise le compteur; lorsqu'il est activé, le compteur est remis à zéro indépendamment des autres entrées. La sortie principale du compteur, COUNT (7:0), donne la valeur actuelle du compteur sur 8 bits. Le bloc principal du compteur, COUNTER:1, contient la logique de comptage, avec des ports internes comme port_data, port_q, port_clk, et port_sclr connectant les signaux d'entrée et de sortie. En résumé, à chaque front montant du signal CLK, le compteur vérifie l'état de CE pour éventuellement incrémenter la valeur de COUNT, et si CLR est activé, il réinitialise COUNT à zéro, indépendamment des signaux CLK et CE.

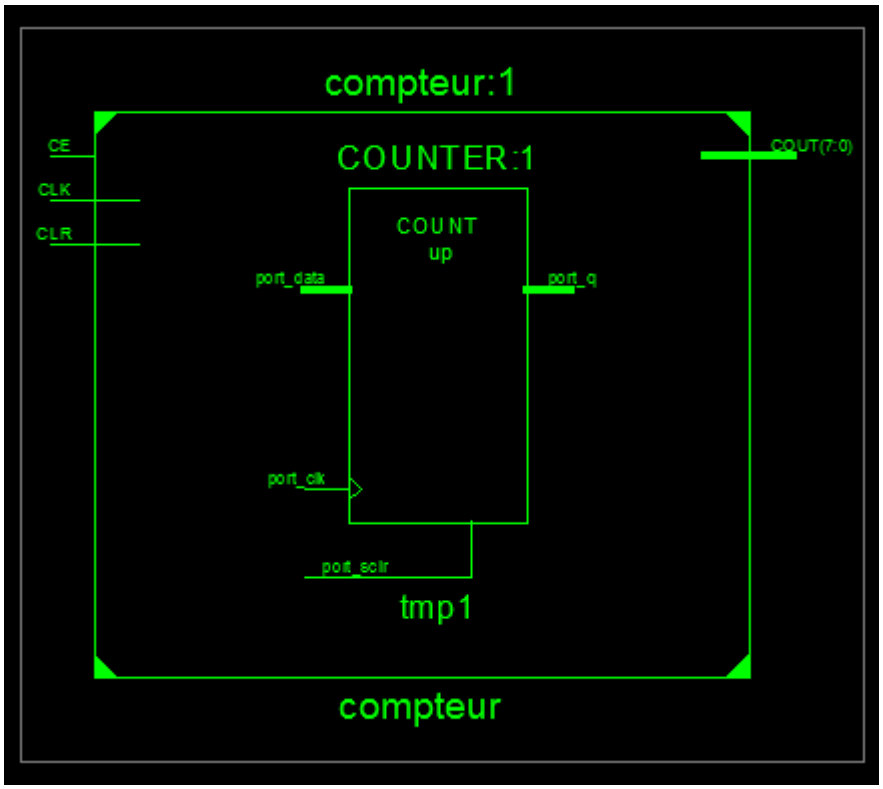

Figure 3.8 Résultat de synthèse compteur

3.5.6 **Component compteur**

Ce schéma représente un compteur modulaire utilisé comme composant pour créer un compteur plus complexe dans un FPGA. Le compteur dispose des entrées CE (Clock Enable), CLK (Clock), et CLR (Clear). CE permet l'activation du comptage; lorsque CE est haut, le compteur peut incrémenter. CLK est le signal d'horloge principal, et à chaque front montant de l'horloge, si CE est activé, le compteur incrémente. CLR réinitialise le compteur à zéro lorsqu'il est activé. La sortie COUNT(7:0) donne la valeur actuelle du compteur sur 8 bits. Dans le contexte du premier compteur, ce compteur modulaire (COMPTEUR8) est encapsulé dans le composant COUNTER:1 pour fournir la fonctionnalité de comptage. COMPTEUR8 agit comme un sous-composant au sein de COUNTER:1, utilisant les mêmes signaux d'entrée (CE, CLK, CLR) et sortant une valeur comptée sur 8 bits (COUNT(7:0)). Les connexions sont directes : les signaux CE, CLK, et CLR sont acheminés vers les entrées correspondantes de COMPTEUR8, et la sortie COUNT(7:0) de COMPTEUR8 est utilisée comme la valeur comptée pour COUNTER:1.

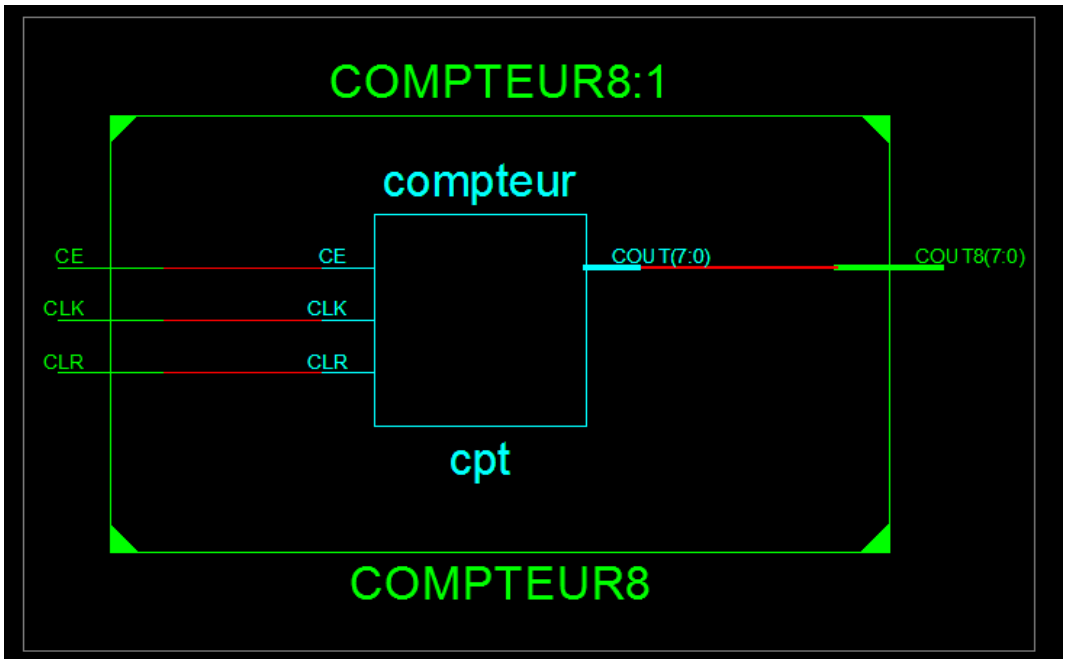

Figure 3.9 Résultat de synthèse component compteur

3.5.7 **Implémentation registre multiplieur**

Le SRM256 se compose d'un registre (fde) et d'un multiplexeur (MUX:1). Le registre possède une entrée de données (D), une sortie de données (Q), une ligne d'activation (CE) et une horloge (C). Le multiplexeur a une entrée de données de 256 bits (Data(255:0)), une sortie (port_result) et une ligne de sélection (port_select). Les entrées et sorties comprennent une ligne de sélection à 8 bits (SL8(7:0)), une entrée de données (Xin), une ligne d'activation (CE), un signal d'horloge (C) et une sortie finale (Yout).

Le fonctionnement du SRM256 est le suivant : les données entrent dans le registre via l'entrée Xin. Le signal CE active le registre, permettant l'écriture des données lorsque l'horloge (C) déclenche une opération d'écriture. Les données sont alors stockées dans le registre et disponibles à la sortie Q. Le registre fournit une sortie de 256 bits (Data(255:0)), qui sont ensuite connectés à l'entrée du multiplexeur (MUX:1). Le multiplexeur utilise la ligne de sélection port_select pour choisir un bit parmi les 256 disponibles, lequel est transmis à la sortie port_result. La sortie finale Yout correspond au bit sélectionné par le multiplexeur.

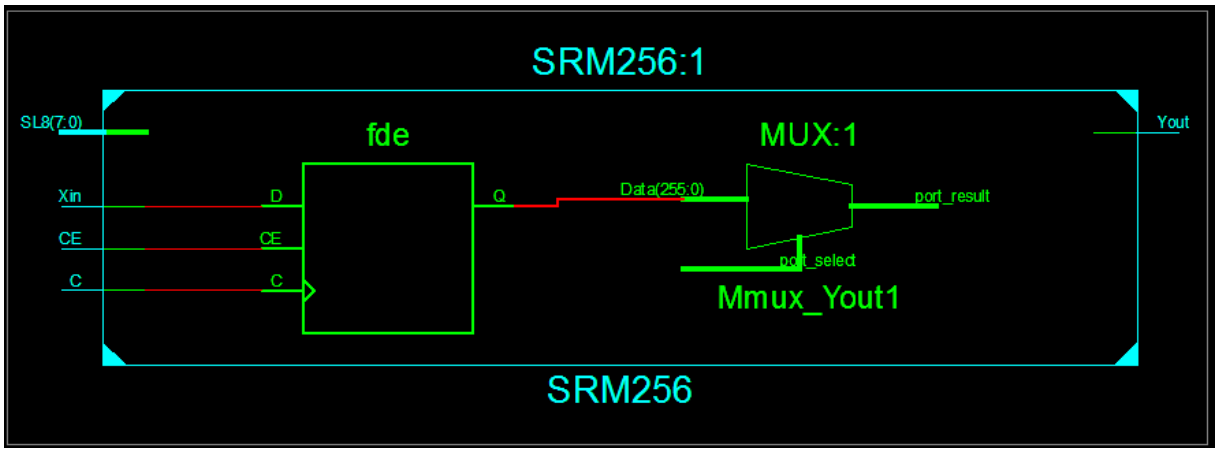

Figure 3.10 Résultat de synthèse registre multiplieur 16 bits

3.5.8 **Component registre multiplieur**

Le schéma représente probablement un registre multiplexeur pour 16 bits composé de 16 registres SRM256, chacun de 256 bits. Chaque bloc SRM256 correspond à un registre de 256 bits, avec des connexions typiques incluant une ligne

de contrôle (C), une ligne d'activation de puce (CE), et une entrée de données (Xin). Les blocs SRM256 sont multiplexés de manière à permettre la sélection et l'accès à des bits spécifiques parmi les 256 bits disponibles dans chaque registre. Le schéma de multiplexeur "SR_8_256:1" indique qu'il y a un mécanisme de sélection qui peut choisir une ligne parmi 256, permettant un accès efficace aux données dans les registres. Les sorties des registres, Yout, représentent les données sélectionnées qui sont transmises en sortie après le multiplexage. Les 16 blocs sont organisés pour gérer des données de manière coordonnée, permettant une manipulation complexe des données dans un système de 16 bits. Chaque bloc est capable de stocker 256 bits, et le système multiplexeur permet d'accéder à n'importe quel bit spécifique à travers les lignes de contrôle. Les données entrent dans les registres via Xin, les lignes de contrôle (C et CE) déterminent quel registre et quel bit particulier sont actifs, et le multiplexeur sélectionne les bits spécifiques à partir des registres actifs et les transmet vers la sortie Yout. En résumé, ce composant est conçu pour fournir une gestion sophistiquée des données à 16 bits, en utilisant 16 registres de 256 bits chacun, avec des capacités de sélection et de multiplexage pour un accès flexible et rapide aux données.

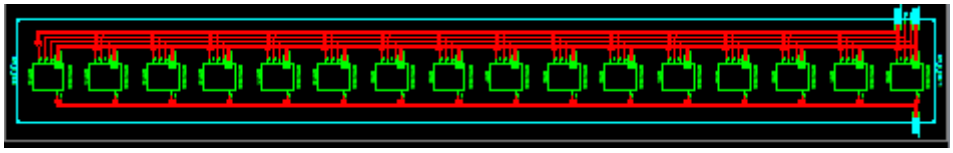

Figure 3.11 Résultat de synthèse component registre multiplieur 16 bits

3.5.9 **Implémentation registre**

Ce schéma représente un registre (RGT:1) dans un circuit numérique. La boîte étiquetée "fd" représente le registre lui-même, souvent abrégé pour un type de flip-flop, comme un flip-flop D. Les entrées et sorties du registre sont marquées par "d(15:0)" et "t(15:0)", respectivement, indiquant des bus de données de 16 bits, signifiant que 16 lignes de données sont utilisées pour l'entrée et la sortie. L'entrée de l'horloge, marquée "clk", contrôle le moment où les données d'entrée (d) sont capturées dans le registre. Les lignes de connexion "D", "C" et "Q" relient respectivement l'entrée de données,

l'horloge, et la sortie de données au registre. Le registre stocke les données numériques et se déclenche sur le signal d'horloge, transférant les données présentes sur l'entrée D (d) à la sortie Q (t) lors de chaque impulsion d'horloge, où elles sont conservées jusqu'à la prochaine impulsion.

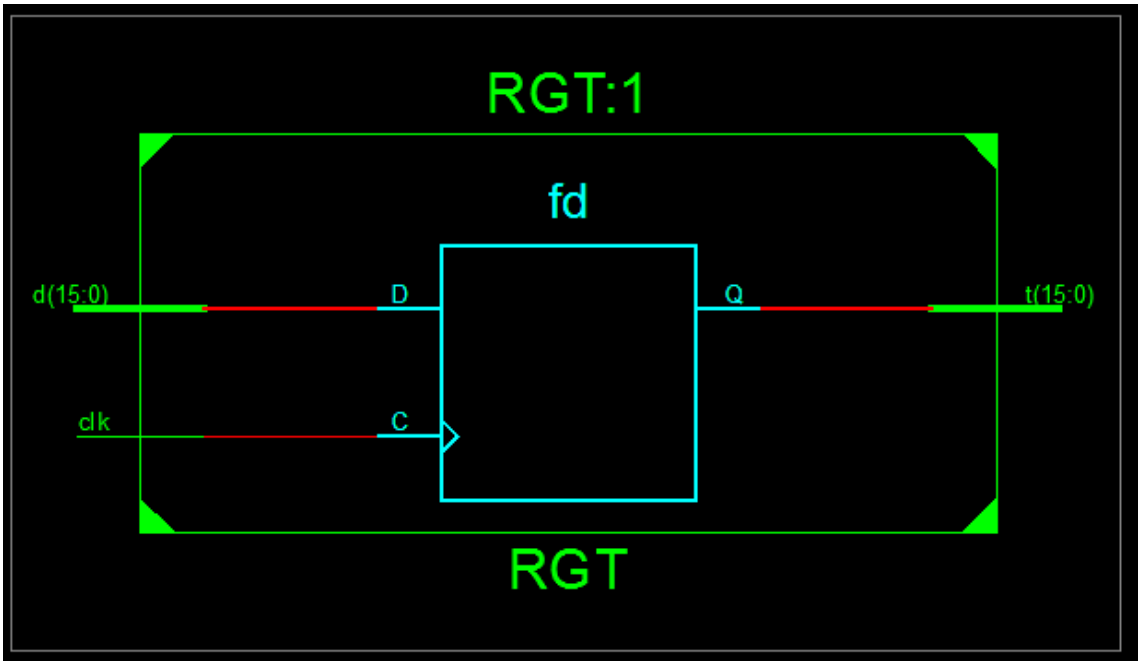

Figure 3.12 Résultat de synthèse registre 16 bits

3.5.10 **Component registre**

Ce schéma représente un coponant registre de 16 bits (RGT16:1) dans un circuit numérique. La boîte étiquetée "CPT" pourrait indiquer un composant de comptage. Les entrées et sorties du registre sont marquées par "d16(15:0)" et "t16(15:0)", ce qui indique des bus de données de 16 bits pour l'entrée et la sortie. Il y a également des connexions internes "d(15:0)" et "t(15:0)". L'entrée de l'horloge, marquée "clk", contrôle le moment où les données d'entrée sont capturées dans le registre. Les lignes de connexion "d", "d16", "clk", "t", et "t16" relient respectivement les entrées et sorties de données ainsi que l'horloge au registre. Les étiquettes vertes "RGT16:1" et "RGT16" identifient le module comme étant un registre de 16 bits, avec "RGT16:1" indiquant qu'il s'agit peut-être du premier registre de 16 bits dans une série ou un contexte spécifique du schéma global. Le registre stocke les données numériques de

16 bits et se déclenche sur le signal d'horloge, transférant les données présentes sur l'entrée "d" à la sortie "t" lors de chaque impulsion d'horloge et les conservant jusqu'à la prochaine impulsion.

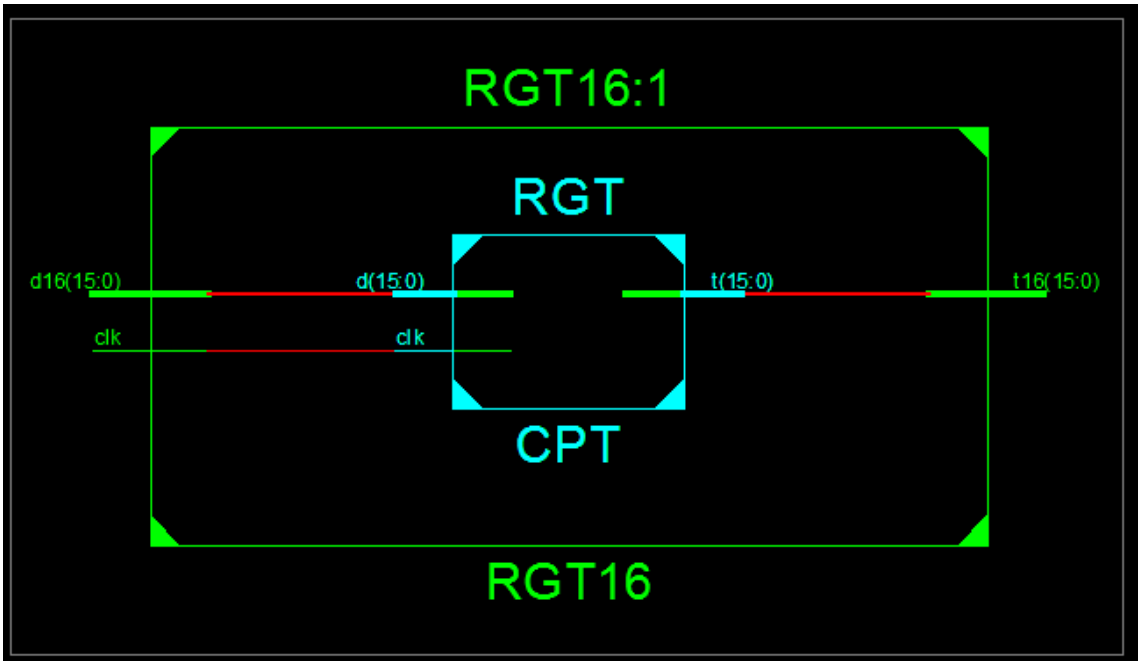

Figure 3.13 Résultat de synthèse component registre 16 bit

3.5.11 **Projet finale component de tous les composant**

Le schéma représente un projet final intégrant plusieurs composants interconnectés : COMPTEUR8, BRAM8, SR_8_256, DSP8 et RGT16, formant un système de 16 bits. Le compteur COMPTEUR8 génère une adresse à 16 bits basée sur une horloge et des signaux de contrôle, transmise à la mémoire RAM BRAM8 qui stocke et transmet les données en fonction de cette adresse.Les données de la RAM sont multiplexées par le registre SR_8_256 selon une ligne de sélection SL8, permettant de choisir des bits spécifiques parmi 256, puis sont traitées par le processeur de signal numérique DSP8. Ce dernier utilise des données d'entrée à 16 bits, les traite en fonction de signaux d'horloge et de réinitialisation, pour fournir en sortie des données finales également à 16 bits.Le registre RGT16 reçoit ensuite les données finales de DSP8 et les transmet selon les besoins du système. Ce système assure une gestion et un traitement flexibles des données à travers plusieurs étapes, facilitant une manipulation efficace dans un contexte de système 16 bits intégré.

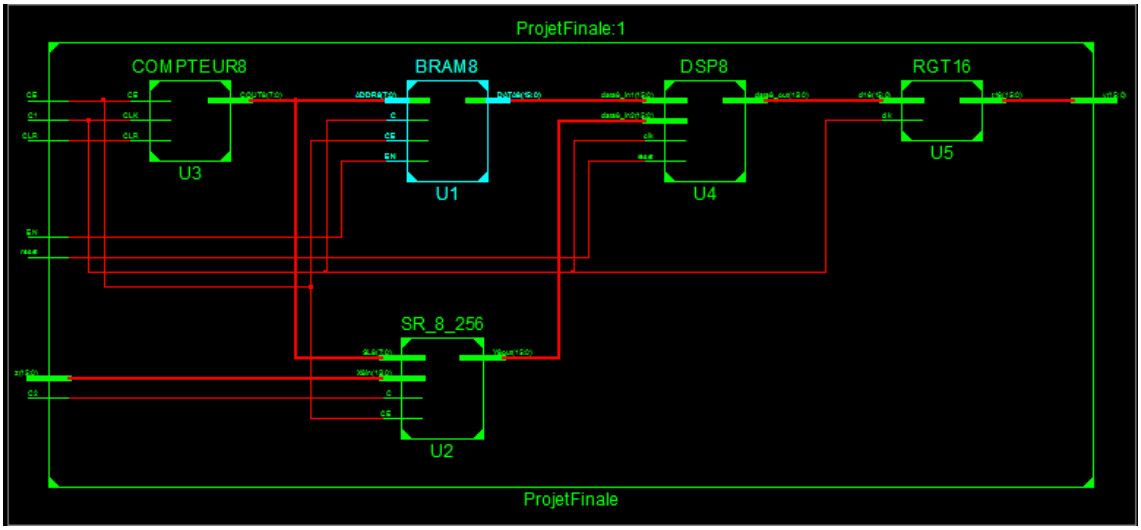

Figure 3.14 Résultat de synthèse component complet

Cette figure 3.15 représente le schéma final dans la carte FPGA

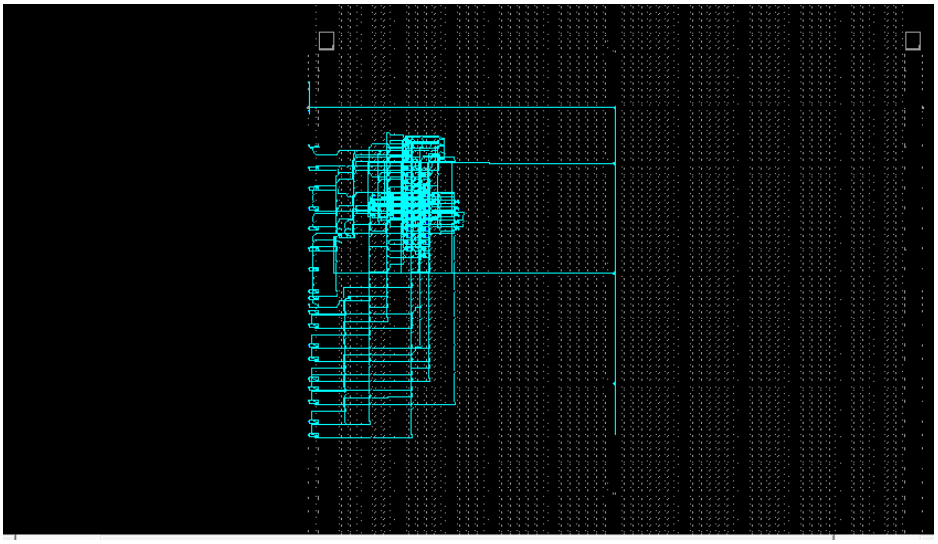

Figure 3.15 Design routé

3.6 Implémentation sur MATLAB

Après avoir terminé notre travail avec Xilinx (ISE 14.7), nous allons maintenant tester notre filtre dans System Generator .

Cette figure 3.16 illustre notre filtre dans Xilinx System Generator. Elle représente trois sous-systèmes : le premier est SR256_1 (un registre multiplexer), le second est DSP-BRAM-COMPTEUR (une implémentation VHDL de DSP-BRAM-COMPTEUR), et le dernier est un simple registre. Ces sous-systèmes sont connectés

entre eux via des boîtes noires représentant un registre D. En entrée, il y a un bloc de signal aléatoire qui génère des signaux pour le filtrage. On peut observer ces signaux avant et après le filtrage.

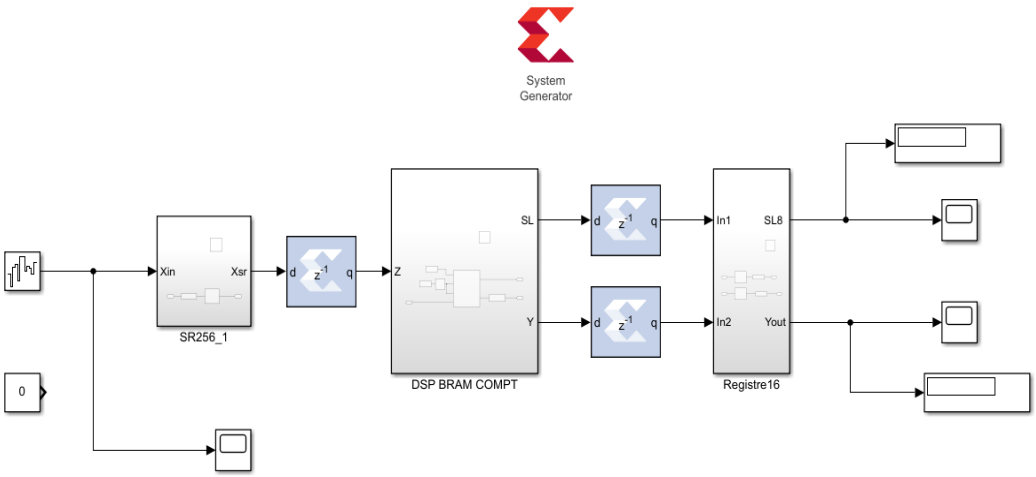

Figure 3.16 Schéma du circuit de simulation sur « Xilinx System Generator » avec Multiple Clocks

La figure 3.17 nous montre comment activer la multi clocks

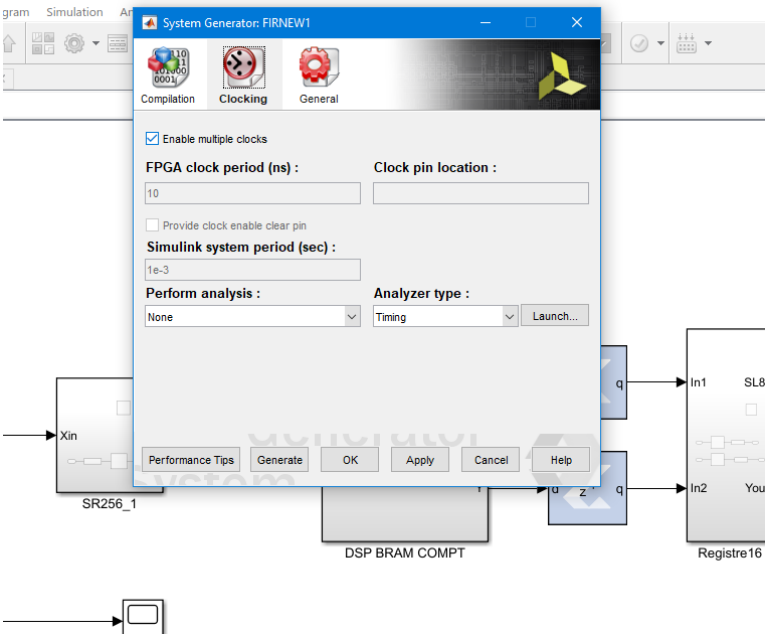

Figure 3.17 Activation du Multiple Clocks sur « Xilinx System Generator »

La figure suivante nous montrent le schéma détaillée de sub-system DSP-BRAM-COMPTEUR

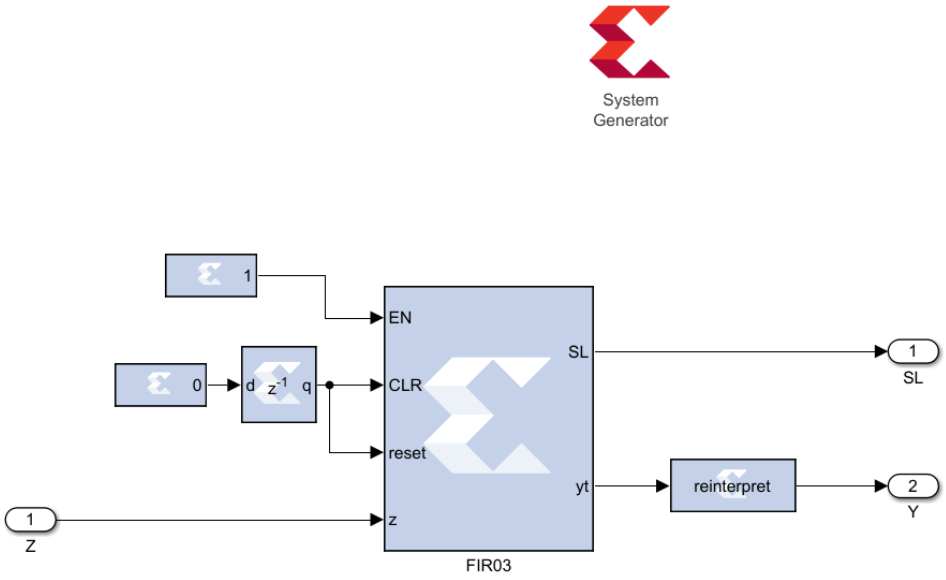

Figure 3.18 Schéma du module DSP BRAM Compteur « Xilinx System Generator » avec Multi Clock

3.7 implémentation et résultats

Maintenant on va configurer les deux fréquences une supérieure et une inferieure .

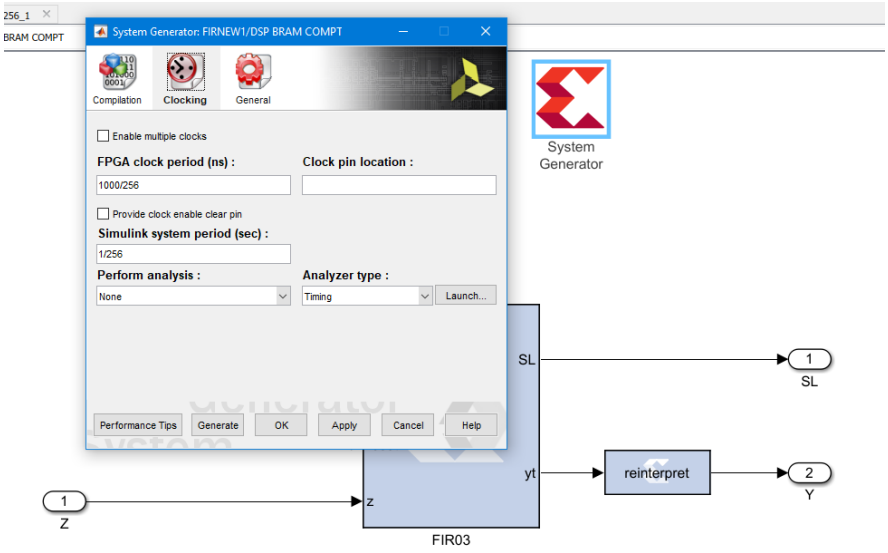

Figure 3.19 Configuration de la fréquence sup (F_DSP_RAM = 256xF_in_out) sur « Xilinx System Generator »

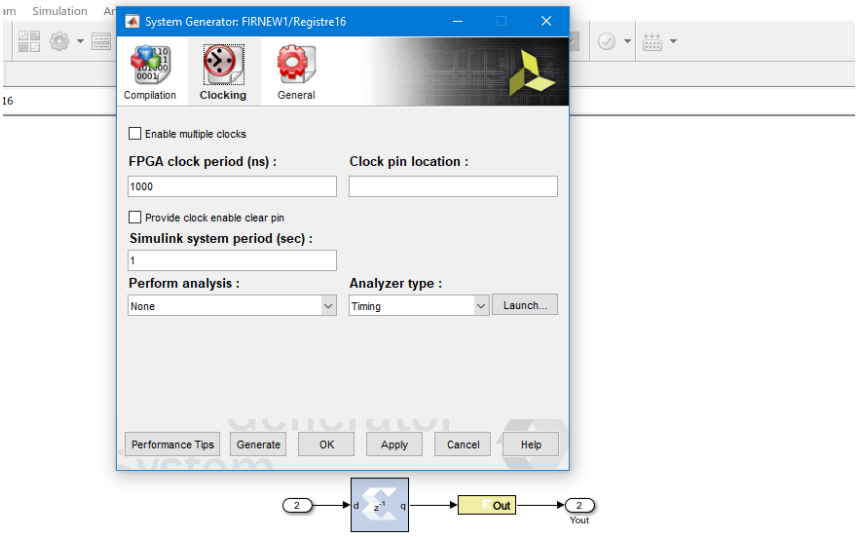

Figure 3.20 Configuration de la fréquence inf (F_in_out = F_DSP_RAM/256) sur « Xilinx System Generator »

et après la configuration des fréquences en lance notre test

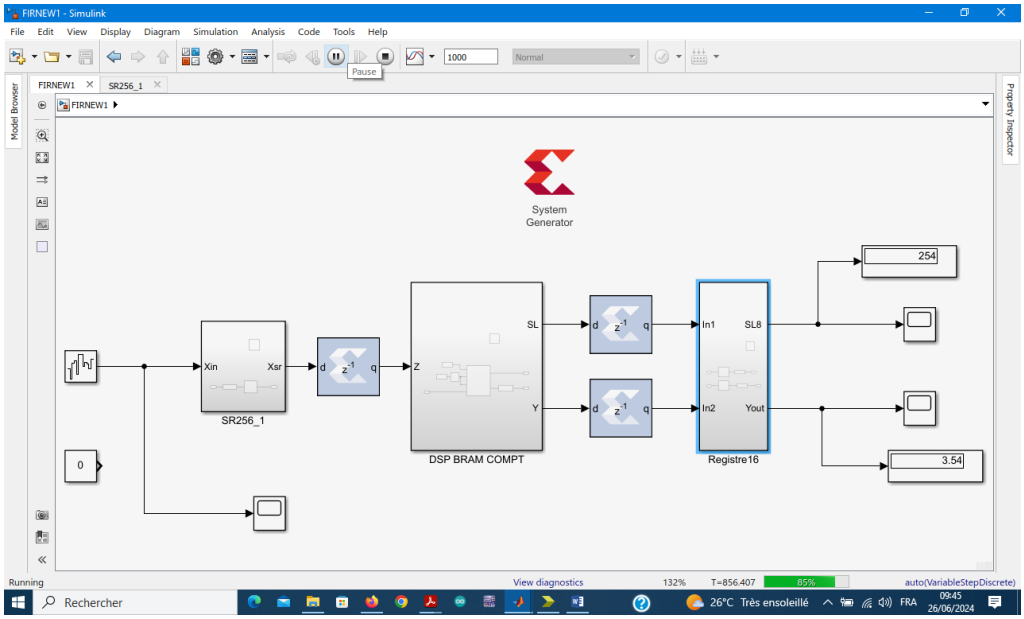

Figure 3.21 Capture d'écran pendant simulation sur « Xilinx System Generator » avec Multiple Clocks

Maintenant les résultats avant et après le filtrage

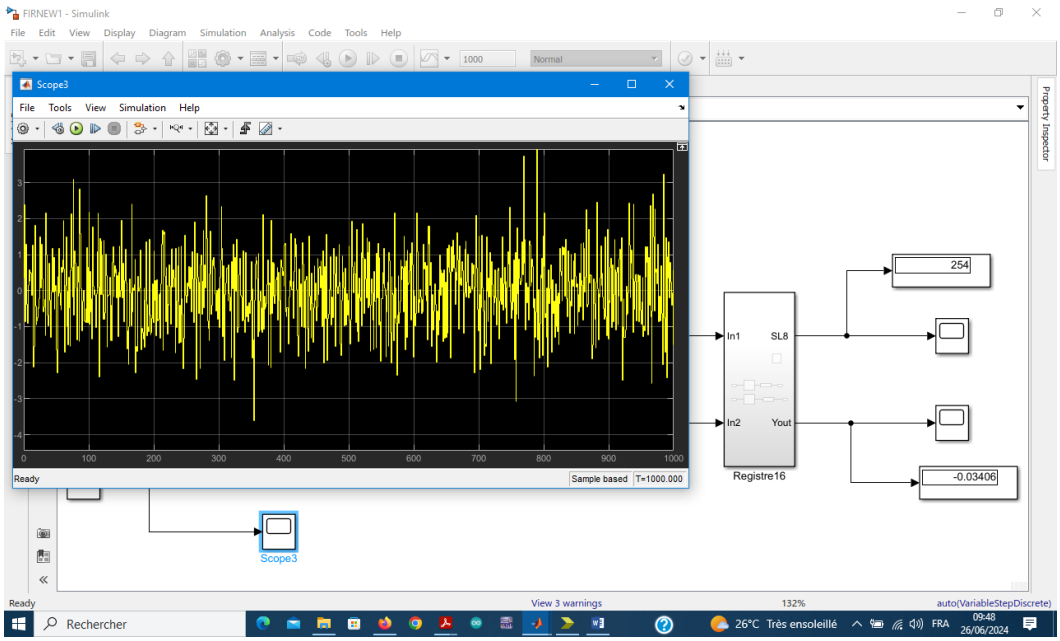

Figure 3.22 exemple d'un signal d'entrée sur « Xilinx System Generator » avec Multiple Clocks

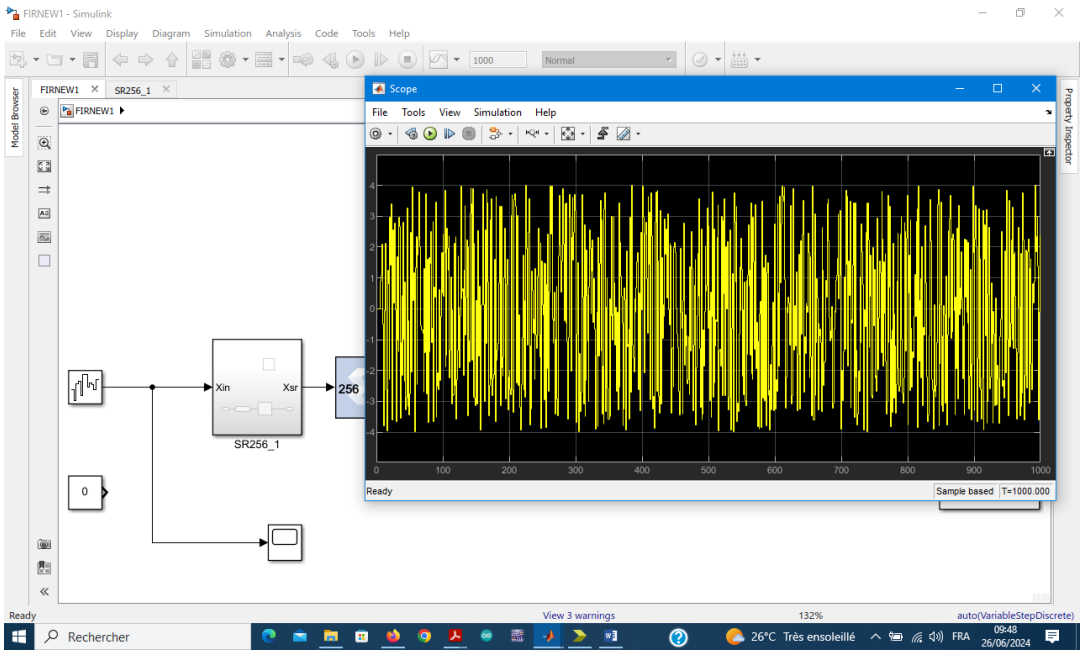

Figure 3.23 exemple d'un signal de sortie sur « Xilinx System Generator » avec Multiple Clocks

Après la simulation le signal de sortie est bien filtré par rapport le signal d'entrée et d'après cette résultat en conclus que notre filtre fonction bien .

3.8 Conclusion

En conclusion, les bons résultats obtenus lors de l'implémentation du filtrage FIR sur FPGA en réduisant l'utilisation logique de DSP et LUT. Les figures et les exemples de signaux montrent clairement l'amélioration des performances du système. L'utilisation efficace des composants BRAM contribue également à l'optimisation globale du système. Ces résultats positifs soulignent l'importance de l'implémentation FPGA pour des applications plus performantes dans le domaine des télécommunications.

Conclusion générale

Lors de la conception d'une structure de filtre FIR réconfigurable de haut ordre et performante pour une implémentation sur FPGA, nous avons proposé une approche équilibrée permettant une utilisation extrêmement réduite de la logique. Pour atteindre cet objectif, nous exploitons à la fois la vitesse et la structure du FPGA. Les BRAMs du FPGA, combinés aux slices DSP, sont personnalisés pour gérer les coefficients réconfigurables et assurer le processus de multiplication-accumulation FIR. De plus, les connexions internes des slices et des LUT sont optimisées pour le pipelining et la sélection des échantillons stockés en entrée.

Les résultats obtenus montrent qu'un filtre FIR symétrique réconfigurable à 256 taps nécessite moins de 1 % des BRAM, moins de 2 % des slices DSP48A1 et moins de 1 % des slices LUT lorsqu'il est implémenté sur un Xilinx Spartan-6 XC6SLX45. Ces chiffres sont révélateurs de l'efficacité de notre approche, démontrant que même avec une conception de haut ordre, les ressources FPGA sont utilisées de manière extrêmement efficace.

En plus de la réduction significative de l'utilisation des slices, la structure proposée offre une amélioration de l'efficacité par rapport à toutes les architectures de filtres FIR réconfigurables basées sur FPGA reconnues. La combinaison de l'utilisation optimale des BRAMs et des slices DSP, avec une architecture bien pensée pour le pipelining et la sélection des échantillons, permet de réduire considérablement la logique nécessaire, tout en maintenant ou en améliorant les performances globales du filtre.

Le principal avantage de notre architecture réside dans sa capacité à augmenter l'efficacité ainsi que la fréquence d'échantillonnage maximale en entrée, tout en

utilisant une logique FPGA extrêmement réduite. Cela signifie que non seulement notre filtre peut fonctionner à des fréquences plus élevées, mais qu'il le fait également en utilisant moins de ressources, ce qui est crucial pour des applications nécessitant des filtrages intensifs et à haute vitesse, sans compromettre la flexibilité ou la réconfigurabilité du système.

En conclusion, notre approche équilibrée et innovante pour la conception de filtres FIR réconfigurables sur FPGA représente une avancée significative dans le domaine, permettant une utilisation plus efficiente des ressources tout en offrant des performances supérieures. Cela ouvre la voie à des applications plus complexes et exigeantes en termes de traitement de signal, tout en conservant une empreinte matérielle réduite et une flexibilité accrue dans les déploiements futurs.

Bibliographie

[1] A,Guellal.«Les circuits FPGA ,description et applications».Définition,(2),1 1 .https://www.cder.dz/vlib/bulletin/pdf/bulletin_024_05.pdf

[2] Benkrid, A., Benkrid, K.: Novel area-efficient FPGA architectures for FIR filtering with symmetric signal extension. *IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst*. **17**(5), 709–722 (2009)

[3] O. Sentiey « Traitement numérique du signal » ENSSAT-Université de Rennes 1, IRISA-équipe de recherche R2D2, 2003.

[4] Wu, L., et al.: Design and implementation of an optimized FIR filter for IF GPS signal simulator, In: *2010 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia), Shanghai*, pp. 25-28 (2010)

[5] Jiang, H., et al.: A high-performance and energy-efficient FIR adaptive filter using approximate distributed arithmetic circuits. *IEEE Trans. Circuit. Syst. I*. **66**(1), 313–326 (2019)

[6] O. Sentiey « Analyse et synthese des filtres numeriques » ENSSAT-Université de Rennes 1, 28 mai 2008.

[7] Antoniou, A.: *Digital Filters: Analysis, Design, and Applications*. McGraw-Hill, New York (1993)

[8] Parhi, K.K.: *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, New York (1999)

[9] Kourgli,A.(2015). Traitement Numérique du Signal,neuropsychology,23(1),67-68.doi:10.13140/RG.2.1.2991.6000

[10] M.Mamoun (2021). Efficient FPGA based architecture for high-order FIR filtering using simultaneous DSP and LUT reduced utilization. PROPOSED RECONFIGURABLE FIRFILTER STRUCTURE,11 ,2-6. Doi :https://doi.org/10.1049/cds2.12043

[11] Peled, A., Liu, B.: A new approach to the realization of nonrecursive digital filters. *IEEE Trans. Audio Electroacoust.* **6**, 477–485 (1973)

[12] Peled, A., Liu, B.: A new hardware realization of digital filters. IEEE Trans. Acoust. Speech, Signal Process. 22(6), 456–462 (1974)

[13] Chen, C.F.: Implementing FIR filters with distributed arithmetic. IEEE Trans. Acoust. Speech, Signal Process. 33(5), 1318–1321 (1985)

[14] White, S.A.: Applications of the distributed arithmetic to digital signal processing: a tutorial review. IEEE ASSP Mag. 6(3), 5–19 (1989)

[15] Spartan-6 FPGA DSP48A1 Slice.UG389 (v1.2).Xilinx. 2014. url: https: //docs.xilinx.com/v/u/en-US/ug389 (cit. on p. 16).

[16] Spartan-6 FPGA Block RAM Resources.UG383 (v1.5).Xilinx. 2011. url: https://docs.xilinx.com/v/u/en-US/ug383 (cit. on pp. 14, 15).

[17] D. B. Thomas and W. Luk, “High quality uniform random number generation using LUT optimised state-transition matrices,” Journal of VLSI Signal Processing, vol. 47, no. 1, pp. 77–92, 2007

[18] Prost-Boucle,A.Muller,O. Rousseau,F.(2014). Outils et langages Gratuits et/ou open-source Pour les FPGA. Xilinx : ISE WebPack,25,8-9.

[19] Xilinx Inc.: *Spartan-6 FPGA DSP48A1 Slice, User Guide, UG389 (v1.2)* (May 2014)

[20] Xilinx Inc.: *UltraScale Architecture DSP Slice, User Guide, UG579 (v1.9)* (September 2019)

[21] https://matlab.fr.malavida.com/windows/