REPUBLIQUE ALGERIENNE DEMOCRATIQUE Et POPULAIRE

MINISTERE DE L'ENSEIGNEMMENT SUPERIEURE Et De La RECHERCHESCIENTIFIQUE

Université Dr .Yahia Farés de MEDEA

Faculté de la Technologie

Département de Génie Électrique

Mémoire De Fin D'études

Présenté pour l'obtention du Diplôme de Master en Télécommunication

Option : Systèmes Intelligents et Communicants

Thème:

Etude et Implémentation D'un IP core FFT sur une Platforme Reconfigurable

Réalisé par :

SOUSSI Nassima
ALLIK Iman

Encadré par :

Mr. BENGHERBIA Billel

ANNEE UNIVERSITAIRE: 2016/2017

ملخـص:

الهدف من هذه الدراسة هي تنفيذ IP CORE على الأرضية التكوينية FPGA , نبدأ أولا بسرد عموميات الأنظمة على الشريحة SOC و معالج المعلومات HARDCORE و SOFTCORE, و كذلك الدائرة المنطقية المبرمجة FPGA,و بعد ذلك نمر على عموميات FFT , حيث نتكلم على مختلف التصاميم ,وبعد ذلك ننهي عملنا بمختلف خطوات دراستنا بدأ بتقديم مخطط تفسيري لتشغيل الدارة ,ومحاكاة التصميم و في الأخير تنفيذها على دارة FPGA.
الكلمات المفتاحية : FFT, FPGA,SOC,

Résumé :

L'objectif de cette Etude et d'implémenter d'un Ipcore FFT sur une plateforme reconfigurable FPGA. Nous allons tous d'abord abordé des généralités sur les systèmes surpuce (SOC) et les processeurs Hardcore et Softcore ainsi que les circuits logiques programmables (FPGA). Après nous avons passées à la généralité sur la FFT d'où on a parlé des différentes architectures. Ensuite nous avons termine parlesdifférentes étapes de conception, commençons par une présentions schématique explicatif pour le fonctionnement de notre circuit, simulation de la conception et enfin implémentation sur circuit FPGA.

Mots clés : SOC, FFT, FPGA.

Abstract :

The objective of this project is to study and implement an Ip core FFT on a reconfigurable FPGA platform. First of all, we are going to discuss about the generality of the systems on the chip (SOC), and Hardcore & Softcore processors as well as Programmable logic circuit (FPGAs). Then, we are going through the generality of the **FFT** where we are going to talk about different architects. Lastly, we are going to see different tips of conception, starting by the presentation of the schematic illustration of the function of our system, simulation of the design, and finally implementation on the FPGA circuit.

Keywords: SOC, FFT, FPGA.

Sommaire

Introduction Générale ..... **Erreur ! Signet non défini.**

Chapitre I : Généralités 3

I.1. Introduction 4

I.2. Les Système sur puce (Soc) basent sur FPGA 4

I.1.1. Définition 4

I.1.2. Architecture du système SoC 5

I.1.3. Les Caractéristiques des Socs 6

I.1.4. Domaines d'application des SoCs 7

I.2. Les Circuit FPGA 7

I.2.1. Définition FPGA 7

I.2.2. Les Différentes **type** de FPGA 8

I.2.3. Les Technologies des FPGA 9

I.2.4 .Configuration et reconfiguration des FPGA 10

I.2.5. Application des FPGA ..... **Erreur ! Signet non défini.**

I.3.Les processeurs softcore ..... **Erreur ! Signet non défini.**

I.3.1.Définition : ..... **Erreur ! Signet non défini.**

I.4.Les processeurs Hardcore ..... **Erreur ! Signet non défini.**

I.4.1.Définition ..... **Erreur ! Signet non défini.**

I.6.Les Multi-Processor System-on-Chip ..... **Erreur ! Signet non défini.**

I.6.1. Définition ..... **Erreur ! Signet non défini.**

I.6.2. la structure de MPSoC ..... **Erreur ! Signet non défini.**

I.7. Les flux de conceptions FPGA_SoC ..... **Erreur ! Signet non défini.**

I.8. Codesign : Conception conjointe logicielle/matérielle ..... **Erreur ! Signet non défini.**

I.9. Conclusion ..... **Erreur ! Signet non défini.**

Chapitre II: Généralité sur l'algorithme FFT ..... **Erreur ! Signet non défini.**

II.1. Introduction ..... **Erreur ! Signet non défini.**

II.2. Présentation deLa transformée de Fourier .....**Erreur ! Signet non défini.**

II.2.1. La transformée de Fourier discrète.....**Erreur ! Signet non défini.**

II.2.2. La transformée de Fourier rapide .....**Erreur ! Signet non défini.**

II.3. Complexité arithmétique de l'algorithmes TFR .....**Erreur ! Signet non défini.**

II.4. Les algorithmes conventionnels de FFT .....**Erreur ! Signet non défini.**

II.4.1. Radix-2 .....**Erreur ! Signet non défini.**

II.4.2. Radix-4 .....**Erreur ! Signet non défini.**

II.5. Le bit-renverse .....**Erreur ! Signet non défini.**

II.6. Architectures conventionnelles de la FFT .....**Erreur ! Signet non défini.**

II.6.1. L'architecture en pipeline .....**Erreur ! Signet non défini.**

II.6.2.Architectures à mémoires.....**Erreur ! Signet non défini.**

II.7. Domaines d'utilisation de la FFT .....**Erreur ! Signet non défini.**

II.8. Conclusion .....**Erreur ! Signet non défini.**

Chapitre III: Conception et Implémentation.....**Erreur ! Signet non défini.**

III.1 Introduction .....**Erreur ! Signet non défini.**

III.2 les blocs principaux de fonctionnement de la FFT.....**Erreur ! Signet non défini.**

III.2.1 Le processeur de contrôle (Microblaze) .....**Erreur ! Signet non défini.**

III.2.2 Fast Fourier Transform(FFT).....**Erreur ! Signet non défini.**

III.2.3 AXI Streaming FIFO .....**Erreur ! Signet non défini.**

III.2.4 SPI.....**Erreur ! Signet non défini.**

III.2.5 AXI Interconnect .....**Erreur ! Signet non défini.**

III.2.6 AXI UART Lite .....**Erreur ! Signet non défini.**

III.2.7 PmodAD1 .....**Erreur ! Signet non défini.**

III.2.8 la Réalisation de la FFT sur vivado .....**Erreur ! Signet non défini.**

III.3 Les résultats de la consommation des architectures internes de Bloc FFT**Erreur ! Signet non défini.**

III.4 Partie pratique : génère le signal.....**Erreur ! Signet non défini.**

III.4.1 Waveforms ..... **Erreur ! Signet non défini.**
III.5 Conclusion 60
Conclusion générale 61
Notations 63
Annexe 64
Bibliographie 69

Liste des acronymes

AMBA : **A**dvanced **M**icrocontrôleur **B**us **A**rchitecture

ASIC: **A**pplication **S**pécific **I**ntegrated **C**ircuit.

BRAM: **B**loc **R**andom **A**ccess **M**emory

CPU: Central **P**rocessing **U**nit.

DFT: **D**iscrete **F**ourier **T**ransform.

DIF: **D**ecimation **I**n **F**requency.

DIT: **D**ecimation **I**n **T**ime.

DMA:**D**irect **M**emory **A**ccess.

DRAM: Dynamic **R**andom **A**ccess **M**emory

DSP:**D**igital **S**ignal **P**rocessor.

FF:**F**lip **F**lop.

FFT: **F**ast **F**ourier **T**ransform.

FPGA: **F**ield **P**rogrammable **G**ate **A**rrays.

IO: **I**nput **O**utput.

LUT:**L**ook **U**p **T**able.

MCS:**M**icro **C**ontrol **S**ystems.

MDC:**M**ulti-path **D**elay **C**ommutator.

MIMO: **M**ultiple **I**nput **M**ultiple **O**utput

MISO: Master **I**N **S**lave **O**UT

MOSI: Master **O**UT **S**lave **I**N

MPSOC : **M**ulti-**P**rocesseur **S**ystème **O**n **C**hip.

PLD:**P**rogrammable **L**ogic **D**evice.

ROM: **R**ead **O**nly **M**emory.

SCK: Serial **C**lock

SDC:**S**ingle-path **D**elay **C**ommutator.

SDF:**S**ingle-path **D**elay **F**eedback.

SDK: **S**oftware **D**eveloppement **K**it

SOC : **S**ystème **O**n **C**hip.

SRAM:**S**tatic **R**andom **A**ccess **M**emory.

SS: **S**lave Select

VHDL:**V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage.

Liste des figures

Figure I.1 *Conception SOC*……5

Figure I.2 *Système on chip* ……5

Figure I.3 *Les composants hétérogènes formants un SoC*……6

Figure I.4*carte FPGA*……8

Figure I.5 *Structure FPGA*……8

Figure I.6 *SRAM based FPGA*……9

Figure I.7*Caractéristiques des technologies FLASH*……9

Figure I.8 *Caractéristiques des technologies ANTI-FUSIBLES*……10

Figure I.9 *Reprogrammable sur site d'un FPGA*……10

Figure I.10 *Generic système on chip Architecture* ……11

Figure I.11 *Architecture du processeur Xilinxmicroblaze soft processeur cor* ……12

Figure I.12 *Architecture processeur Actel/MicrosemiSmartFusion Cortex-M3 Hard processeur cor*……13

Figure I.13*Multi-processeur Soc*……13

Figure I.14 *Architecture du MPSoC*……14

Figure I.15 *Flux de conception FPGA-SOC*……15

Figure I.16*L'architecture de codsigne*……17

Figure II.1.*Diagramme de l'opération Papillon radix-2*……25

Figure II.2.*Diagramme papillon déployé pour une FFT N=16*……27

Figure II.3. *Diagramme papillon radix-2 à entrelacement fréquentiel*……27

Figure II.4.*Diagramme papillon radix-4*……28

Figure II.5.*Exemple d'opération bit-renverse*……29

Figure II.6. *Architectures de TFR en pipeline a) à 8 points radix-2 MDC b) à 64 points radix-4 MDC c) à 64 points radix-4 SDC*……31

Figure II.7.*FFT à 8 points radix-2 SDF*……..…….32

Figure II.8. *Architecture en pipeline-parallèle SDF mixed-radix-*$2^3/2^4$*de 128 points pour l'UWB*……..33

Figure II.9**.** *Architecture à base de mémoires*……35

Figure II.10*. Différentes configurations des modules de calculs des architectures à mémoires*……..36

Figure III.1 *schéma des blocs principaux de fonctionnement de FFT*……41

Figure III.2*Architecture interne du MicroBlaze[13]*……42

Figure III.3*Architecture de core FFT*……43

Figure III.4 *Diagramme AXI4 Stream FIFO*……44

Figure III.5*Liaison SPI Maitre-Esclave*……45

Figure III.6 *Diagramme de Core AXI Interconnect*……45

Figure III.7 *Diagramme de CoreAXI UART Lite*……46

Figure III.8 *Le PmodAD1*……47

Figure III.9 *LaRéalisation de la FFT sur vivado*……48

Figure III.10 *Interface waveforms*……51

Figure III.11 *génère un signal* sinusoïdal à partir de waveforms……53

Figure III.12 *matériels utilises*……54

Figure III.13*Calcul de la FFT avec le logiciel Hercules*……56

Figure III.14 *signal sinusoïdal choisi signal de 10kHz et 500 HZ*……57

Figure III.15 *les résultats obtenus après les calculs de la FFT du signal sinusoïdal choisi*……57

Figure III.16*signal triangulaire choisi de 10 kHz et de 500HZ*……58

Figure III.17 *les résultats obtenus après les calculs de la FFT du signal triangulaire choisi*……..*58*

Figure III.18*signaux caret choisi de 10 kHz et de 500HZ*……..……..*59*

Figure III.19 *les résultats obtenus après les calculs de la FFT du signal caret choisi*…… ..*59*

Liste des Tableaux

Tableau II.1*Complexité arithmétique des algorithmes pour un multiplieur 4M2A……..…..21*

Tableau II.2.*Complexité arithmétique des algorithmes pour un multiplieur 3M3A……..…..22*

Tableau II.3.*Nombre de multiplications/additions non triviales pour un multiplieur 4M2A……22*

Tableau II.4.*Nombre de multiplications/additions non triviales pour un multiplieur 3M3A……23*

Tableau II.5.*Gain apporté par l'écriture de la DFT sous forme radix-2……..…..26*

Tableau II.6.*Dénombrement des opérations entre une DFT et une FFT radix-2……..26*

Tableau II.7.*Ressources matérielles selon l'architecture en pipeline utilisée……..…..34*

Tableau II.8.*Ressources matérielles et fréquences d'opération des architectures à base de mémoire.……37*

Tableau III.1.*Consommation des ressources de FFT_ Radix-2…… ……..…….49*

Tableau III.2.*Consommation des ressources de FFT_ Radix-4……49*

Tableau III.3.*Consommation des ressources de FFT_ Pipelined……..…..50*

Tableau III.4. *Présentation des éléments de la figureInterface waveforms……52*

Tableau III.5 *Calcul du pourcentage d'erreur……60*

Introduction Générale

Les systèmes sur une puce ("System on Chip" ou SoC) sont désormais une réalité et leurs utilisation se répand, notamment grâce aux technologies reprogrammables (SoPC : System on Programmable Chip) de type FPGA.

La réalisation de ces systèmes repose sur une nouvelle approche de conception qui consiste à associer dans un même composant, des cœurs de processeurs exécutant des programmes et des blocs fonctionnels spécifiques appelés IP (IntellectualProperty).Avec l'évolution des technologies FPGA, des architectures SoPC reconfigurables (matériel et logiciel) permettent d'améliorer la souplesse d'utilisation et la compacité des systèmes. La conception de ces systèmes mixtes logiciel/matériel (hard/soft) nécessite d'introduire de nouvelles méthodologies, traitant des spécifications soft et des blocs matériels synthétisables, et de la Co vérification**Erreur ! Source du renvoi introuvable.**0

Dans ce contexte, le FPGA (Field Programmable GateArray) avec ses grandes capacités d'intégration et de reconfiguration en font un composant clé pour développer rapidement des prototypes. Dans l'objectif d'encourager la large diffusion de ce type de circuits, il est nécessaire d'améliorer les environnements de développement pour les rendre plus accessibles à des non experts en électronique. (Conception d'un outil de prototypage rapide sur le FPGA pour des ...).

Notre travail s'insère dans ce cadre qui est basé sur l'étude et l'implémentation d'un IP core FFT sur une plateforme reconfigurable.

Ce mémoire est partagé en trois chapitres :

Le premier chapitre présente les différentes notions de base liées aux systèmes sur puce, et une étude détaillée sur le FPGA et les processeurs (soft / hard) core et muli-processeurs ainsi que les flux de conception FPGA.

Dans le deuxième chapitre nous présentons des généralités sur l'FFT (Fast Fourier Transform), on a étudié les algorithmes conventionnels les plus communs: Radix-2, Radix-4. Nous donnons ainsi un aperçu sur la complexité arithmétique des algorithmes de l'FFT. Finalement nous présentons les différentes architectures conventionnelles de l'FFT.

Finalement, Tandis que le dernier chapitre dernière partie, celle-ci était consacré à la réalisation pratique de l'implémentation sur un circuit FPGA et de la FFT générée par Microblaze. Pour cela nous avons commencé par la maitrise du langage de développement VHDL. Le profit de cette étape était de faire un passage d'un schéma de description d'une

architecture vers le programme de description de cette architecture. Puis nous sommes passés à la prise en main de l'outil de développement **vivado**qui fournit un environnement complet de développement d'application à base de circuits FPGA.

Après avoir maitrisé le langage VHDL et l'outil vivado nous sommes passés au calcul de la FFT où nous avons choisis des signaux et on les a fait passer dans un convertisseur analogique qui fera l'échantillonnage et les calculs ainsi que le stockage dans la mémoire et il conclura par l'affichage des résultats en Matlab pour la vérification de ces derniers.

.

Chapitre I : Généralités

I.1. Introduction

Les premiers systèmes sur puce ont vu le jour au milieu des années 90. Un SOC intègre tous les aspects d'un système sur un seul circuit intégré.Avec l'évolution des systèmes sur puce (SoCs) et l'augmentation de leur complexité, la possibilité d'introduire des changements après la fabrication devient intéressante. L'intégration d'un FPGA embarqué (FPGA) dans un SoC permet de réutiliser une portion de la puce. Grâce à sa reconfigurabilité facile et illimitée un FPGA est capable d'implanter des circuits spécialisés qui aide à améliorer les performances du système [10]

Dans le chapitre 1 de ce manuscrit et qui matérialise notre premier contact avec ce projet, nous allons tout d'abord présenter et définir les éléments utilisée dans notre projet. Ce chapitre est consacré à l'introduction de certaines notions de base liées aux systèmes sur puce, et plus précisément les systèmes sur puce (SoC).en outre on va décrire les différents éléments utilisés en détaille le FPGA qui a été mis en œuvre dans notre projet.

I.2. Les Système sur puce (Soc) à base d'FPGA

I.2.1. Définition

Un système monopuce, appelé encore SoC (System-on-Chip) ou système sur puce, Ce mot provient du verbe 'synistanai' qui veut dire combiner, établir, rassembler. Il désigne l'intégration d'un système complet sur une seule pièce de silicium. Une définition plus appropriée de système monopuce serait : Un système complet sur une seule pièce de silicium, résultant de la cohabitation sur silicium de nombreuses fonctions déjà complexes en elles-mêmes : processeurs, DSP, mémoires, bus, convertisseurs, blocs analogiques, etc. Il doit comporter au minimum une unité de traitement de logiciel (un CPU) et doit dépendre d'aucun (ou de très peu) de composants externes pour exécuter sa tâche[2]

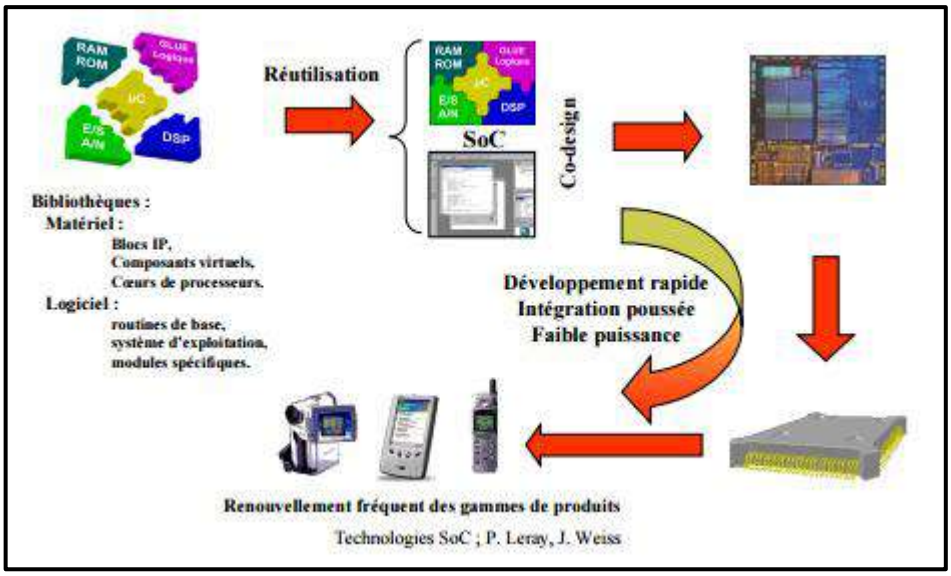


Figure I.1 *Conception SOC*

I.2.2. Architecture du système SoC

Les systèmes Soc se compose d'un cœur de processeur CPU (central processing unit), d'un processeur de signal numérique DSP (Digital Signal Processor), de la mémoire embarquée, et de quelques périphériques tels qu'un DMA (drainage minier acide) et un contrôleur d'E/S.

- ✓ Le CPU peut exécuter plusieurs tâches via l'intégration d'un OS.
- ✓ Le DSP est habituellement responsable de décharger le CPU en faisant le calcul numérique sur les signaux deprovenance du convertisseur A/N. (mpc.pdf)[2]

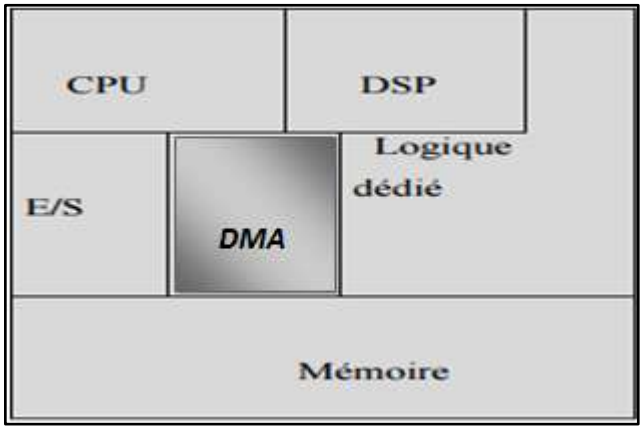


Figure I.2 *Système on chip*

I.2.3. Les Caractéristiques des Socs

I.2.3.1. *Hétérogénéité des SoCs*

Un Système sur puce est par définition, est un système hétérogène du fait qu'il regroupe généralement au sein de sa partie électronique ou numérique des composants de différentes natures.

Prenant l'exemple du téléphone portable qui présente un exemple quotidien de tels systèmes hétérogènes ou la partie analogique (radio) et la partie numérique (traitement de signal) cohabitent sur la même puce.**[2]**

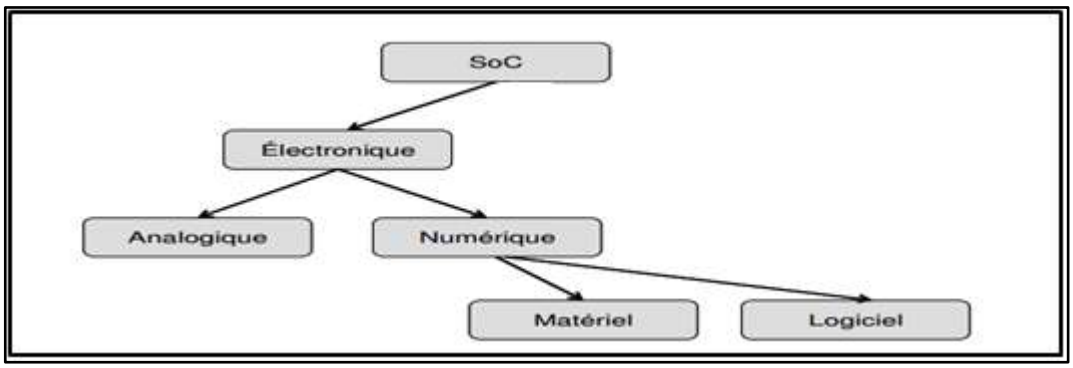


Figure I.3 *Les composants hétérogènes formants un SoC*

I.2.3.2. Embarquement des SoCs

Les systèmes sur puce sont des systèmes embarqués dans le sens où ils sont toujours appelés à évoluer dans le contexte d'un système plus large qui constitue l'environnement du système mono-puce, cela dans le but d'avoir des fonctions particulières. L'interaction entre le SoC (numérique) et les éléments de l'environnement (capteurs et actionneurs) se fait généralement via des convertisseurs analogique/numérique. [2]

I.2.3.3. Spécification des SoCs

Les systèmes sur puce ciblent toujours une application particulière bien définie, à la différence des systèmes a usages généraux tel que les ordinateurs, on parle souvent de SoC 15 spécifique. Cette caractéristique a fait la différence au niveau de la méthodologie de conception entre les systèmes mono-puce et les systèmes à usage général.[2]

I.2.4. Domaines d'application des SoCs

Le domaine d'application couvert par des systèmes multiprocesseurs sur puce est très vaste, puisqu'il comprend toutes les applications dont les impératifs économiques et les contraintes de conception nécessitent une architecture mixte, miniature et efficace. Les applications télécommunications et multimédias sont les plus représentatives de ce domaine. Elles constituent l'un des marchés le plus en expansion actuellement (modems, téléphones mobiles, les décodeurs audio et vidéo, etc.), notamment par le développement des télécommunications sans fil et d'Internet. Il constitue l'une des plus intéressantes applications, car il nécessite la production au moindre coût de systèmes à contraintes strictes (performance, poids, consommation, etc.)[2]

I.3. Les Circuit FPGA

I.3.1. Définition

Un FPGA est un type spécial de PLD (Logiciel Programmable).

Les FPGAs sont une catégorie de circuits reconfigurables (circuits qui peuvent être reconfigurés après leur fabrication). Ces circuits reconfigurables sont apparus comme une solution au manque de flexibilité dont souffrent les SoCs classiques. Les FPGAs peuvent être utilisés pour implémenter n'importe quelle fonction logique que les ASICs peuvent implémenter. Leur reconfiguration, qui peut être effectuée un nombre arbitraire de fois, représente l'un de leurs avantages majeurs par rapport aux ASICs[3]

Ils ont pour avantages

- ✓ la conception rapide et la reconfiguration.
- ✓ permet aux circuits reconfigurables de changer de configuration selon l'application.
- ✓ La possibilité d'effectuer la reconfiguration durant l'exécution de l'application



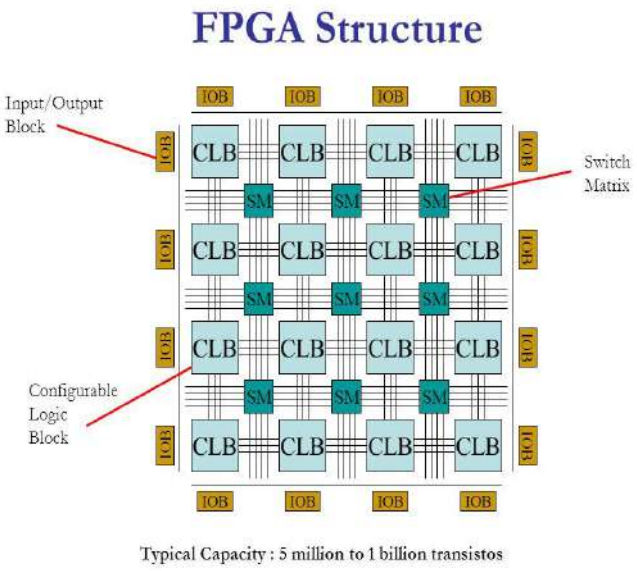


Figure I.4*Carte FPGA***Figure I.5** *Structure FPGA*

I.3.2. Les Différentes types de FPGA

Les PLD sont configurés par le téléchargement d'une séquence de valeurs numériques[4] (0 ou 1) contenu dans un fichier appelé bitstream. Ces valeurs sont stockées dans la mémoire du circuit, où la mémoire peut être volatile ou non-volatile.

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

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

I.3.3. Les Technologie des FPGA

I.3.3.1. FPGA à base des SRAM

utilisant les mémoires statistiques car la configuration des FPGA est charger dans une mémoire statistique .ils sont les FPGA les plus répondus [5].

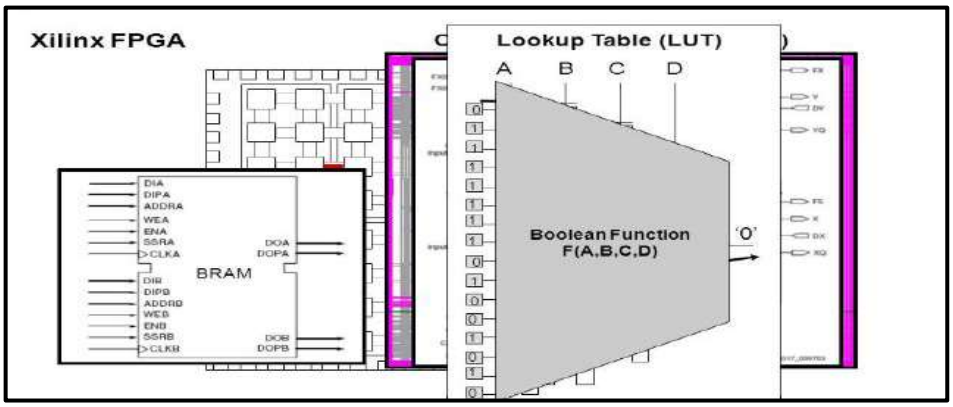


Figure I.6 *SRAM basé sur FPGA*

I.3.3.2. FPGA à base des Flash

Ce type de mémoire peut être configure via une source externe .ils consommes moins que ceux équipes d'une mémoire statistique (FPGA à base des SRAM). [5]

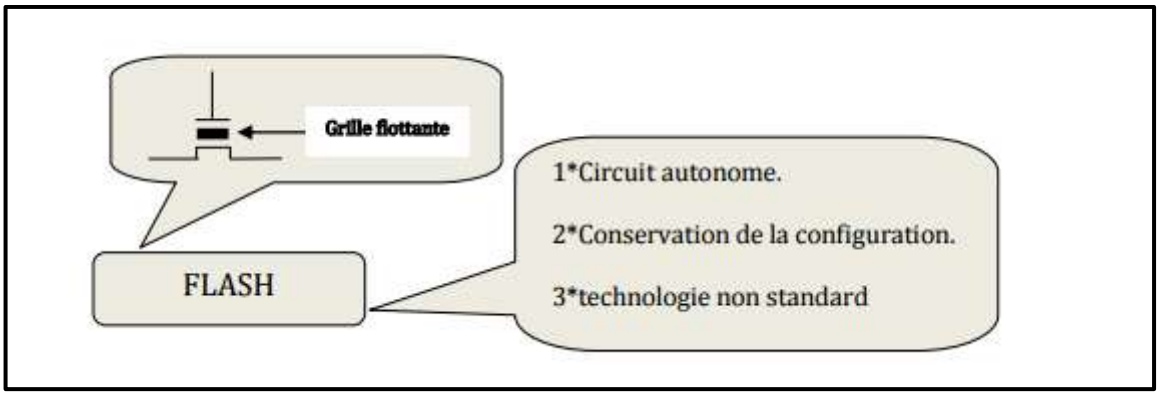


Figure I.7*Caractéristiques des technologies FLASH*

I.3.3.2. FPGA à base des Antifuse

Ne sont pas équipés d'une mémoire statistiques est gardés leurs configuration même quand ilsne sont pas alimentes .le principale avantages de cette technologie est qu'elle consomme

moins et aussi tolérante aux radiations, c'est pourquoi elle est principalement utiliser dans le domainemédicales.

Elles sont aussi compose d'une série de fusibles et ils sont principalement utiliser dans le domaine spéciale ces fusible sont brule lors de la configuration des FPGA .il n'est donc pas possible de le reprogrammer.[5]

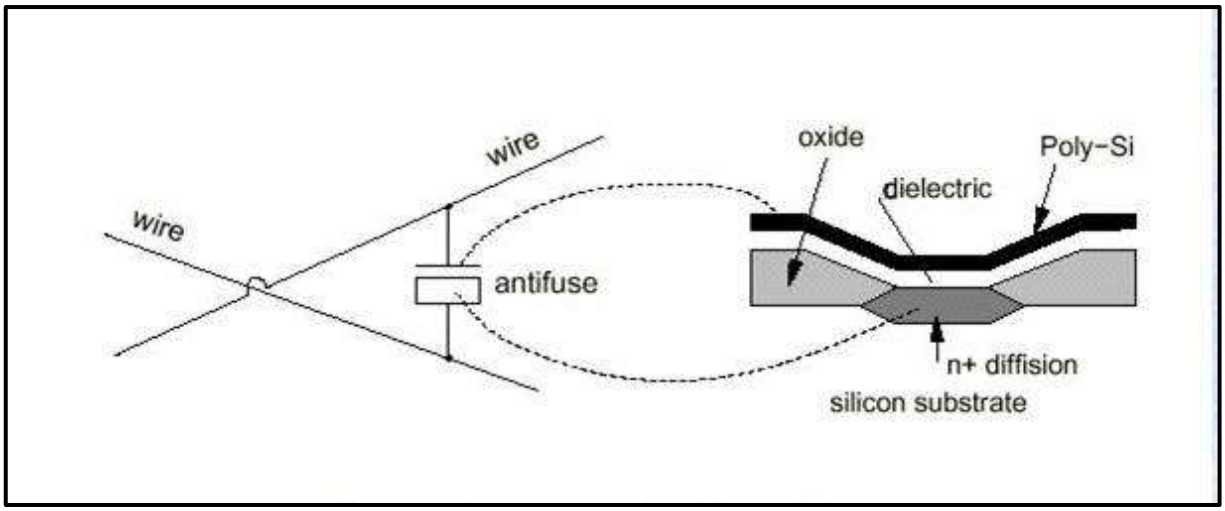


Figure I.8 *Caractéristiques des technologies ANTI-FUSIBLES*

I.3.4 .Configuration et reconfiguration des FPGA

Un système reconfigurable est un système qui est constitué de composants ou entités à architecture modifiable afin de répondre à un objectif bien déterminé. Ce systèmereconfigurable dispose d'un mécanisme permettant de choisir une nouvelle configuration et de la mettre en place dans le cadre du processus de reconfiguration.

Les circuits FPGA sont un type de ces circuits reconfigurables. Ils sont programmables ou configurables sur les cartes sur quelles ils sont implantés par l'utilisateur. Cette reconfiguration est une propriété nécessaire face aux systèmes à charges et contraintes Variables. ce qui est décrit dans la figure suivante.[5]

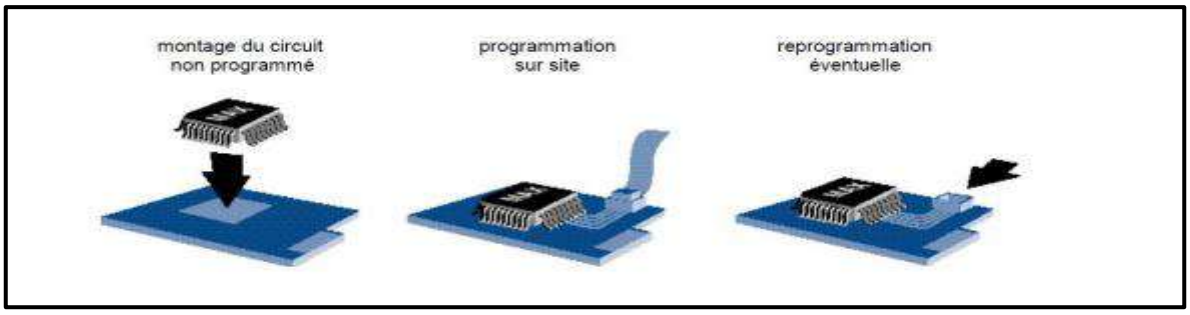


Figure I.9 *Reprogrammable sur site d'un FPGA*

I.3.5.Application des FPGA

Les FPGA sont utilisés dans de nombreuses applications, parmi eux :

- Prototypage de nouveaux circuits.
- Fabrication de composants spéciaux en petite série.
- Adaptation aux besoins rencontrés lors de l'utilisation.
- Systèmes de commande à temps réel.
- DSP (Digital Signal Processor)
- Imagerie médicale.**[6]**

I.4.Les processeurs softcore :

I.4.1.Définition :

Un processeur softcore est un processeur implémenté en langage de description haut niveau (VHDL, Verilog, etc.) sur un réseau logique reprogrammable comme un FPGA.

Cependant il existe un certain nombre de processeurs "softcore" sur lesquels des codes s'exécutent plus ou moins efficacement. Certaines familles de FPGA supportent une fonctionnalité de reconfiguration dynamique qui ouvre de nouvelles possibilités dans la mise en place de systèmes matériels[7]

Ils sont utiliséscar :

- Architecture flexible : Mise à jour
- Le code source peut être librement distribué et implanté dans n'importe quel circuit programmable FPGA
- Portabilité vers n'importe quel circuit FPGA : indépendance vis-à-vis du choix technologie

- Il est privilégie pour bénéficier des évolutions apportées en refaisant une synthèse

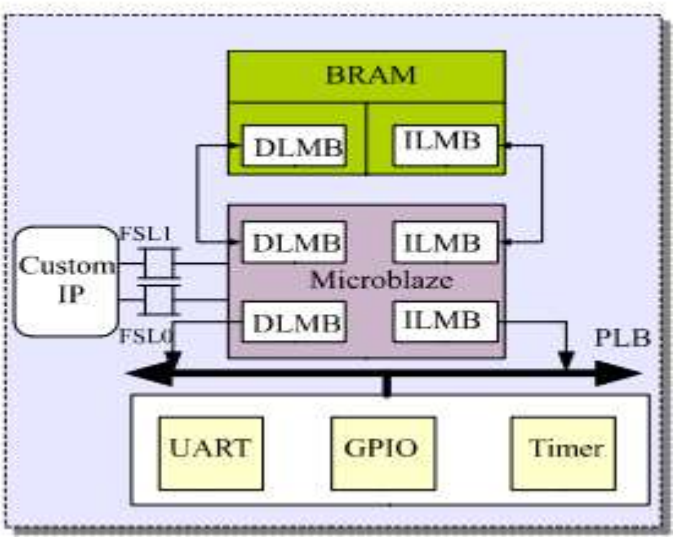


Figure I.10 *Architecture du processeur Xilinx microblaze soft processeur cor*

I.5.Les processeur Hardcore :

I.5.1.Définition :

Le processeur Hard-core désigne un système qui ne peut pas être reconfiguré et tous les composants sont déjà fixés par le fabricant et intégrés dans une carte de développement. Par contraste soft

Core se réfère à un système sur une puce programmable (SOPC). Dans un SOPC, le processeur mémoires et composants sont créés en utilisant les ressources disponibles dans une logique programmable et il peut être personnalisé en fonction d'un ensemble particulier de spécifications.

Il existe plusieurs types de processeur Hardcore, Le processeurPowerPC405 de Xilinx est le processeur intègre pour les cartes de cette famille qui a été développe à l'origine par la société Xilnix…..[7]

Les systèmes embarqués concilient le processeur hard core comme :

- ✓ Le processeur est très souvent le cœur du système
- ✓ Les fonctions critiques sont confiées à des circuits spécialisés
- ✓ Processeurs spécialisés (DSP, ASIP)

✓ Architecture matérielles (décodeurs MP3, MPG2, etc.)

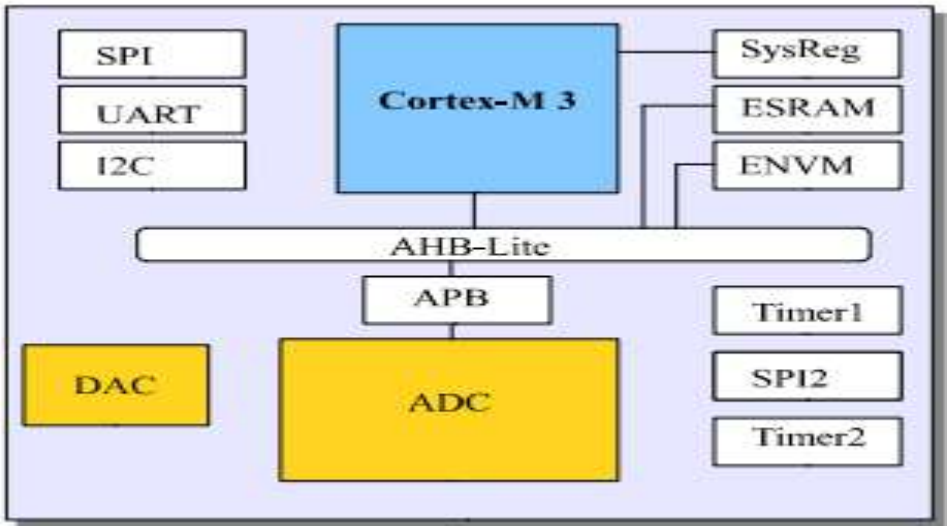


Figure I.11 *Architecture processeur Actel/MicrosemiSmartFusion Cortex-M3 Hard processeur cor*

I.6.Les Multi-Processor System-on-Chip

I.6.1. Définition

Un système multiprocesseur sur puce MPSoC est un système complet intégrant sur une seule pièce de silicium plusieurs composants complexes et hétérogènes tels que des unités de calcul spécifiques programmables et/ou non programmable (CPU, DSP, ASICIP,FPGA) des réseaux de communication complexes (Bus hiérarchiques sur puce, réseau sur puce) des composants de mémorisation variés, périphériques E/S, etc. …[2]

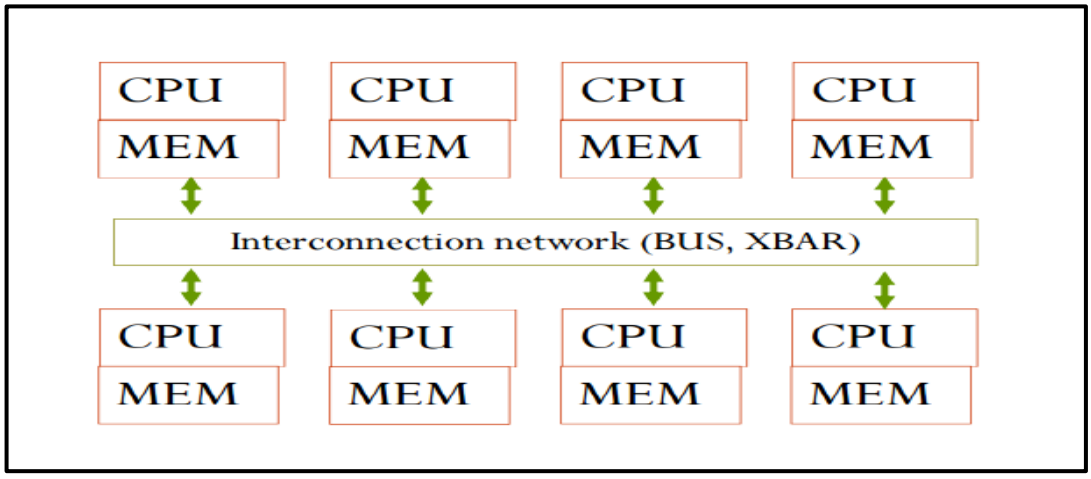


Figure I.12*Multi-processeur Soc*

I.6.2.la structure de MPSoC

La représentation d'un modèle générique d'un système MPSoC hétérogène avec des parties matérielles et logicielles structurées en couches pour maîtriser la complexité. Le matériel se divise en deux couches :**[2]**

A. La couche basse : contient les composants de calcul et de mémorisation utilisés par le système (µP, µC, DSP, matériel dédié IPs, mémoires) ;

B.La couche matérielle de communication : est embarquée sur la puce composée de deux

✓ sous-couches : média de communication (liens point-à-point, bus hiérarchique, réseau sur puce) et adaptateurs de communication entre le réseau et les composants de la première couche.

✓ Le logiciel embarqué est aussi découpé en couches :

C. La couche la plus basse : est l'abstraction du matériel (HAL, pour Hardware Abstraction Layer en anglais) permet de faire le lien avec le matériel en implémentant les pilotes E/S des périphériques, des contrôleurs de composants, les routines d'interruption (ISR, pour Interruptif Service Routine).

D. La couche système d'exploitation : qui permet de porter l'application sur l'architecture (gestion de ressources, communication et synchronisation, ordonnancement).

E. La couche application..

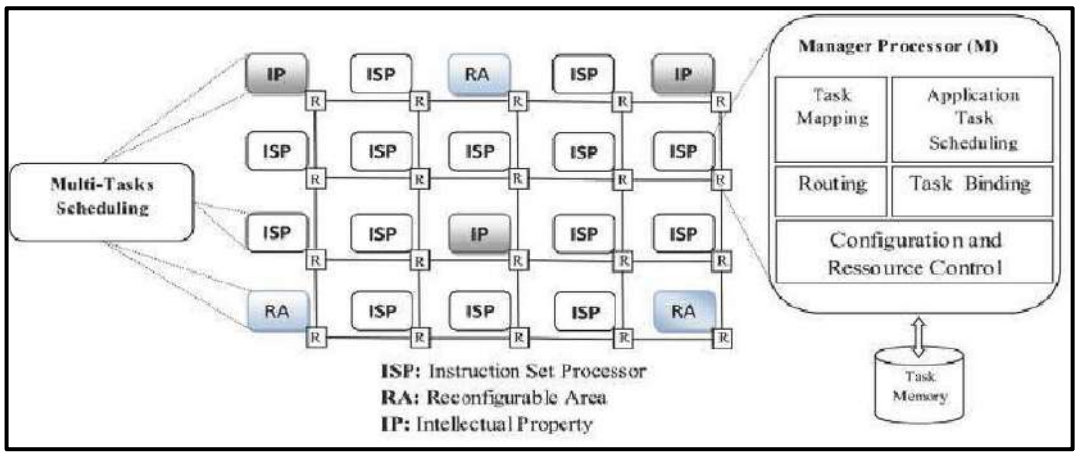


Figure I.13 *Architecture du MPSoC*

I.7.Les flux de conceptions FPGA_SoC :

Le silicium SoC avec un élément FPGA intégré ajoute la fonctionnalité de l'appareil final, mais peut ajouter de la complexité aux développeurs de logiciels de processeur. Des flux de

conception distincts pour le développement du processeur et du FPGA permettent aux développeurs de rester dans l'environnement de développement existant en utilisant des outils de développement appropriés et familiers.

Ce mémoire d'architecture décrit les flux d'outils de développement logiciel pour maximiser le développement. Pour fournir les flux de conception préférés pour les FPGA SoC aux développeurs de logiciels embarqués et aux concepteurs FPGA, quatre domaines clés, le débogage logiciel, le transfert de matériel / logiciel, la séquence d'amorçage et le support OS, doivent être résolus. Il s'ensuit que le développeur de logiciels peut utiliser les mêmes outils, Comme tout autre SoC. Toute fonctionnalité requise en raison de l'intégration du FPGA, est maintenue aussi distincte que possible. Cela permet des flux de conception où des équipes de développement séparées de processeurs et de FPGA continuent de se développer comme si les composants SoC-FPGA intégrés étaient des dispositifs distincts.

Toutefois, le développeur de logiciels de processeur a des avantages à avoir le FPGA intégré au processeur, en particulier dans le développement de pilotes et les outils de débogage. Les outils de débogage traitent la conception FPGA comme composante ASIC intégrée durcie. Cela supprime la complexité FPGA, permettant le développement et le débogage sans la nécessité d'apprendre le FPGA adaptatif**[8]**.

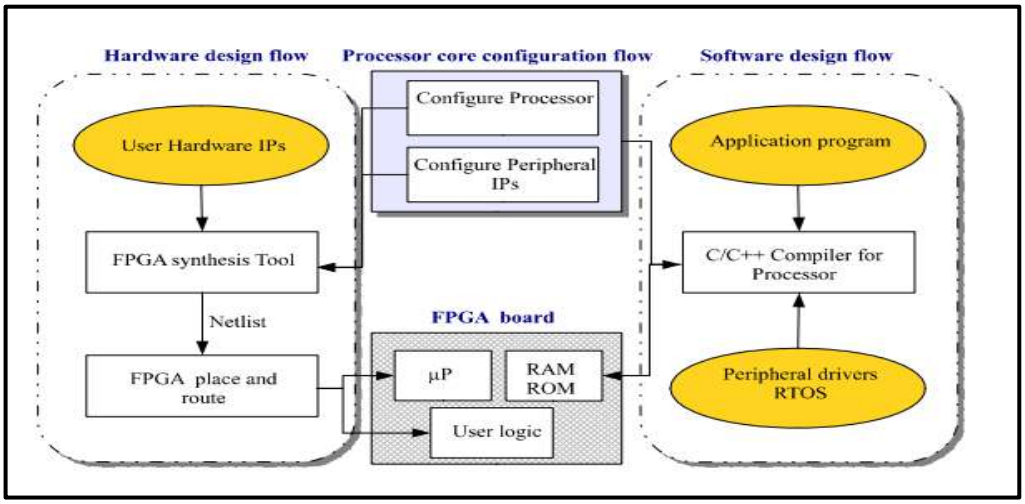


Figure I.14 *Flux de conception FPGA-SOC.*

I.8.Codesign : Conception conjointe logicielle/matérielle

Le terme "Hardware/Software Concurrent Design" souvent abrégé par "Hw/SwCodesign" et qui se traduit par conception conjointe matériel et logiciel, est défini comme un La méthodologie et la technique de conception des logiciels et du matériel en même temps.

Quatre grandes étapes sont distinguées dans l'architecture de codesign :

A. **Spécification**: qui concerne les étapes de spécification et de conception Fonctionnelle et pour lesquelles beaucoup de formalismes existent. Ces formalismes (modèles et langages) servent à décrire principalement l'aspect fonctionnel. Les spécifications non-fonctionnelles telles que les performances, la fiabilité, la sécurité, le coût, sont beaucoup plus délicates à formaliser et pourtant elles sont essentielles pour décider d'un partitionnement approprié et justifié.

B. Partitionnement HW/SW: le Partitionnement matériel/logiciel qui concerne l'étape de conception architecturale ; qui vise à identifier à partir de a solution fonctionnelle complète la partie purement logicielle, la partie purement matérielle, la partie concernée par le co-design.

C. Co-Synthèse: (étape de réalisation) qui se charge de transformer les descriptions fonctionnelles en descriptions directement implantables sur les processeurs matériels et logiciels de l'architecture cible. Considérés séparément, le développement des constituants logiciels (Proc, DSP) et matériels (ASIC, FPGA) sont aujourd'hui bien maîtrisés.Beaucoup de travaux de co-synthèse concernent aussi la génération des interfaces matériel/logiciel.

D. Co-Simulation: est une simulation de la description mixte matériel-logiciel résultant d'un partitionnement:"Hardware-software co_simulationis a means of verifying thefunctionality of mixed hardware-software descriptions" .[9]

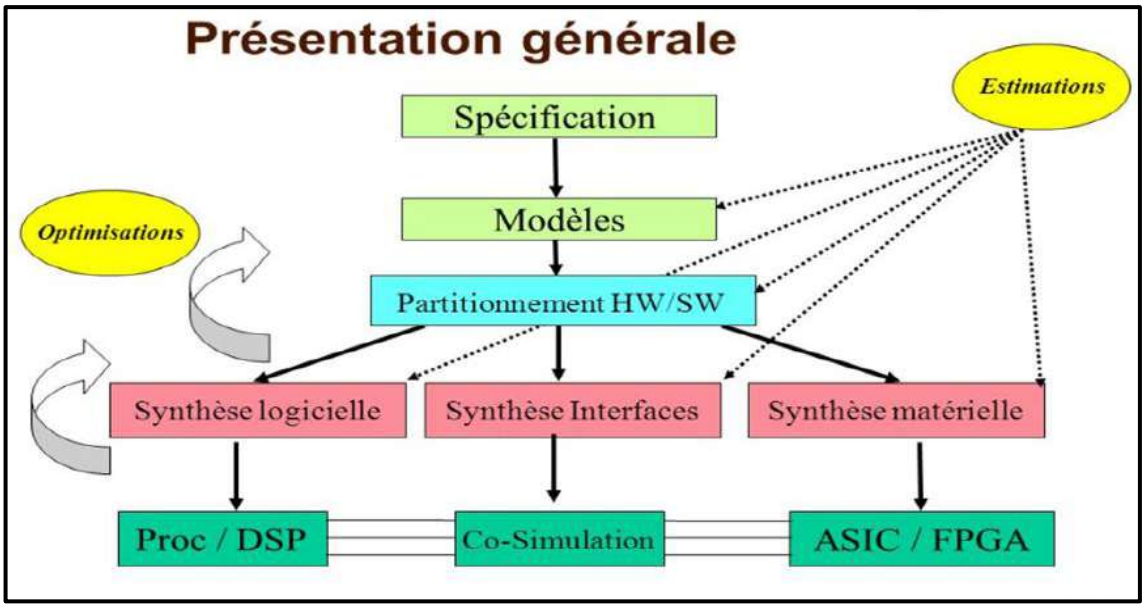


Figure I.15*L'architecture de codsigne*

I.9. Conclusion

Ce chapitre présente les systèmes sur pus(SOC), son domaine d'application à basé d'un circuit FPGA qui consiste les éléments de base de notre projet,le softcore (partie logiciel), le hardcore (partie matériel) et le codsigne qui conjoint les deux parties

Chapitre II:Généralité sur l'algorithme FFT

II.1.Introduction

Largement utilisée en Physique, la transformée de Fourier d'une fonction dépendant d'une variable (par exemple du temps) est devenue si naturelle que sa représentation graphique est généralement aussi utile, voire plus, que celle de la fonction elle-même. La Transformée de Fourier Rapide (notée par la suite FFT) est simplement une Transformée de Fourier discrète (TFD) calculée selon un algorithme permettant de réduire le nombre d'opérations et, en particulier, le nombre de multiplications à effectuer. Il faut noter cependant, que la réduction du nombre d'opérations arithmétiques àeffectuer. Tout dépend de l'architecture du processeur qui exécute le traitement. Pour calculer une TFD. Et c'est ce que on va le découvrir dans ce chapitre. **[10] [11]**

Dans ce chapitre, nous allons présenter la transformée de Fourier ou la FFT ainsi que les différents algorithmes conventionnelles radix-2, radix-4 qui illustrent la complexité de l'algorithme FFT. Enfin nous allons donner un aperçu sur les architectures les plus connues.

II.2. Présentation deLa transformée de Fourier

La transformée de Fourier est depuis longtemps un outil analytique principal dans des domaines aussi divers que les systèmes linéaires, l'optique, la théorie des probabilités, la physique quantique, les antennes et l'analyse du signal…ect. Une déclaration similaire n'est pas vraie pour la transformée de Fourier discrète. Même avec les vitesses informatiques énormes disponibles avec les ordinateurs modernes, la transformée discrète de Fourier a trouvé relativement peu d'applications en raison de la quantité exorbitante de temps de calcul requis. Cependant, avec le développement de la transformée de Fourier rapide (un algorithme qui calcule efficacement la transformée de Fourier discrète), de nombreuses facettes de l'analyse scientifique ont été complètement révolutionnées .Comme pour tout nouveau développement qui engendre des technologies qui subies des importants changements.

La transformée de Fourier est une opération qui transforme une fonction intégrale sur $\mathbb{R}$ en une autre fonction, décrivant le spectre fréquentiel de cette dernière. Ainsi si une fonction f est intégrable sur $\mathbb{R}$, sa transformée de Fourier est la fonction F donnée par la formule suivante:

$$\boldsymbol{F(v) = \int_{-\infty}^{+\infty} f(t)\, e^{-i2\pi vt}\, dt} \qquad \textbf{(II.1)}$$

Et son inverse :

$$F(v) = \int_{-\infty}^{+\infty} f(v)\, e^{+i2\pi vt}\, dv \quad \textbf{(II.2)}$$

II.2.1.La transformée de Fourierdiscrète

L'implémentation des algorithmes de traitement de signal en informatique ne se fait malheureusement pas sur l'espace infini des réels $\mathbb{R}$, ce qui nécessiterait un temps de calcul infini. En revanche, la théorie du traitement de signal numérique permet d'adapter la transformée de Fourier sur un nombre fini d'échantillons donnant lieu à l'algorithme de transformée de Fourier discrète (DFT).**[11]**

Soit un signal numérique de N échantillons, la définition mathématique de sa transformée de Fourier discrète est:

$$X(k) = \sum_{n=0}^{N-1} x(\mathrm{n}) e^{-2i\pi\frac{nk}{N}} \textbf{(II.3)}$$

Et son inverse :

$$x(n) = \frac{1}{N}\sum_{K=0}^{N-1} X(\mathrm{k}) e^{+2i\pi\frac{nk}{N}} \textbf{(II.4)}$$

II.2.2.La transformée de Fourier rapide

Les algorithmes de transforméede Fourier rapide (FFT, Fast Fourier Transform) permettent de faire une DFT de manière efficace et de réduire la complexité des calculs à l'ordre $\mathbf{CN}Log_2\mathbf{N}$, où **C**est un coefficient inférieur ou égal à un et qui dépend de l'algorithme choisi. En effet, il existe de nombreux algorithmes de FFT, l'algorithme le plus connu est celui de J. W. Cooley et de J. W. Tukey de par leur célèbre publication datant de 1965. Il est construit autour d'une propriété de la transformée de Fourier qui démontre que le calcul d'une transformée de Fourier sur N échantillons, sachant que N = r.n, peut être factorisé par le calcul de r transformées de Fourier sur n échantillons. Il existe, à ce jour, un grand nombre d'algorithmes FFT, nécessitant chacun d'entre eux le respect d'hypothèses permettant d'exploiter les propriétés correspondantes et d'obtenir le maximum des performances de ces algorithmes FFT. Ainsi un algorithme peut être efficace dans une situation et inutile dans

d'autres situations. Il est donc indispensable de bien choisir son algorithme avant de se lancer dans l'implémentation et l'optimisation de ce dernier.**0**

II.3.Complexité arithmétique de l'algorithme TFR

Le nombre de multiplications/additions réelles nécessaires pour réaliser une TFR à N points dépend de la capacité de l'algorithme à générer des coefficients de rotation triviaux au lieu de valeurs complexes non triviales. Par coefficients triviaux on entend des facteurs ayant la valeur ±1 ou ±j, qui correspondent aux multiples de $W_N^{\frac{N}{4}} = \boldsymbol{e}^{-2i\pi\frac{nk}{N}}$. Dans ce cas, il s'agit simplement d'une permutation des valeurs réelles et imaginaires. Un gain supplémentaire en terme de multiplications triviales est obtenu en considérant les multiplications par les facteurs de rotation multiple impaire de $W_N^{\frac{N}{8}} = \frac{\sqrt{2}}{2} - \mathrm{j}\,\frac{\sqrt{2}}{2}$.

Dans ce cas, la partie réelle et imaginaire du facteur de rotation sont les mêmes, nous permettant de diviser par deux le nombre de multiplications réelles nécessaires. La complexité arithmétique est la même peu importe l'entrelacement utilisé, en temps ou en fréquence. En effet, dans les deux cas nous avons logr N étages chacun ayant N/r modules papillons. Dans cette étude, nous considérons les multiplieurs complexes classiques constitués de 4 multiplications réelles et 2 additions réelles (qu'on désignera par 4M2A) et des multiplieurs basse consommation nécessitant seulement 3 multiplieurs réels mais au prix d'un additionneur réel supplémentaire (qu'on désignera par 3M3A). **Les tableaux*II.1***et ***II.2*** illustrent les expressions mathématiques de la complexité arithmétique pour les différents algorithmes présentés et le type de multiplieur complexe utilisé. **Les tableaux*II.3*** et ***II.4*** donnent le nombre de multiplications/additions non triviales pour différentes valeurs de N.**[10]**

Tableau II.1*Complexité arithmétique des algorithmes pour un multiplieur 4M2A*

Algorithmes	***Le nombre de mult réelles non triviales***	***Le nombred'add Réelles non triviales***
Radix-2	$2Nlog_2\,\mathrm{N} - 7\mathrm{N} + 12$	$3Nlog_2\,\mathrm{N} - 3\mathrm{N} + 4$
Radix -4 et 2^2	$\frac{3N}{2}log_2\mathrm{N} - 5\mathrm{N} + 8$	$\frac{11N}{4}log_2\mathrm{N} - \frac{26N}{12} + \frac{8}{3}$
Radix- 8 et2^3	$\frac{4N}{3}log_2\mathrm{N} - \frac{59N}{14} + \frac{40}{7}$	$\frac{33N}{12}log_2\mathrm{N} - \frac{57N}{28} + \frac{16}{7}$

TableauII.2.*Complexité arithmétique des algorithmes pour un multiplieur 3M3A.*

Algorithmes	***de mult. réelles non triviales***	***d'add. Réelles non triviales***
Radix-2	$\frac{3N}{2}log_2\mathrm{N} - 5\mathrm{N} + 8$	$\frac{7N}{2}log_2\mathrm{N} - 5\mathrm{N} + 8$
Radix -4 et 2^2	$\frac{9N}{8}log_2\mathrm{N} - \frac{43N}{8} + \frac{16}{3}$	$\frac{25N}{8}log_2\mathrm{N} - \frac{43N}{12} + \frac{16}{3}$
Radix- 8 et2^3	$\frac{25N}{24}log_2\mathrm{N} - \frac{350N}{112} + 4$	$\frac{146N}{48}log_2\mathrm{N} - \frac{25N}{8} + 4$

Tableau II.3.Nombre de multiplications/additions non triviales pour un multiplieur 4M2A

	Radix-2		***Radix -4 et 2^2***		***Radix- 8 et2^3***	
N	***Mult***	***Add***	***Mult***	***Add***	***Mult***	***Add***
4	0	16	0	16		
8	4	52			4	52
16	28	148	24	144		
32	108	388				
64	332	964	264	920	248	928
128	908	2308				
156	2316	5380	1800	5080		
512	5644	12292				
1024	13324	27652	10248	25944		
2048	30732	61444				
4096	69644	135172	53256	126296	48280	126832
8192	155660	294916				
16384	344076	638980	262152	595288		

Tableau II.4.*Nombre de multiplications/additions non triviales pour un multiplieur 3M3A*

	Radix-2		***Radix -4 et 2^2***		***Radix- 8 et2^3***	
N	***Mult***	***Add***	***Mult***	***Add***	***Mult***	***Add***
4	0	16	0	16		
8	4	52			4	52
16	28	148	24	144		
32	108	388				
64	332	964	264	920	248	928
128	908	2308				
256	2316	5380	1800	5080		
512	5644	12292				
1024	13324	27652	10248	25944		
2048	30732	61444				

Toutefois, l'utilisation d'algorithmes de plus hauts degrés augmente la complexité d'intégration dans un circuit dédié. Même si le nombre d'additions et de multiplications réelles non triviales offre un excellent indice sur l'efficacité d'un algorithme, l'intégration matérielle tient compte d'autres critères de performances, qui sont la régularité de l'algorithme ainsi que la complexité de réalisation et de contrôle de l'architecture

II.4.Les algorithmes conventionnels de FFT

Il existe deux algorithmesconventionnels les plus connus radix-2 et radix-4.

II.4.1.Radix-2

L'algorithme FFT radix-2 ou radical-2 en français est un algorithme procèdent un structure de papillon. L'algorithme FFT radix-2 ce divise en deux algorithmes la DIT (Decimation In Time, Décimation dans le temps) et DIF (Decimation In Frequency, Décimation en fréquence).

Le Radix-2 utilise la technique « diviser pour régner », il divise la FFT en sous-système de N/2 points, puis calcule chaque sous-système tout seul pour obtenir à la fin les composantes du spectre fréquentiel du signal. **0**

Prenons un signal discret x[n] de longueur N (N est pair). Définissons alors deux nouveaux signaux x_1[n] et x_2[n] chacun de longueur N/2 et qui sont constitués par les échantillons pairs et impairs de x[n] respectivement:

x_1[n] = x[2n] n=0 ,1,…….,(N/2) -1

x_2[n] = x[2n+1]

II.4.1.1. L'algorithme FFT radix-2 a Décimation temporelle

L'algorithme FFT radix-2, est construit autour de deux propriétés de ce coefficient:
La première propriété met en évidence le lien entre les coefficients invoqués dans le calcul de la transformée de Fourier pour un signal de N échantillons avec ceux de la même transformée d'un signal de N/2 échantillons.

$$\boldsymbol{w_N^{2nk} = w_{\frac{N}{2}}^{nk}} \textbf{ (II.5)}$$

La deuxième propriété montre une certaine périodicité des coefficients de par leur nature trigonométrique

$$\boldsymbol{w_N^{n(k+\frac{N}{2})} = (-1)^n w_N^{nk}} \textbf{ (II.6)}$$

Ainsi, dans le cas où N est un multiple de deux, si l'on réécrit la transformée de Fourier discrète en deux sommes indépendantes tout en distinguant les échantillons d'indice pair de ceux d'indice impair, nous obtenons:

$$\mathbf{X(k)} = \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x(2n)} w_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x(2n+1)} w_N^{(2n+1)k} \textbf{ (II.7)}$$

$$\mathbf{X(k)} = \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x(2n)} w_N^{2nk} + w_N^k \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x(2n+1)} w_N^{2nk} \textbf{(II.8)}$$

En réinjectant la première propriété dans l'équation précédente, celle-ci devient:

$$\mathbf{X(k)} = \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x}(2n) w_{\frac{N}{2}}^{nk} + w_N^k \sum_{n=0}^{\frac{N}{2}-1} \mathbf{x}(2n+1) w_{\frac{N}{2}}^{nk} \textbf{(II.9)}$$

Ainsi pour k∈ [0, N/2-1], le calcul d'une transformée de Fourier de N échantillons revient à calculer deux transformées de Fourier sur N/2 échantillons:

$$\boldsymbol{X(k) = DFT x_{2n}(k) + w_N^k\, DFT\,(x_{2n+1})(k)} \qquad \textbf{(II.10)}$$

Concernant la deuxième moitié des calculs, c'est à dire pour k∈ [N/2, N-1], ceci revient à écrire l'équation toujours pour k∈ [0, N/2-1] de la façon suivante:

$$\boldsymbol{X\left(k+\frac{N}{2}\right) = DFT(x_{2n})\left(k+\frac{N}{2}\right) + w_N^{k+\frac{N}{2}}\, DFT\,(x_{2n+1})(k+\frac{N}{2})} \textbf{(II.11)}$$

Le spectre d'un signal de N échantillons étant N-périodique et en appliquant la deuxième propriété énoncée concernant les coefficients, l'équation devient:

$$\boldsymbol{X\left(k+\frac{N}{2}\right) = DFT(x_{2n})(k) - w_N^k\, DFT\,(x_{2n+1})(k)} \qquad \textbf{(II.12)}$$

En récapitulant, dans le cas où le nombre N d'échantillons est multiple de deux, calculer une transformée de Fourier d'un tel signal revient à calculer deux transformées de Fourier sur deux signaux de N/2 échantillons.

$$\boldsymbol{X(k) = DFT(x_{2n})(k) + w_N^k\, DFT\,(x_{2n+1})(k)} \qquad \textbf{(II.13)}$$

$$\boldsymbol{X\left(k+\frac{N}{2}\right) = DFT(x_{2n})(k) - w_N^k\, DFT\,(x_{2n+1})(k)} \textbf{(II.14)}$$

De cette écriture, est tiré le diagramme du célèbre opérateur papillon de l'algorithme FFT radix-2.

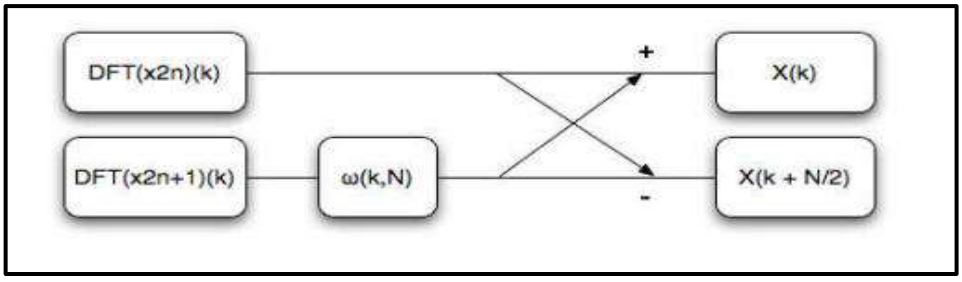


Figure II.1.*Diagramme de l'opération Papillon radix-2*

L'intérêt de cette opération est de réduire le nombre d'opérations à effectuer pour le calcul de la transformée de Fourier. Le tableau suivant montre les gains apportés en fonction du nombre d'échantillons constituant le signal.

Tableau II.5.*Gain apporté par l'écriture de la DFT sous forme radix-2*

	DFT		Radix-2	
	Multiplica-tions	Additions	Multiplica-tions	Additions
N	N^2	N(N-1)	$(N^2 + N)/2$	$N^2/2$
16	256	240	136	128

Cette propriété étant récursive, il en est de même pour les deux DFT de taille N/2. Ainsi, dans le cas où le nombre d'échantillons constituant le signal est égal à une puissance de deux,$N = 2^m$, cette propriété est applicable m fois jusqu'à obtenir des DFT sur 2 échantillons qui se traduisent par une addition complexe et une soustraction complexe.

m étages de N/2 papillons sont alors à calculer, ce qui donne une complexité de calcul de:

- $m.N/2 = \log_2(N).N/2$ multiplications complexes
- $m.N = \log_2(N).N$ additions complexes

Tableau II.6.*Dénombrement des opérations entre une DFT et une FFT radix-2*

	DFT		FFT Radix-2	
	Multiplications	Additions	Multiplications	Additions
N	N^2	N(N-1)	$\log_2(N).N/2$	$\log_2(N).N$
16	256	240	32	64

La multiplication complexe représente quatre multiplications flottantes et deux additions flottantes. L'addition complexe, quant à elle, représente deux additions flottantes. Ainsi cet algorithme diminue le nombre d'opérations flottantes à $5N.\log_2(N)$, soit une complexité de calcul en $O(N.\log_2(N))$. En revanche, cet algorithme entraîne un ordonnancement des données en amont du calcul FFT, d'où son nom précisant l'entrelacement temporel des données. Le déploiement des opérations pour N=16 permet de remarquer qu'en entrée d'algorithme,

l'ordre des données n'est pas l'ordre naturel du signal mais ce dernier est obtenu par bit reversing de la position de chaque échantillon.

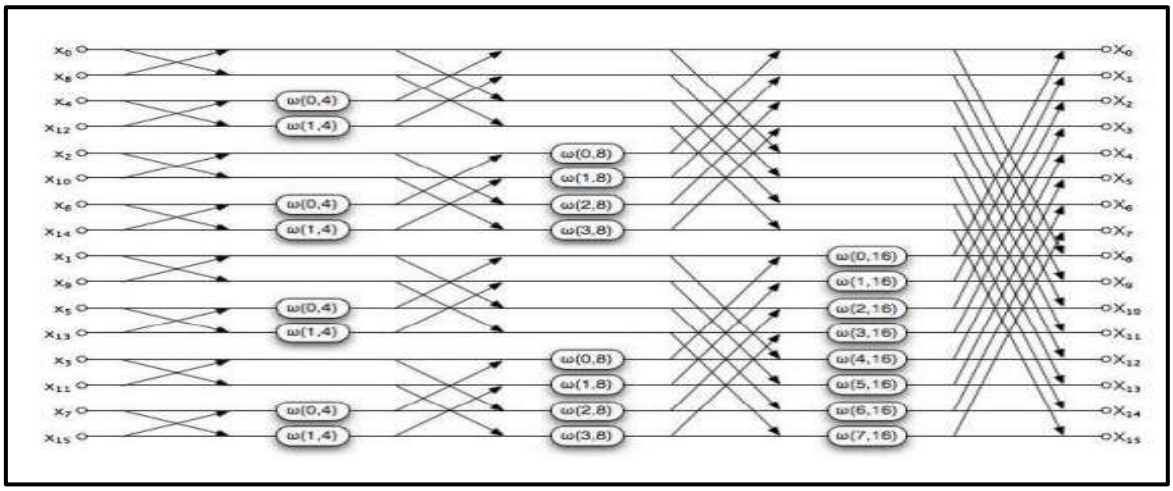


Figure II.2.*Diagramme papillon déployé pour une FFT N=16*

II.4.1.2 l'algorithme FFT radix-2 a décimation fréquentiel

Contrairement à son homologue à entrelacement temporel, l'algorithme FFTradix-2 à entrelacement fréquentiel se caractérise par l'ordre résultant de la transforméesur le signal d'entrée. En effet, l'ordre obtenu après transformée est analogue àl'ordre nécessaire en entrée de l'algorithme à entrelacement temporel, faisant appelau bit-reversing. En revanche, l'ordre nécessaire en entrée de cet algorithme correspondà l'ordre naturel des échantillons.

L'opérateur papillon de cet algorithme devient:

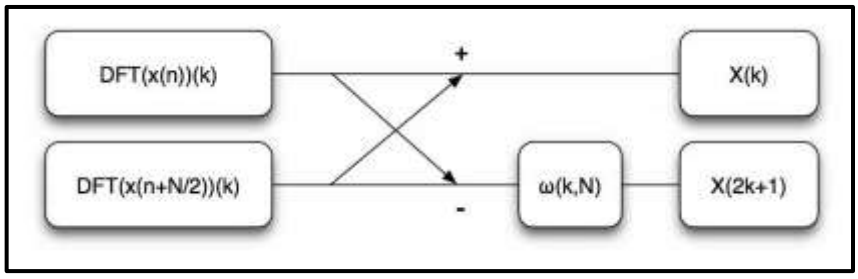


Figure II.3.*Diagramme papillon radix-2 à entrelacement fréquentiel*

Le nombre d'opérations par papillon est identique à l'algorithme radix-2 entrelacement temporel, 3 additions, 3 soustractions et 4 multiplications. L'algorithme se déroule aussi de la même façon avec $\log_2(N)$ étages de N/2 papillons où N est une puissance de 2.

Les deux algorithmes se distinguent ainsi par la localité de l'ordonnancement et de la nature du papillon. **0**

II.4.2.Radix-4

L'algorithme FFT radix-4 est une décomposition de l'algorithme de transformée de Fourier discrète en 4 transformées indépendantes de taille N/4. Ainsi le papillon caractéristique de cet algorithme se présente sous la forme:

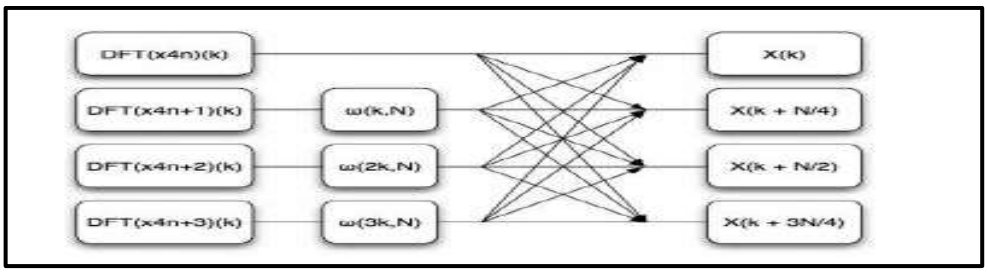


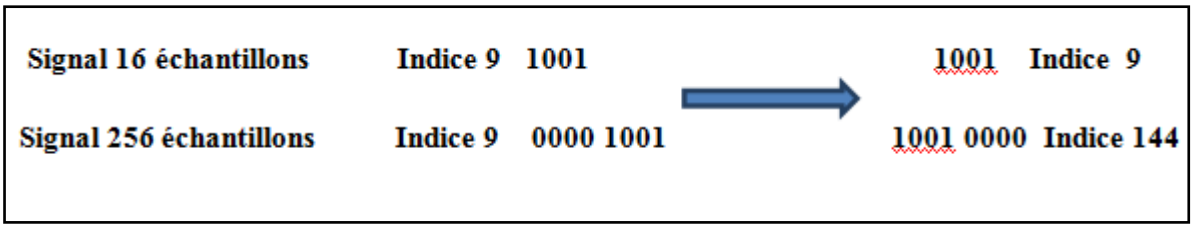
Figure II.4.*Diagramme papillon radix-4*

En comparaison avec l'algorithme radix-2, l'algorithme radix-4 est une succession de $\log_4(N)$ étages de N/4 papillons où N est une puissance de 4 représentants le nombre d'échantillons constituant le signal à considérer.**0**

II.5.Le bit-renverse

Une grande partie des algorithmes de transformée de Fourier rapide présente une caractéristique commune, celle d'un ordonnancement des données. Considérons l'ordonnancement de l'algorithme FFT radix-2 à entrelacement temporel.

Cet ordre est obtenu par application de l'opération bit-reverse. Chaque échantillon dispose d'un indice permettant de les différencier au sein du signal. Ainsi l'ordre naturel des échantillons va de l'indice 0 à N-1 où N représente la taille du signal. L'opération bit-reverse consiste à inverser l'écriture binaire de ces indices. Le nombre de bits nécessaire à l'écriture binaire de l'indice dépend directement de la taille du signal. Ainsi un signal de 16 échantillons nécessitera 4 bits pour exprimer ses indices en écriture binaire tandis qu'un signal de 256 échantillons nécessitera quant à lui un total de 8 bits. Ceci a un impact direct sur l'opération bit-reverse. L'échantillon d'indice 9 du premier signal ne sera pas permuté alors le même échantillon du deuxième signal sera permuté avec l'indice 144.**0**



FigureII.5.*Exemple d'opération bit-renverse*

II.6.Architectures conventionnelles de la FFT

Il existe deux types d'architecture pour la réalisation de la FFT, les architectures en pipeline et les architectures à base de mémoires.

II.6.1. L'architecture en pipeline

L'architecture en pipeline est caractérisée par un traitement continu des entrées séquentielles de la FFT. Une telle architecture est composée de logr N, modules de calcul (MC), un par étage, qui correspond aux opérateurs papillon (multiplexage spatiale).

La valeur de r correspond au radix de l'algorithme utilisé. Chaque MC traite N/r opérations papillon successives (multiplexage temporel). Par conséquent, la taille maximale réalisable pour ce type d'architecture est dictée par le nombre de MC.

Les architectures en pipeline ont une mémoire des échantillons distribuée sur tous les étages. En effet, à chacun des logr N, étages, il y a seulement N/2k échantillons, k étant le numéro de l'étage $1 \leq k \leq \log_r N$. Il en est de même pour les mémoires des coefficients W, dont le nombre est diffèrent d'un étage à un autre.

Les différentes architectures en pipeline diffèrent principalement selon deux points. La première différence est le nombre de chemins de données utilisés pour traiter les échantillons. On trouve ainsi deux types d'architectures, à chemin unique (single-path) ou à chemin multiple(Multi-path). Un plus grand nombre de chemins permet d'augmenter le débit de traitement mais au prix d'une augmentation des ressources nécessaires. La deuxième différence consistent en la stratégie de mémorisation pour créer les différents délais nécessaires à l'ordonnancement des échantillons.**0**

II.6.1.1Multi-path Delay Commutator

L'approche Multi-path Delay Commutator (MDC) est une des premières approches utilisées pour des architectures de la TFR en pipeline ; elle est encore largement utilisée .Elle comprend un commutateur pour permuter les échantillons et des délais pour synchroniser les mêmes échantillons. Selon l'algorithme radix-r utilisé, on aura r − 1 délais avant et après chaque module papillon. De plus, cette architecture nécessite r − 1 multiplieurs par module papillon. La figure(**II.6 a)** illustre un exemple d'une architecture radix-2 MDC de 8 points. Les modules papillon, les multiplieurs et les délais ont un taux d'utilisation de 50% du temps. La figure**(II.6 b)** illustre le cas pour le radix-4. Cette architecture a un taux d'utilisation des multiplieurs de 25%. En général, on lui préfère la version SDC qui permet d'augmenter ce taux et de diminuer le nombre de multiplieurs.**[16]**

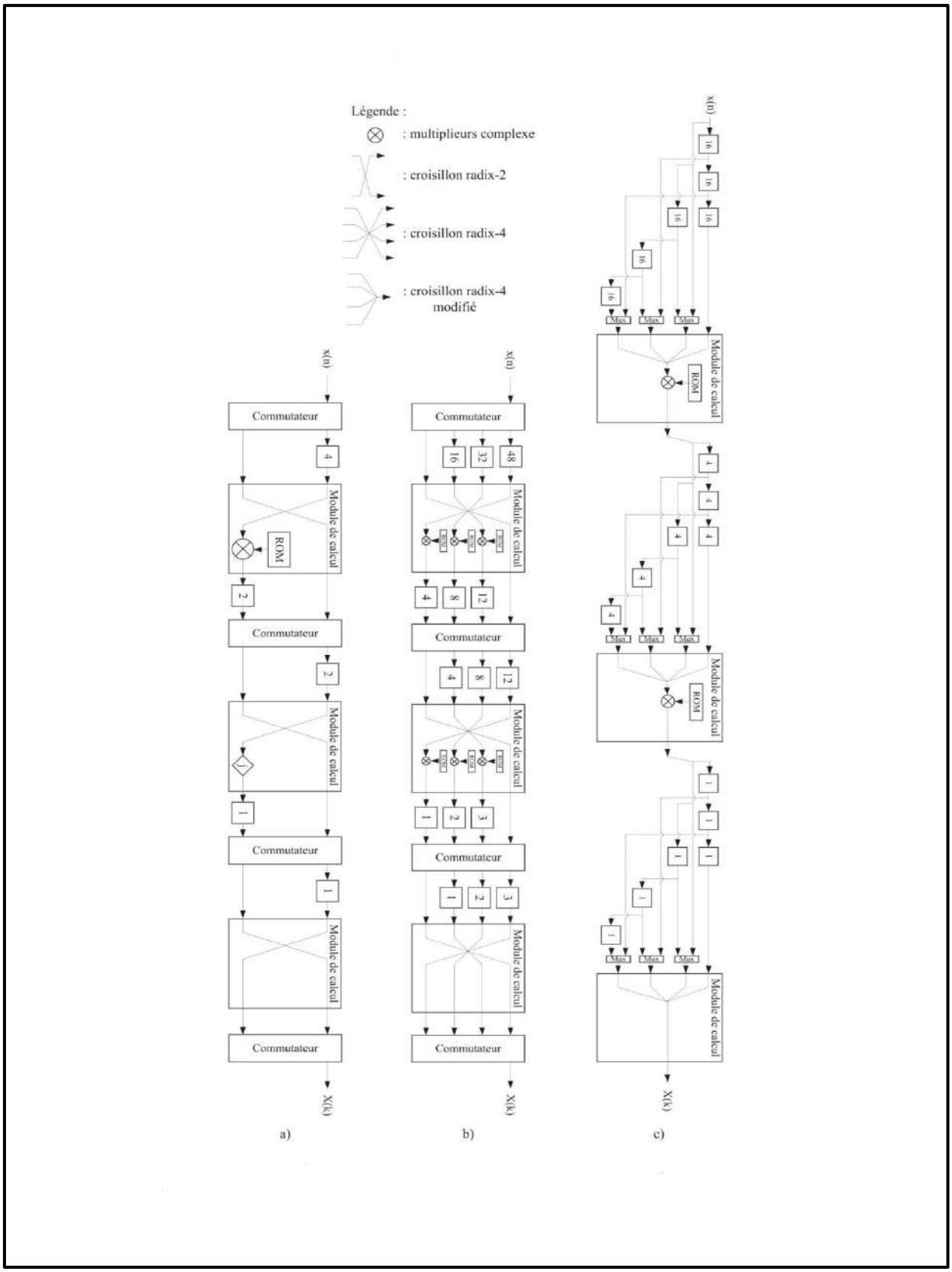


Figure II.6.*Architectures de TFR en pipeline* ***a)*** *à 8 points radix-2 MDC* ***b)*** *à 64 points radix-4 MDC* ***c)*** *à 64 points radix-4 SDC*

II.6.1.2.Single-path Delay Commutator

La version Single-path Delay Commutator (SDC) pour le radix-4 permet d'augmenter le taux d'utilisation du multiplieur à 75% en utilisant 1 multiplieur complexe au lieu de 3 multiplieurs complexes par étage . Cette modification divise le papillon radix-4 en un papillon simplifié radix-4, traitant seulement une sortie du papillon radix-4 d'origine à la fois. Le papillon conventionnel radix-4 utilise 8 additionneurs/soustracteurs tandis que la version simplifiée utilise 3 additionneurs/soustracteurs. Cependant, cette modification engendre un contrôle plus complexe. La figure illustre l'architecture d'une TFR radix-4 SDC de 64 points.

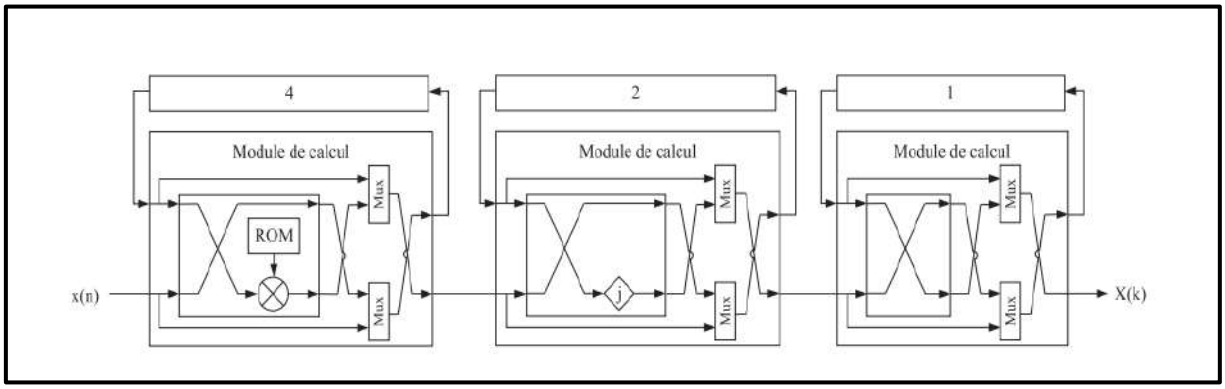


Figure II.7.*FFT à 8 points radix-2 SDF*

II.6.1.3Single-path Delay Feedback

Alors que les registres à délai sont utilisés à 50% du temps dans l'approche MDC, l'architecture Single-path Delay Feedback (SDF) permet d'optimiser le nombre de registres en utilisant deux fois moins de registres avec un taux d'utilisation de 100%. Contrairement à l'approche MDC, au lieu de stocker les échantillons entrant et sortant du module papillon dans deux ensembles différents de r − 1 registres (pour une algorithme radix-r), l'approche SDF utilise un seul ensemble de r − 1 délais. La figure illustre l'architecture d'une TFR radix-2 à 8 points utilisant l'approche SDF. An d'exploiter le parallélisme et ainsi répondre à des débits plus importants, plusieurs chemins de données sont utilisés. On obtient ainsi des architectures en pipeline-parallèles. La figure***II.8*** illustre une architecture en pipeline SDF avec un parallélisme des chemins de données de P = 4 proposée notamment pour l'UWB.**[17]**

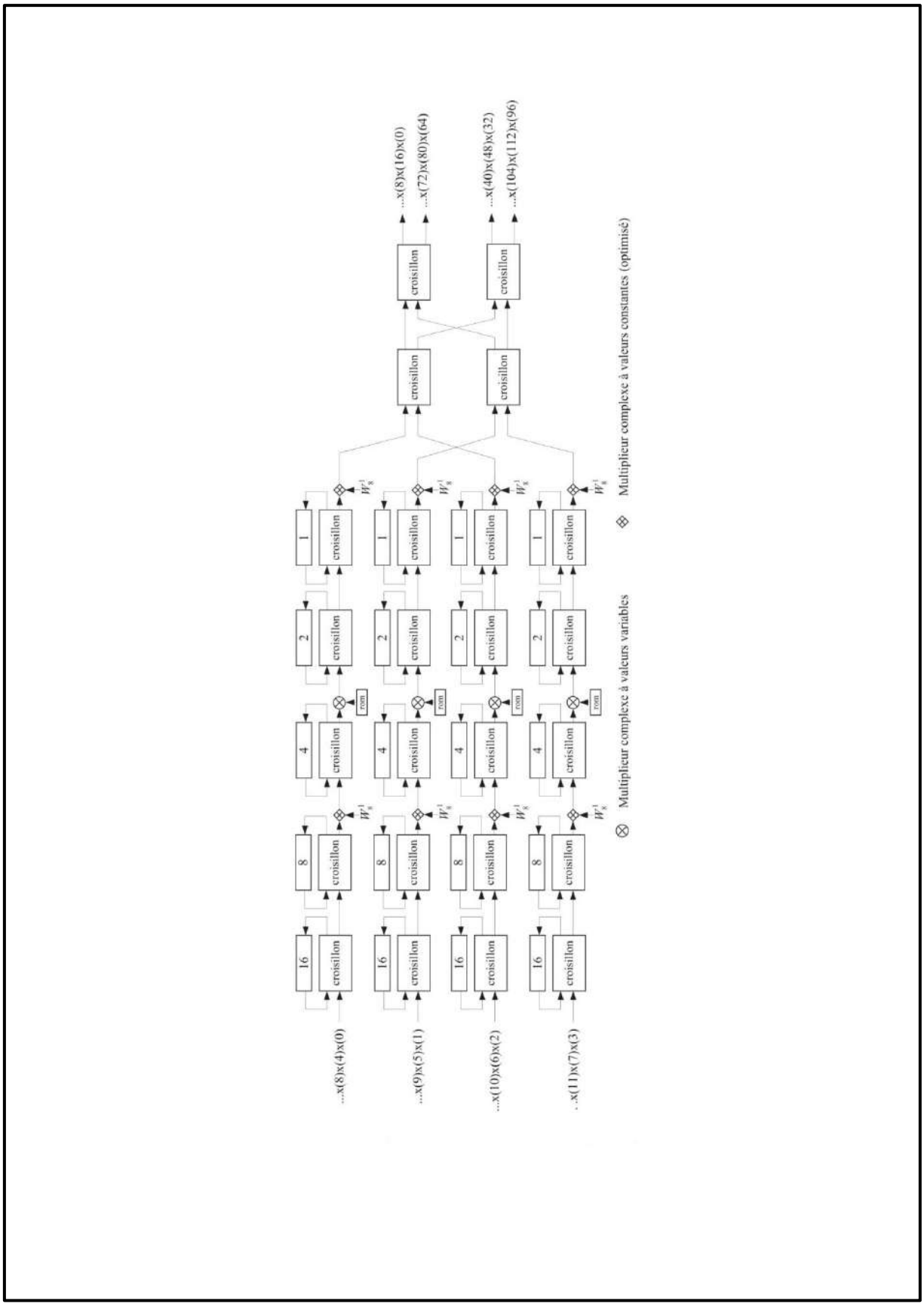


Figure II.8. *Architecture en pipeline-parallèle SDF mixed-radix-$2^3/2^4$de 128 points pour l'UWB*

➢ Comparaison

L'approche radix-r MDC est r fois plus rapide que l'approche radix-r SDF. Toutefois en termes de mémoires, l'approche SDF est plus économe que l'approche MDC (qui souffre d'un surplus de surface par rapport à l'approche SDF). En termes de ressources, les architectures radix-2^4et radix-4^2 SDF sont plus performantes que n'importe quelle autre approche comme on le voit au tableau. Elles nécessitent toujours le moins de ressources arithmétiques et de mémoires. Ceci est dû entre autres au fait qu'elles utilisent plusieurs multiplieurs par des valeurs constantes optimisées et caractérisées par une complexité inférieure. En termes de complexité de contrôle, les architectures pour le radix-4-SDC et split-radix sont les plus complexes à contrôler.**0**

Tableau II.7. *Ressources matérielles selon l'architecture en pipeline utilisée*

Architecture	***#Multiplieurs complexes***	***#additionneurs complexes***	***Mémoire***	***Contrôle***	$f^{(1)}op$
R2 MDC	$2\log_4 N-1$	$4\log_4 N$	3N/2-2	simple	$f^{(2)}$*échant*/ r
R2 SDF	$2\log_4 N-1$	$4\log_4 N$	N-1	simple	*f échant*
R4 MDC	$3(\log_4 N-1)$	$8\log_4 N$	5N/2-2	simple	*f échant / r*
R4 SDF	$\log_4 N-1$	$8\log_4 N$	N-1	moyen	*f échant*
R4 SDC	$\log_4 N-1$	$3\log_4 N$	2N-2	complexe	*f échant*
$R2^2$ SDF	$\log_4 N-1$	$4\log_4 N$	N-1	simple	*f échant*
$R2^3$ SDF	$\approx\log_4 N-1$	$4\log_4 N$	N-1	simple	*f échant*
$R2^4$ SDF	$\leq 0.7\log_4 N-1$	$4\log_4 N$	N-1	simple	*f échant*
$R4^2$ SDF	$\leq 0.7\log_4 N-1$	$4\log_4 N$	N-1	simple	*f échant*

SR MDC	$4(\log_4 N-1)$	$12\log_4 N-8$	$3N/2-2$	complexe	f *échant* / *r*
SR SDF	$\text{Log}_4 N-1$	$4\log_4 N$	$N-1$	complexe	f *échant*

(1)Fréquence d'opération. (2) Fréquence d'échantillonnage.

II.6.2.Architectures à mémoires

Les architectures à base de mémoires mémorisent en général les N échantillons à l'entrée avant d'entamer le traitement de la FFT. Elles sont composées d'un ou de plusieurs modules de calculs (papillons), d'unités de mémorisation et d'une unité de contrôle. Contrairement à l'approche en pipeline, la plus grande taille de FFT ne dépend que de la taille mémoire disponible et non du nombre de modules de calcul .Du fait du multiplexage temporel du traitement de la FFT qui caractérise ce type de solution, la surface totale du circuit est inférieure à celle des architectures de type pipelines qui sont caractérisées par un multiplexage spatial et qui nécessitent donc plus de ressources. La figure illustre l'architecture à base de mémoires.

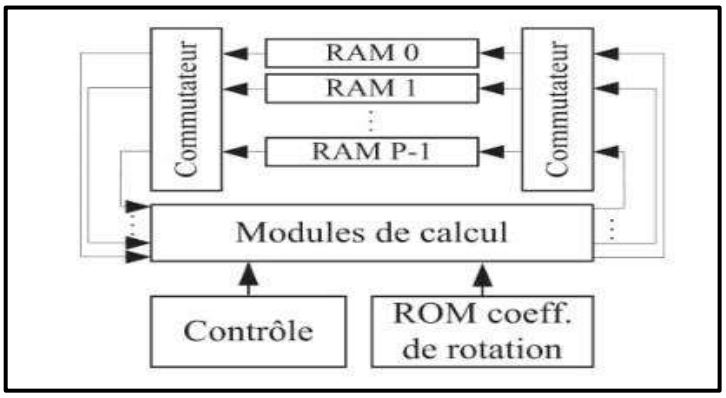


Figure II.9.*Architecture à base de mémoires*

Il est possible d'obtenir plusieurs degrés de parallélisme et de profondeur de pipeline selon le type de configuration des modules de calculs. La figure illustre quatre configurations différentes des modules de calcul. La première approche décrite à la figure **(II.10.a)** est la plus simple. Elle consiste en un module de calcul réalisant l'opérateur papillon radix-2. La fréquence d'opération de ce type de configuration est plus grande que dans le cas des

architectures en pipelines. L'augmentation du degré de parallélisme réduit la fréquence d'opération. Ceci peut être obtenu soit en utilisant plusieurs modules papillon en parallèle soit par la réalisation d'un module papillon d'un radix élevé, ce qui permet de traiter plusieurs données à la fois. Les figures **(II.10.b)** et **(II.10.c)** illustrent ces cas de figures avec la configuration de deux modules radix-2 en parallèle et d'un module radix- 2^2 respectivement. Dans les deux cas, nous avons le même degré de parallélisme de 4. Néanmoins, le casradix-2^2 à la figure **(II.10.c)** exploite un plus grand multiplexage spatial et réalise deux étages de suite. La figure **(II.10.d)** illustre une autre approche pour réaliser l'algorithme radix- 2^2dans une architecture à mémoires. Dans ce cas, les deux modules de calcul sont en pipeline (MDC) et aucun parallélisme intra-étage des modules n'est utilisé. La fréquence d'opération de cette architecture est moins élevée que celle du cas simple à la figure (**II.10.a)** mais reste un peu plus élevé que celle de l'approche **(II.10.c)** Toutefois, cette approche possède un bon compromis entre les solutions **(II.10.a)** et **(II.10.c).** En effet, la figure **(II.10.d)** nécessite moins de ressources matérielles que l'approche **(II.10.c)** tout en offrant de bonnes performances de puissance de calcul

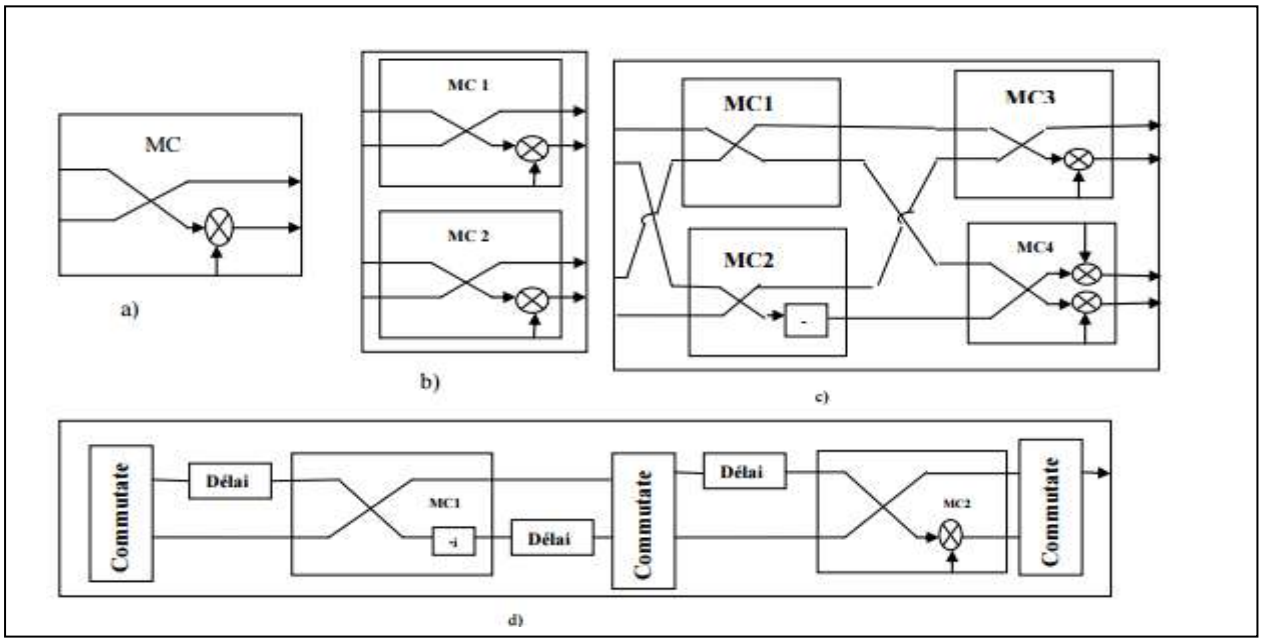


Figure II.10.*Différentes configurations des modules de calculs des architectures à mémoires*a) un papillon radix-2,**b)** deux papillons radix-2 en parallèle, **c)** un papillon radix-2^2**, d)** un papillonradix- 2^2 en pipeline MDC.

Un des avantages supplémentaires de l'utilisation d'un radix plus élevé dans les architectures à base de mémoires est la diminution des opérations de lecture/écriture dans les

mémoires des échantillons. En effet, ces opérations s'effectuent tous les logr N étages. Les configurations des types **(II.10.c)** et **(II. 10.d)** auront donc besoin de moins d'accès mémoires comparativement aux configurations **(II.10.a)** et **(II.10.b),** ce qui permet de limiter grandement la puissance consommée. Le tableau **(II.8)** donne le nombre de modules de calcul nécessaires et la fréquence d'opération selon la configuration choisie (pour des algorithmes radix-2 i). PD Correspond au parallélisme des données, c'est-à-dire le nombre d'échantillons à l'entrée des modules de calcul à chaque coup d'horloge. P correspond au nombre de modules papillon radix- 2 i en parallèle. PC Correspond au nombre de chemins de données pour la configuration 15d). PC Vaut 1 pour un module papillon SDF et 2 pour l'approche MDC (en considérant les algorithmes radix- 2 i). Tcalcul Correspond au nombre de cycles de traitement des modules de calcul.

Tableau II.8.*Ressources matérielles et fréquences d'opération des architectures à base de mémoire*

Configuration	MC	Parallélisme des données P_D	Fréquence d'opération
A, B et C	$P \times i \times \left(\frac{2^i}{2}\right)$	$P_D = P \times 2^4$	$\frac{\log_r N\left[\left(\frac{N}{P_D}-1\right)+T_{calcul}\right]}{N} f_{échant}$
D	$P \times \log_2 r$	$P_D = P \times P_C$	

Configurations A : un papillon radix-2,

B : deux papillons radix-2 en parallèle,

C : un papillon radix-2^2 ,

D : un papillon radix- 2^2en pipeline MDC

II.7.Domaines d'utilisation de la FFT

L'algorithme FFT est utilisée dans divers domaines,elle convient mieux à l'analyse des enregistrements audio numériques qu'au filtrage ou à la synthèse sonore. Parmi ses applications :

✓ Elle permet d'obtenir l'équivalent logiciel d'un analyseur de spectre, que les ingénieurs utilisent pour tracer le graphe des fréquences contenues dans un signal électrique

✓ Application sonore : utilisé pour déterminer la fréquence d'une note dans une musique enregistrée, pour essayer d'identifier des oiseaux ou des insectes

✓ le traitement d'image (avec une version bi-dimensionnelle de la FFT)

✓ applications statistiques : essayer de détecter des fluctuations périodiques dans les prix du marché, les populations animales… etc

✓ applications scientifiques : l'analyse des informations sismographiques, qui permettent de prendre des "sonogrammes" de l'intérieur de la Terre. Même l'analyse des séquences d'ADN

II.8.Conclusion

En résumé, Les techniques de transformation rapide de Fourier (FFT) ont révolutionné les techniques de traitement du signal numérique au cours des 30 dernières années, vu les multiples utilisations possibles de cet outils de traitement du signal. il fournit non seulement une réponse rapide, mais aussi des pensées logique impossible à réaliser mais qui sont réalisable grâce a la FFT. On retrouve plusieurs variantes des algorithmes de FFT radix-2 et radix-4. En général, celles-ci se font par rapport à l'ordre dans lequel les opérations papillon sont exécutées.

Des algorithmes utilisant des bases différentes de deux ou de quatre ou utilisant une combinaison de différentes bases existent aussi. Les algorithmes de FFT peuvent aussi bien être programmés en logiciel qu'implémenté en matériel. Ce chapitre porte sur la conception matérielle d'un module de FFT configurable. Les architectures intéressantes à un tel design sont principalement celles utilisant les algorithmes de types radix-2 ou radix-4.

L'avantage de l'utilisation d'un Radix plus grand, réside dans la diminution du nombre deMultiplications ainsi que la diminution du nombre d'étages pour exécuter la FFT et diminue la

Charge d'interconnexion et d'accès mémoires. Nous avons vu aussi un aperçu sur les différents architectures FFT pipelines et parallèles parmi ces architectures on a choisi la plus adéquate pour notre besoin SDF (Single-path Delay Feedback), MDC (Multi-path Delay Commutator).

L'architecture SDF est considérée comme la meilleure approche afin de calculer la FFT pourles systèmes à faible débit. Puisque l'architecture SDF possède un seul chemin d'accès.

L'architecture MDC est l'architecture pipeline la plus populaire et la plus utilisée pour L'implémentation des systèmes de communications qui demandent un haut débit binaire.

Chapitre III: Conception et Implémentation

III.1 Introduction

Ce dernier chapitre traite la partie de l'implémentation de FFT sur FPGA, pour cela, nous allons détaillés les blocs principaux du fonctionnement de FFT dont nous servirons de passerelle. Tout d'abord nous allons commencer par un schéma explicatif des blocs utiliser pour le fonctionnement de notre circuit.

On a expliqué par la suite chaque blocs utilisé en détaille, puis nous somme passé a l'étape suivante celle de pratique sur le logiciel de description de matériel **vivado**d'où on a illustré ces blocs en les reliant et simulant.

On a représenté notre travail sur un organigramme pour clarifier les étapes de l'exécution de travail.

Enfin on a terminé par la réalisation de ce circuit sur matériels, on a utilisé la carte FPGA qui est la base de notre circuit,la carte AnalogDiscovery 2 qui est le générateur des signaux et enfinle convertisseur analogique numérique pmodAD1 avec ces matériaux on a fait les calculs et les résultats des testes obtenues seront exposés et détaillés.

III.2 les blocs principaux de fonctionnement de la FFT

Nous avons proposéces blocs principaux pour le fonctionnement de FFT la figure suivante montre le :

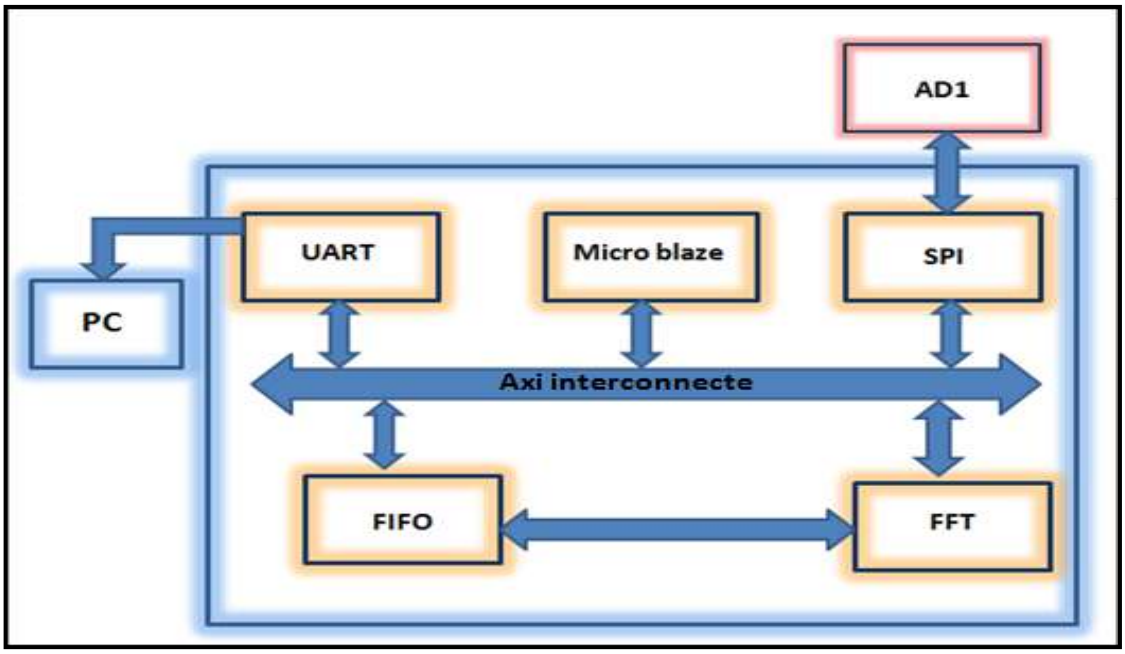


Figure III.1 *schéma des blocs principaux de fonctionnement de FFT*

III.2.1 Le processeur de contrôle (Microblaze)

Le système de microcontrôleur Microblaze ™ (MCS) LogiCORE ™ est un système complet de processeur autonome destiné aux applications de contrôleur. Il est très intégré et comprend le processeur Microblaze, la mémoire locale pour le programme et le stockage de données, ainsi qu'un module IO étroitement couplé mettant en œuvre un ensemble standard de périphériques.**[18]**

Le processeur MicroBlaze inclus dans le MCS a une configuration fixe, optimisée pour Zone minimale. Le processeur MicroBlaze complet est disponible dans ISE® Design Suite Embedded Edition. **[19]**

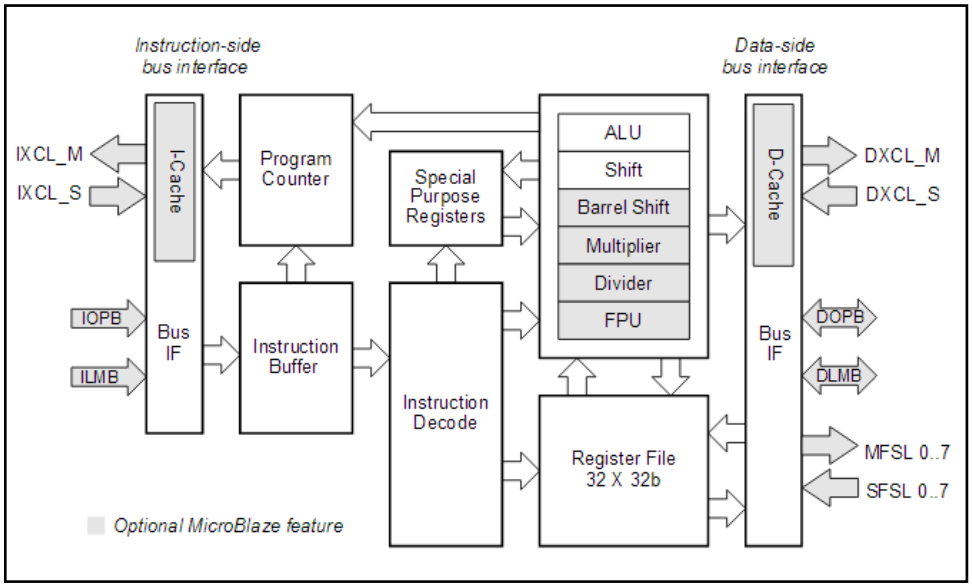


Figure III.2 Architecture *interne du MicroBlaze***[13]**

III.2.2 Fast Fourier Transform(FFT)

Le noyau Fast Fourier Transform (FFT) est utilisé pendant la calcul de la transformation FFT 1024 points. Le noyau FFT Xilinx offre un certain nombre d'architectures différentes et prend également en charge plusieurs calculs arithmétiques. L'architecture de Core FFT est illustrée à la **figure III.3[20][21]**

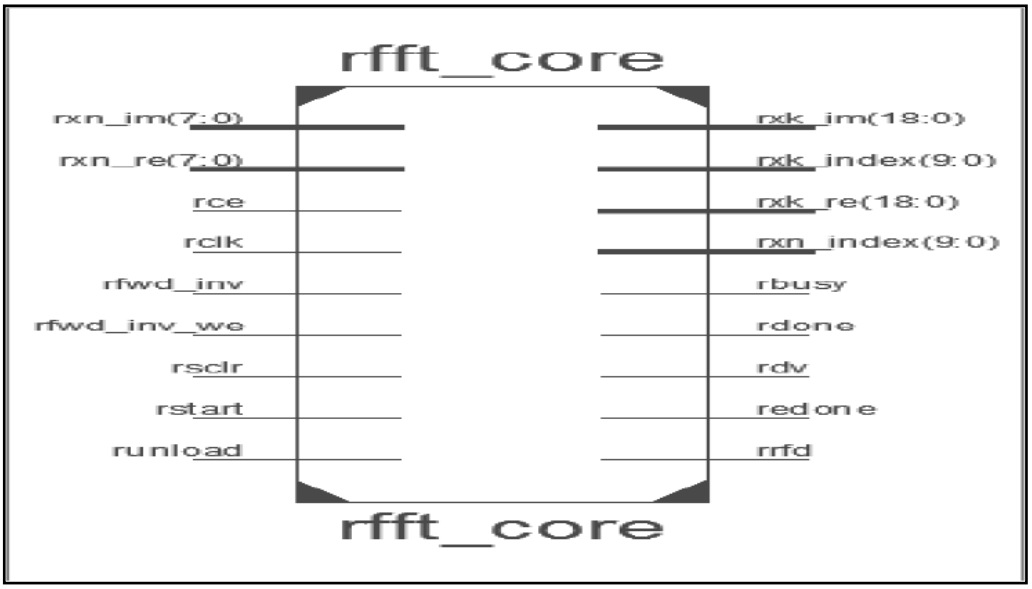


Figure III.3 *Architecture de core FFT*

III.2.3 AXI Streaming FIFO

Le noyau FIFO permet l'accès mappé en mémoire à un Interface AXI4-Stream. Le noyau peut être utilisé pour l'interface avec les IP AXI Streaming, similaire au noyau Ethernet LogiCORE IP AXisans avoir à utiliser une solution DMA complète. **[22]**

L'opération principale de ce noyau permet d'écrire ou de lire des paquets de données vers ou à partir d'un périphérique sans aucune préoccupation par rapport à la signalisation de l'interface AXI4-Stream.

Figure III.4 : Montre les principaux composants du noyau FIFO AXI-Stream: un bloc d'interface AXI avec une interface slave AXI4 / AXI4-Lite, un contrôleur d'interruption, un module Registres, un module de contrôle de réception, un module de contrôle de transmission, un FIFO de réception pour les données de réception Et la longueur, et un FIFO de transmission pour les données d'émission et la longueur.**[23]**

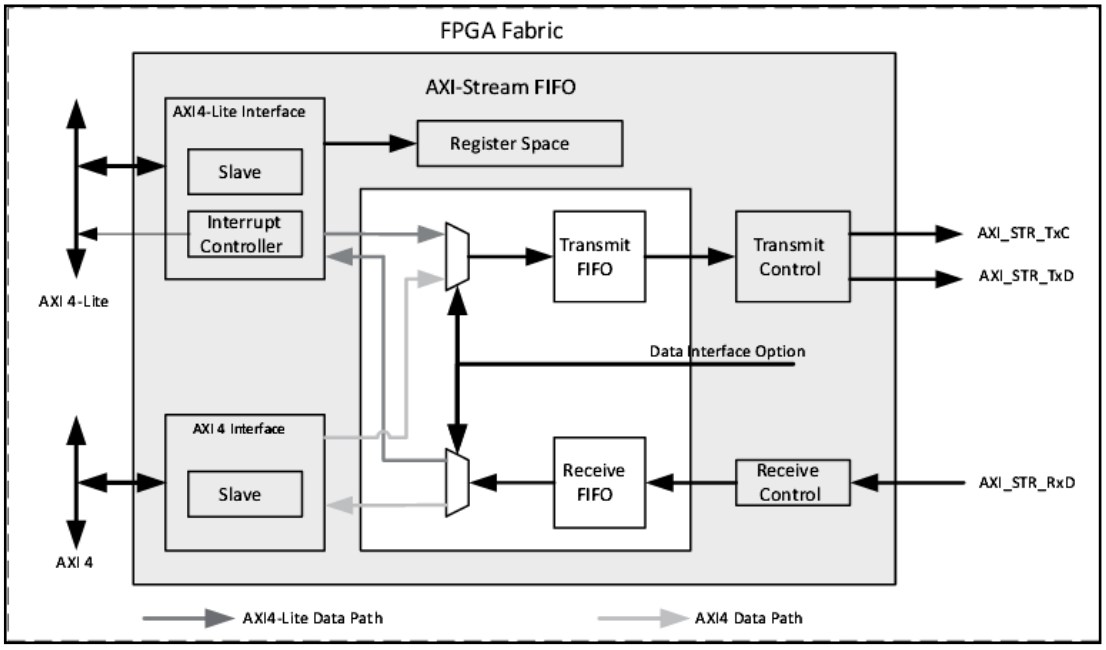


Figure III.4 *Diagramme AXI4 Stream FIFO*

III.2.4 SPI

Une liaison **SPI** (pour ***Serial Peripheral Interface***) est un bus de donnéessérie synchrone baptisé ainsi par Motorola, qui opère en mode full-duplex. Les circuits communiquent selon un schéma maître-esclaves, où le maître contrôle la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée Slave Select (SS) .La liaison série synchrone (SPI) permettant d'envoyer et de recevoir des données sous forme série à l'aide de 4 lignes :

- ✓ Une ligne d'entrée de données série soit MISO (Master IN Slave OUT)
- ✓ Une ligne de sortie de données série autrement MOSI (Master OUT Slave IN)
- ✓ Une ligne d'horloge soit SCK (Serial Clock)
- ✓ Une ligne de validation des transferts autrement dit SS.

Une des particularités de cette interface est de pouvoir fonctionner dans 2 modes : le mode maître et le mode esclave. **[17]**

La Capacités de SPI :

•Toujours full-duplex

-Communique dans les deux sens simultanément

-Les données transmises (ou reçues) peuvent ne pas être significatives

• Vitesse de transmission à plusieurs Mbps

-Vitesse d'horloge 0-50 MHz pas rare

• Transfert de données en caractères de 4 à 16 bits

• Supporte les esclaves multiples

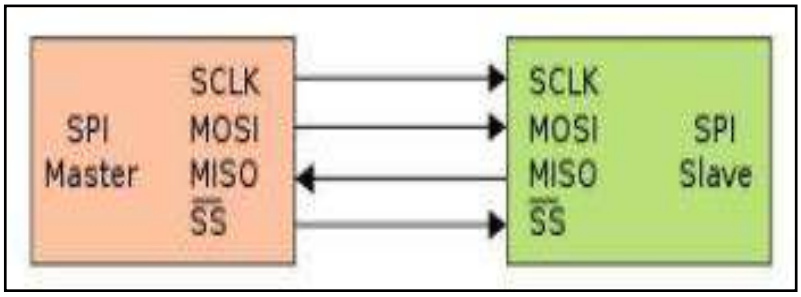


Figure III.5 *Liaison SPI Maitre-Esclave*

III.2.5 AXI Interconnect

L'interface d'interconnexion AXI connecte un ou plusieurs périphériques **master** par mémoire AXI à un ou plusieurs périphériques **slave** en mémoire**[26][27]**

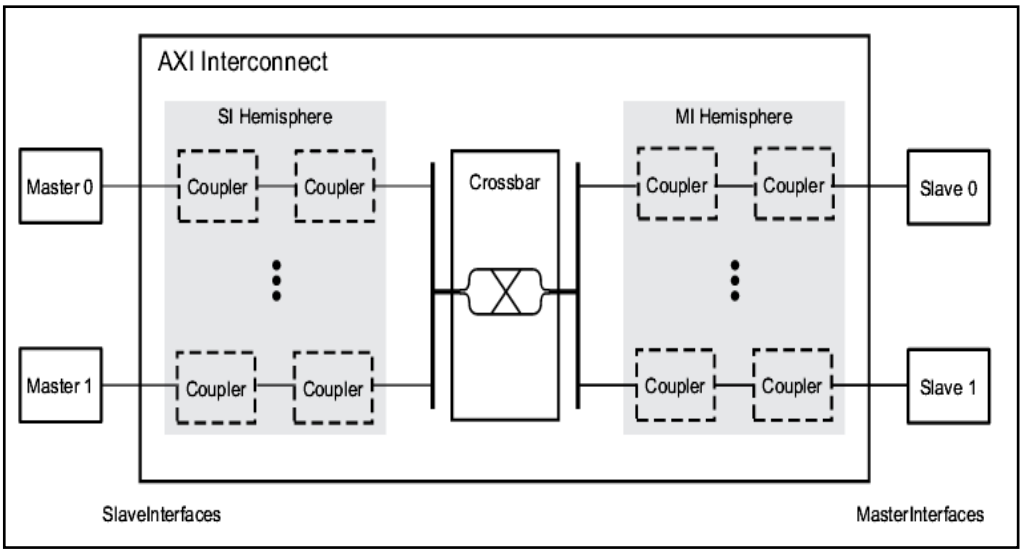


Figure III.6 *Diagramme de Core AXI Interconnect*

III.2.6 AXI UART Lite

AXI **universel asynchrone Transmetteur récepteur** (UART) Lite core fournit Entre les signaux UART et l'Advanced Microcontrôleur Bus Architecture (AMBA ®) AXI Interface et fournit également un contrôleur Interface pour le transfert de données en série asynchrone. Ce noyau Soft LogiCORE ™ IP est conçu pour Interface avec le protocole AXI4-Lite.**[28]**

UART est un moyen simple de connecter un FPGA à un PC. Nous avons juste besoin d'un module émetteur et récepteur**.**

Les modules AXI UART Lite sont affichés en **Figure III.7** et décrite dans les sections qui suivent :

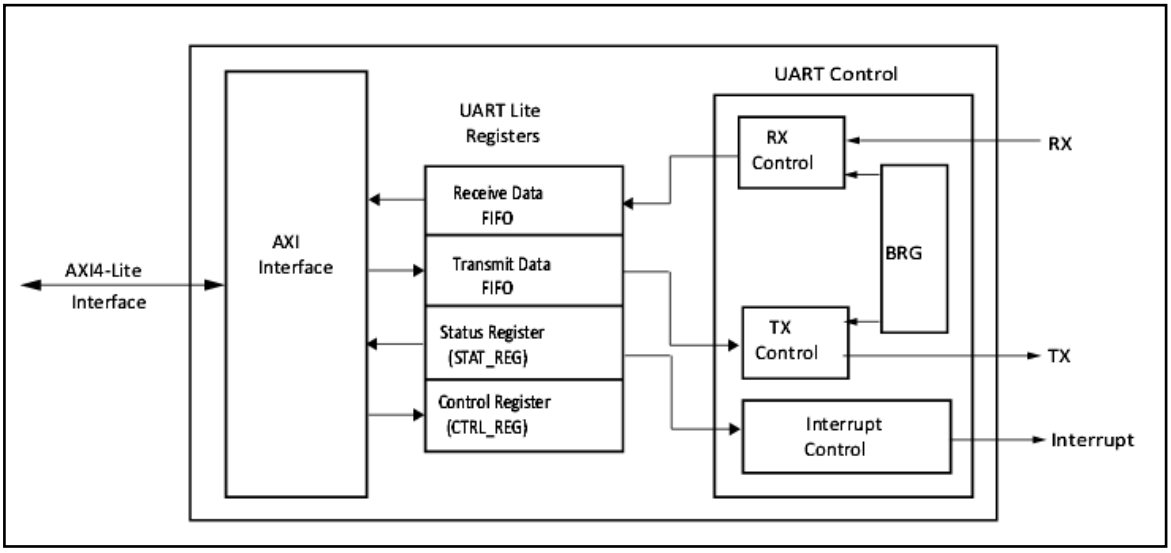


Figure III.7 *Diagramme de CoreAXI UART Lite*

III.2.7 PmodAD1

Le Digilent PmodAD1 est un convertisseur analogique-numérique à deux canaux de 12 bits doté d'AnalogDevices AD7476A Avec un taux d'échantillonnage d'un maximum de 1 million d'échantillons par seconde, ce Pmod est capable d'exceller dans les applications audio même les plus exigeantes.

Caractéristiques :

- ✓ Convertisseur analogique-numérique à deux canaux de 12 bits.
- ✓ Conversion A / D simultanée jusqu'à une MSa par canal.
- ✓ Deux filtres anti-alias à 2 pôles Sallen-Key.
- ✓ Petite taille de PCB pour les conceptions flexibles 0.95 in × 0.8 in (2.4 cm × 2.0 cm).
- ✓ Port Pmod à 6 broches avec interface GPIO.
- ✓ Library et code d'exemple disponibles dans le centre de ressources

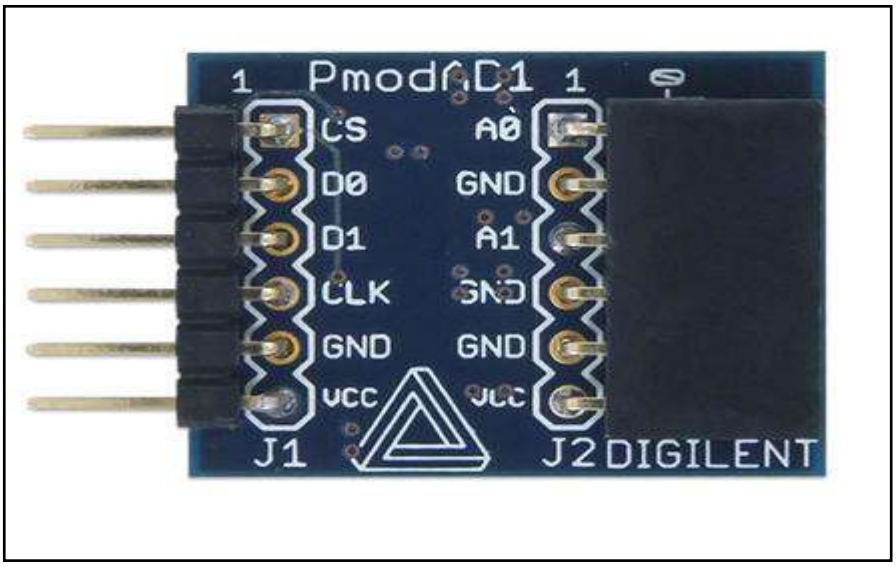


Figure III.8 *Le PmodAD1*

Le PmodAD1 communique avec la carte hôte via un protocole de communication de type anSPI. La différence entre le protocole SPI standard et ce protocole se manifeste dans l'agencement de broches sur ce Pmod. Une interface SPI typique s'attendrait à un Chip Select, un Master-Out-Slave-In, un Master-in-Slave-Out et un signal d'horloge série. Cependant, avec les deux ADC sur cette puce, les deux lignes de données (MOSI et MISO) sont conçues pour fonctionner uniquement comme des sorties, ce qui les rend à la fois Master-In-Slave-Sortie des lignes de données. **[29] [30]**

III.2.8 la Réalisation de la FFT sur vivado

Apres avoir détaille les blocs principaux a la réalisation du fonctionnement FFT, sur le logiciel vivado on a réalise un circuit a partir de ces blocs et cela on les relie les un au autre.Les résultats obtenus se présentent sur la figure suivante :

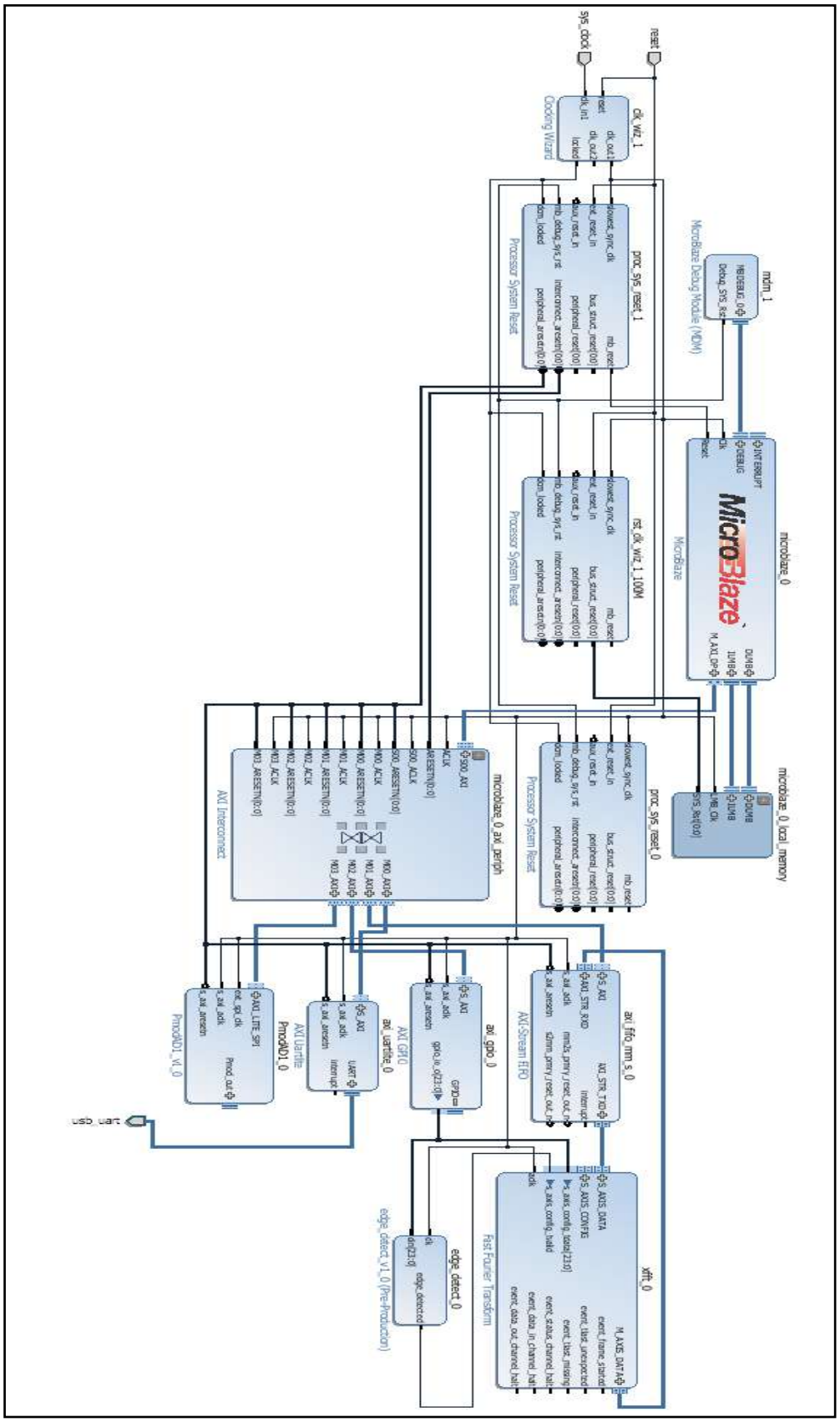


Figure III.9 *LaRéalisation de la FFT sur vivado*

III.3 Les résultats de la consommation des architectures internes de Bloc FFT

Il existe plusieurs architecturespour réaliserl'algorithme FFT, comme on a expliqué dans le chapitre II, donc nous présentons les résultats de l'implémentation des trois architectures qui sont Radix-2, Radix-4 et Pipelined.

Le rapport de consommation des ressources de FFT_Radix-2 est donné dans le tableau Suivant:

Tableau III.1.Consommation des ressourcesdeFFT_Radix-2.

Resource	Utilisation	Avalable	Utilisation %
LUT	3280	20800	15.77
FLIP FLOP(FF)	3429	41600	8.24
BRAM	43.50	50	87.00
DSP	5	90	5.56
IO	4	106	3.77

Le rapport de consommation des ressources de FFT_Radix-4 est donné dans le tableau suivant:

Tableau III.2.Consommation des ressourcesdeFFT_Radix-4.

Resource	Utilisation	Avalable	Utilisation
LUT	3730	20800	17.93
FLIP FLOP(FF)	4701	41600	11.30
BRAM	45.50	50	91.00
DSP	17	90	18.89
IO	4	106	3.77

Le rapport de consommation des ressources de FFT_Pipelined est donné dans le tableau suivant:

Tableau III.3.Consommation des ressourcesdeFFT_Pipelined.

Resource	Utilisation	Avalable	Utilisation
LUT	5134	20800	24.68
FLIP FLOP(FF)	7451	41600	17.91
BRAM	44	50	88.00
DSP	36	90	40.00
IO	4	106	3.77

D'après se qu'on a remarque dans les tableaux précédent que ya une petite différence dans la consommation de la RAM et on a remarque que la consommation de Radix_4 et en première position précède piplined et Radix_2.

III.4 Partie pratique : génère le signal

Dans cette partie nous allons utiliser la carte AnalogDiscovery 2. Cette carte contient waveforms qui est un générateur de fonction.

III.4.1 Waveforms

Les générateurs Waveforms on l'utilisé pour tester les circuits en générant des signaux définis qui sont ensuite introduits dans le circuit en analyse. On peut avoir un générateur waveforms arbitraire (AWG) qui produit n'importe quelle forme d'onde définie par l'utilisateur, ou avoir un générateur de fonctions qui émet des signaux prédéfinis comme sine, carré, etc.… **[31]**

III.4.1.1.Présentation du logiciel waveforms

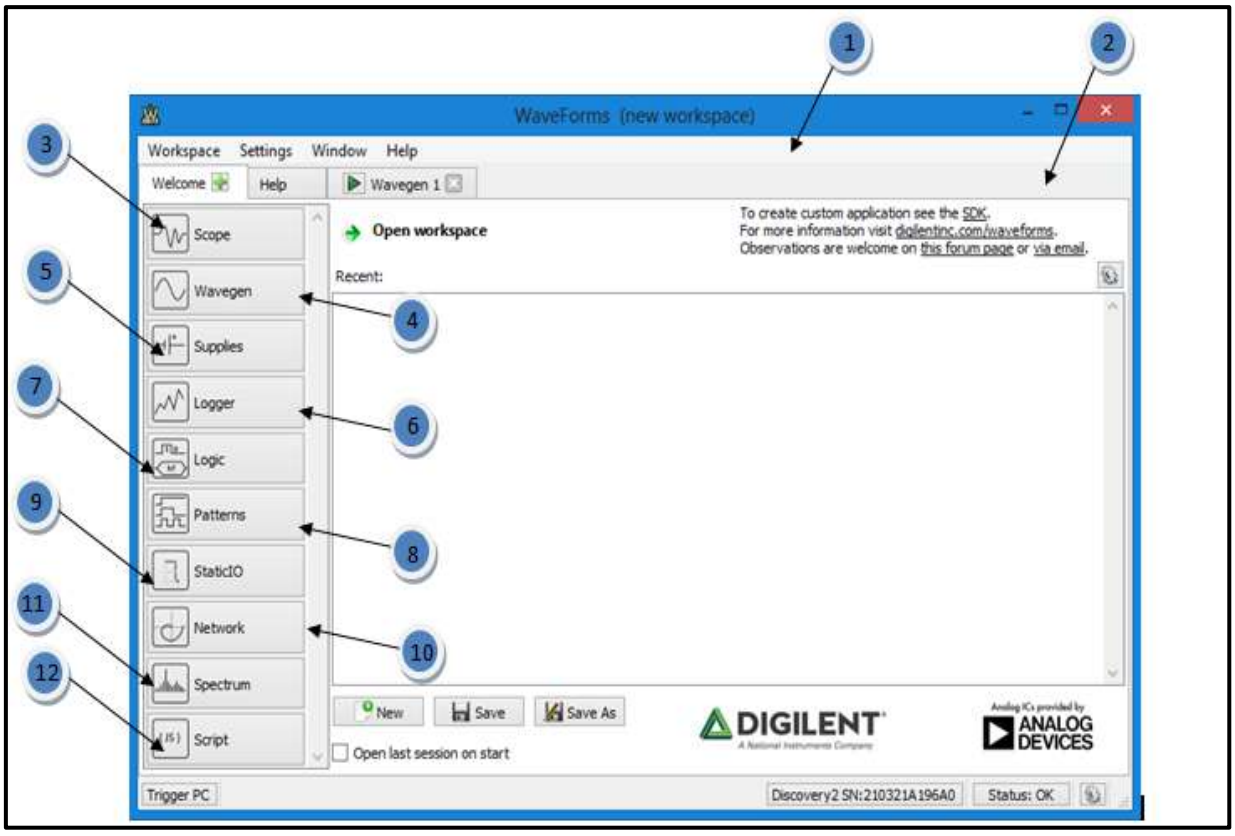


Figure III.10 *Interface waveforms*

Tableau III.4.*Présentation des éléments de la figureInterface waveforms*

N°	Description des éléments
1	***Option Workspace ou espace de travail*** : se réfère à tous les instruments ouverts et à leur état actuel. Peut être chargé et sauvegardé ***Option settion :*** sélectionnez le menu Paramètres / Options dans la fenêtre Waveforms ***Option window*** : permet de sélectionner différentes préférences d'affichage et de configuration.
2	Listes des fenêtres en cours d'utilisation
3	***L'oscilloscope :*** offre toutes les fonctionnalités d'acquisition, de déclenchement et de visualisation de données, fournit toute la puissance analytique.
4	***Le générateur wavefrom:*** produit des Waveforms prédéfinis aime les ondes de sinus, de dents de scie et de triangle, ainsi que des Formes Waveforms (arbitraires) définies par l'utilisateur à des fréquences allant jusqu'à 20MHz. Il génère également des balayages entre les limites de fréquence définies par l'utilisateur et les sorties modulées AM ou FM.
5	Power Supplies : il contrôle les deux alimentations qui peuvent être réglées dans les gammes 0V à 5VDC et 0V à -5VDC avec une précision de 10mV.
6	***Data Logger :*** permet de consigner toutes les formes d'ondes générées dans leur système, y compris les formes d'onde DAQ.
7	***Logic Analyzer :*** Enregistre simultanément jusqu'à 16 signaux numériques jusqu'à 100 MS / s et les affiches au tant que signaux individuels, bus parallèles ou décodés en octets de données série sur les bus SPI, I2C ou UART.
8	***Pattern Generator :*** Permet de générer des signaux numériques individuels ou des motifs de bus de 16 signaux jusqu'à 100 SP / s.
9	***GPIO Generator :*** 16 E / S numériques peuvent être configurés comme des périphériques d'E / S virtuels comme les boutons-poussoirs, les commutateurs à glissière et les LED qui sont visualisés et contrôlés par les périphériques d'E / S virtuels affichés sur le périphérique hôte.
10	***Network Analyzer*** **:** L'analyseur de réseau entraîne un circuit avec une onde sinusoïdale balayée jusqu'à 10 MHz et mesure la réponse du circuit à mesure que la fréquence d'entrée change.
11	***Spectrum Analyzer*** : L'analyseur de spectre effectue un algorithme FFT ou CZT en temps réel sur les données de l'oscilloscope et affiche les données du domaine fréquentiel (spectre de puissance) ainsi que le signal du domaine temporel capturé.
12	***Voltmeter*** : contiens deux voltmètres autonome indépendants peuvent mesurer et afficher les signaux DC et AC (RMS et True RMS), en utilisant des mesures différentielles ou à simple extrémité.

Dans notre travail nous avons utilisé le générateur **waveforms** cette application marche avec la carte FPGA .pour générer un signal sinusoïdal comme montre la figure (III.11) nous allons choisir Waveforms avec une fréquence de **50 HZ** Et amplitude 2V, le choix de l'amplitude se fait en fonction de capteur utilisé car il prend que des valeurs positives. L'offset sera donc varier entre 0v et 2v .on clic sur RUN la carte va clignoter qui montre que le signal est sortie on obtient la figure suivante.

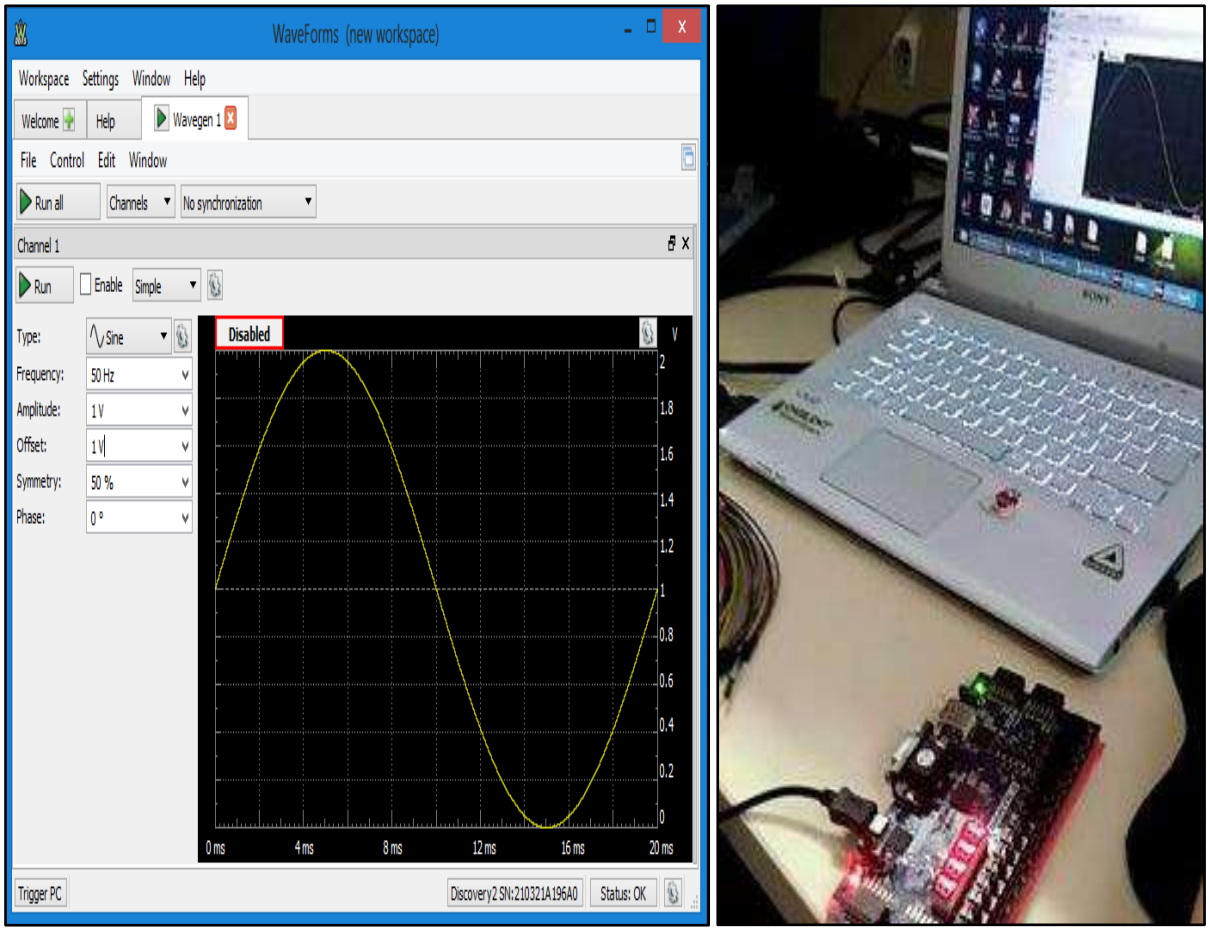


Figure III.11 *génère un signal* sinusoïdal *à* partir de waveforms

III.4.1.2. Les tests et les essais

Matériels utilises

- ✓ La carte FPGA
- ✓ Le convertisseur analogique numérique pmodAD1
- ✓ la carte AnalogDiscovery 2
- ✓ Pc

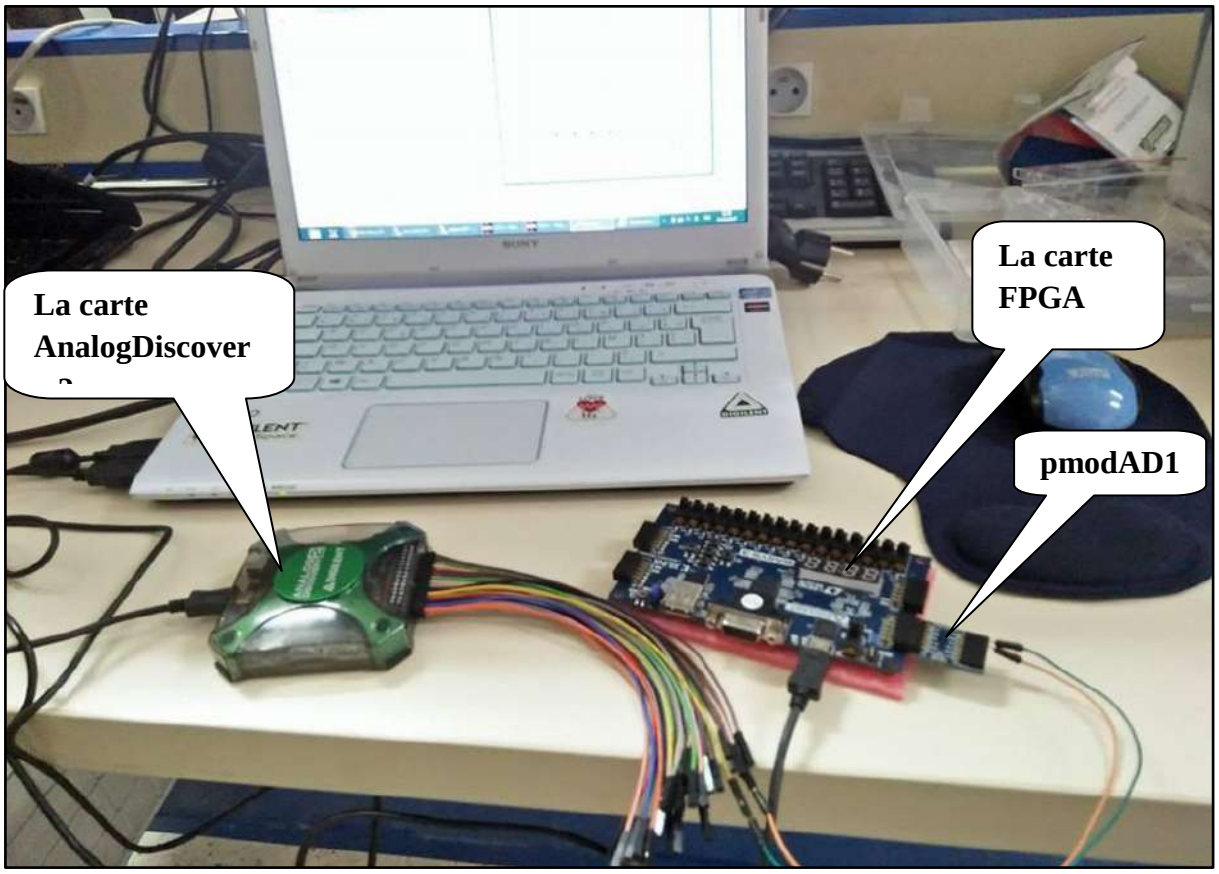
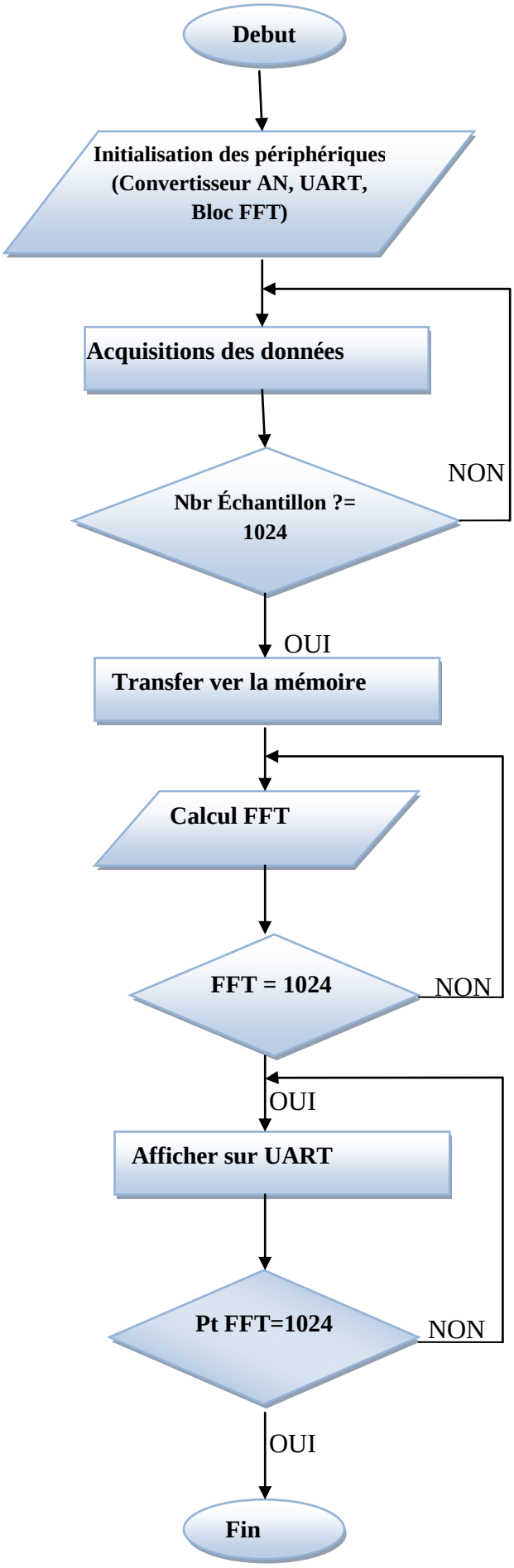


Figure III.12 *matériels utilises*

Dans cette partie nous allons relier le convertisseur AN avec le bloc FFT et l'UART, une fois que tous les dispositifs associes sont relies entre eux. Nous allons entamer nos tests sur la carte FPGA :

III.4.1.3. Organigramme


Debut
Initialisation des périphériques (Convertisseur AN, UART, Bloc FFT)
Acquisitions des données
Nbr Échantillon ?= 1024
NON
OUI
Transfer ver la mémoire
Calcul FFT
FFT = 1024
NON
OUI
Afficher sur UART
Pt FFT=1024
NON
OUI
Fin

Pournos tests nous avons supposé comme signaux sinusoïdal, rectangulaire et triangulaire de 500 HZ et 10 kHz pour calculer la FFT.

On fait entrer les signaux choisis dans le convertisseur pmodAD1 qui se trouve au niveau de la carte FPGA , lorsque le signal va passer sur la carte , celle-ci va opérer l'échantillonnage avec 1024 points et faire les calcules de la FFT , une fois les calcules faite le transfert se fera vers le logiciel **HERCULES** qui fera l'affichage des résultats qui a sont tour le transmettra vers la mémoire.

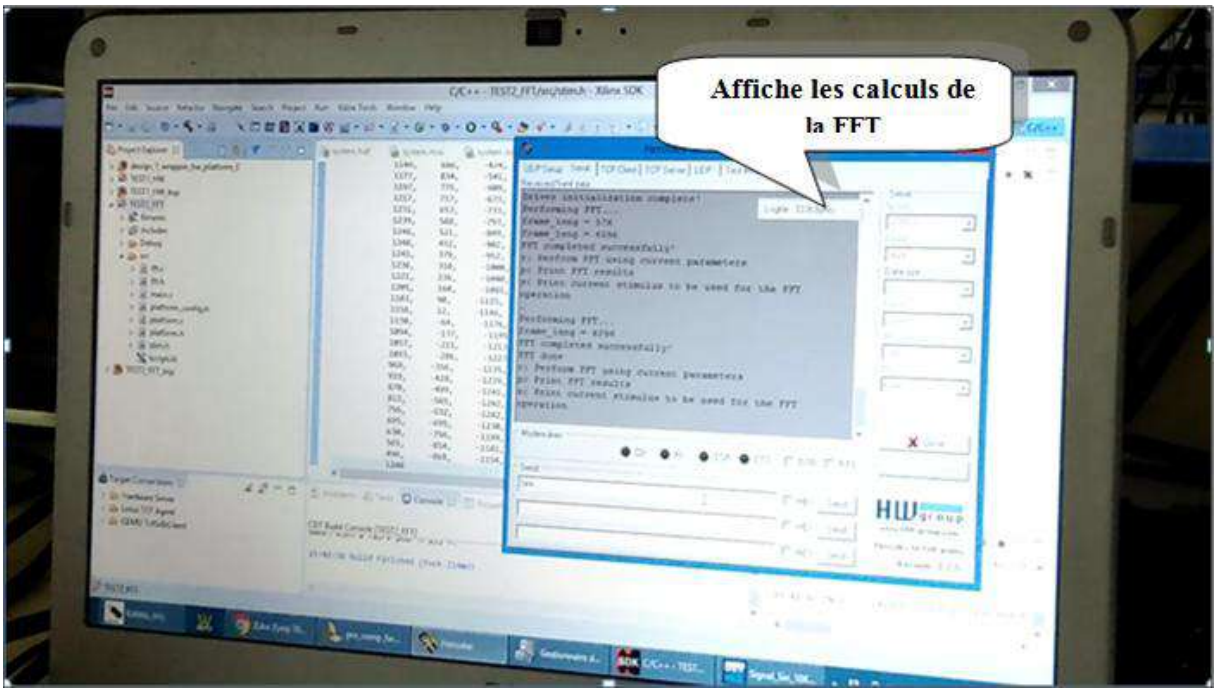


Figure III.13 *Calcul de la FFT avec le logiciel Hercules*

Enfin nous obtenons les résultats et nous ferons la configuration de chaque signal choisis et le sauvegarder dans la mémoire .Les résultats obtenus serons affiches sur MATLAB.

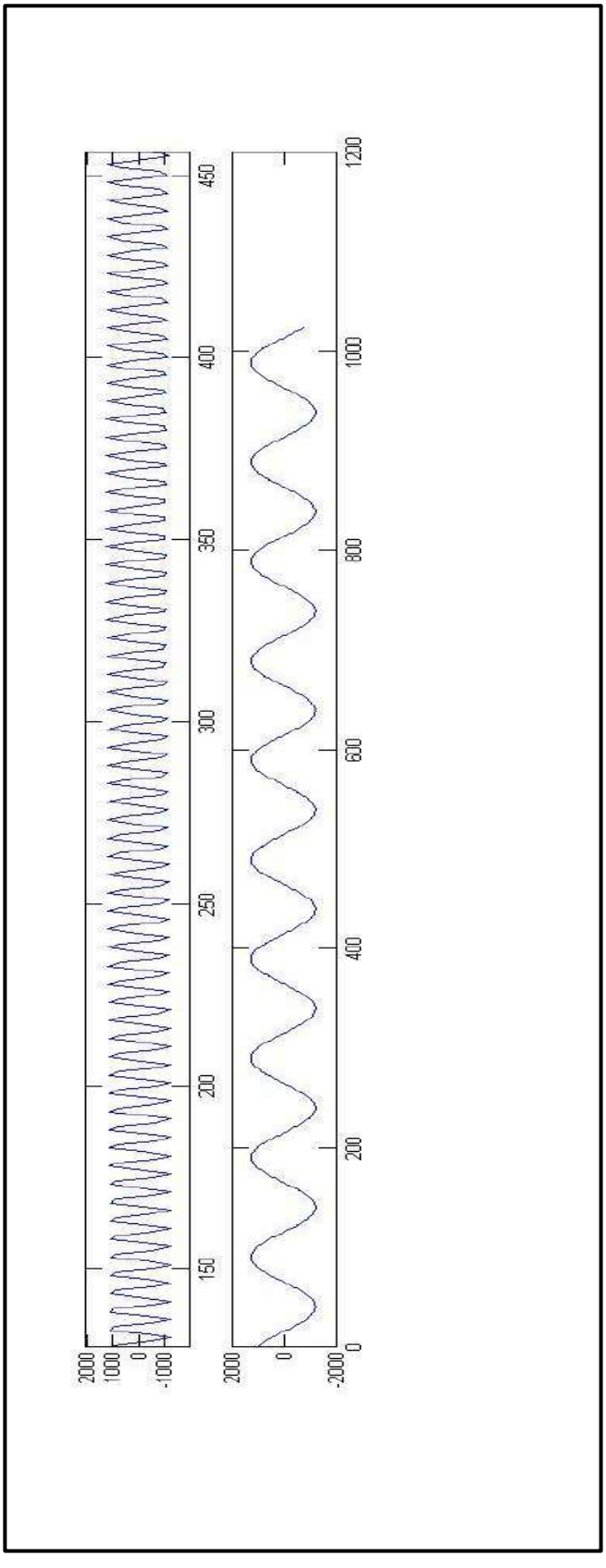


Figure III.14 *signal sinusoïdal choisi signal de 10kHz et 500 HZ*

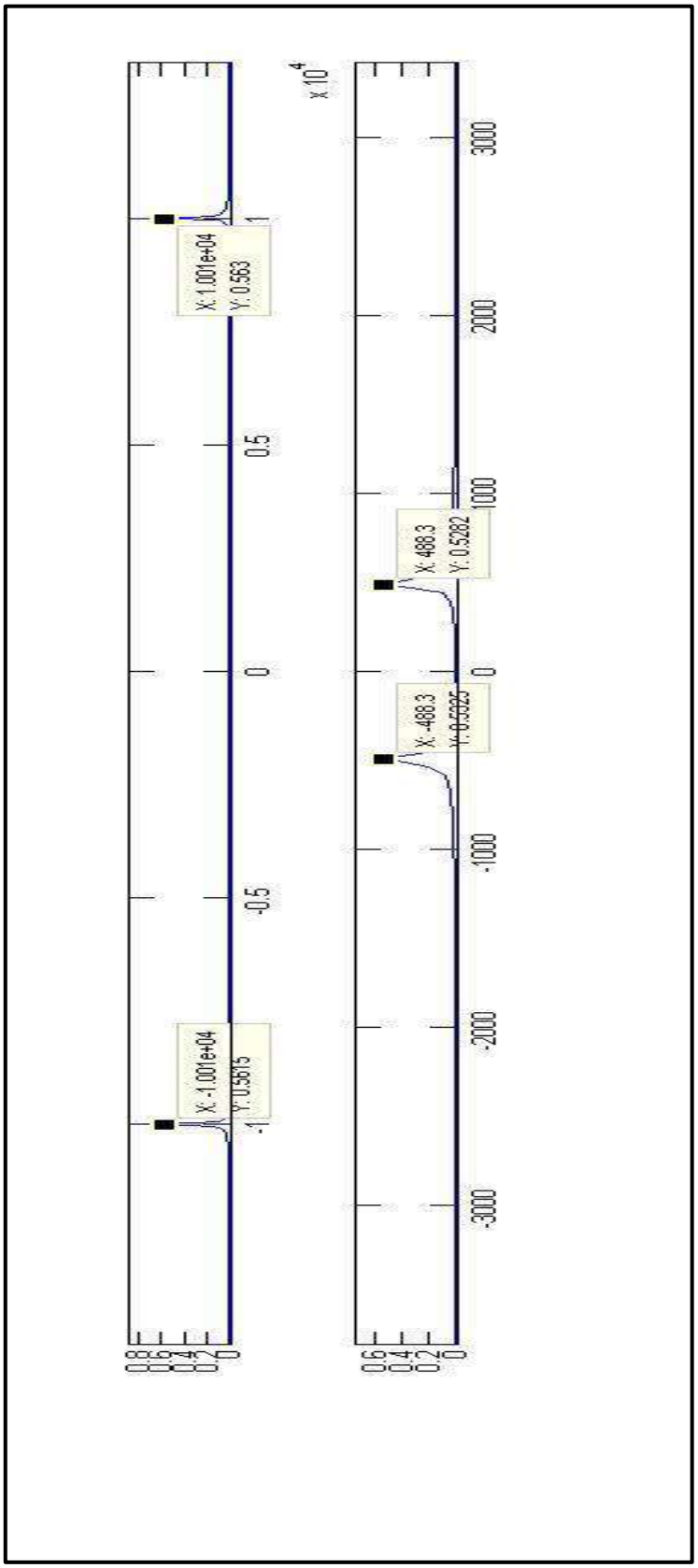


Figure III.15 *les résultats obtenus après les calculs de la FFT du signal sinusoïdal choisi*

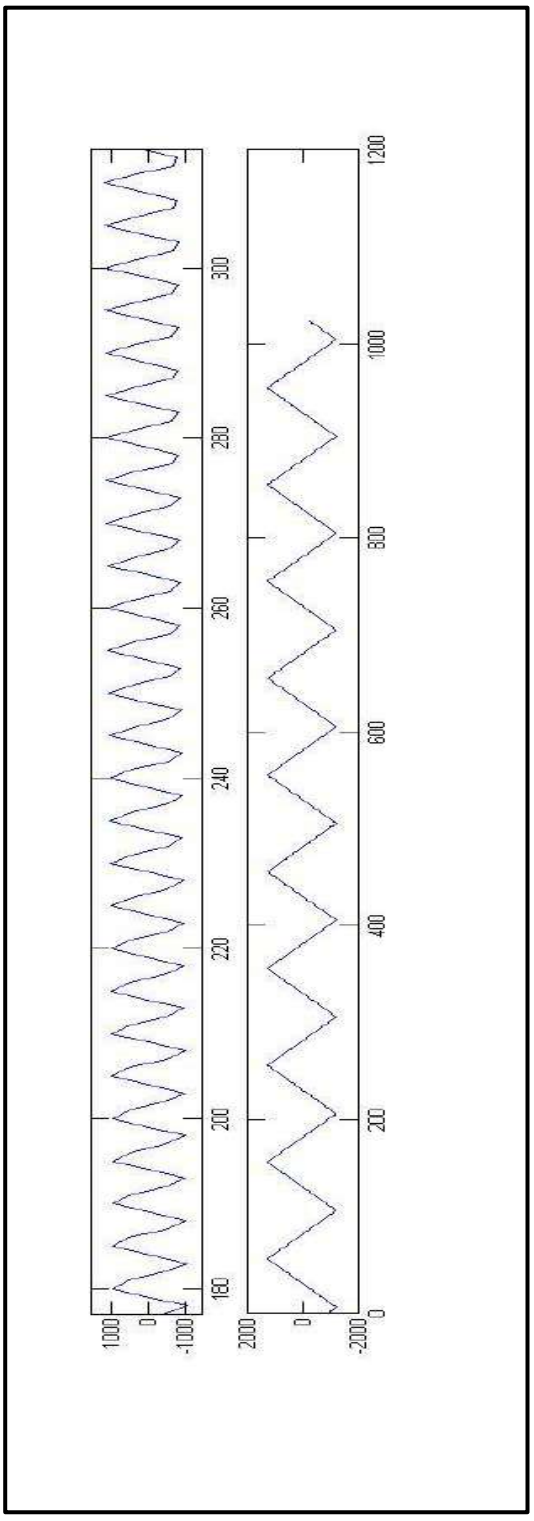


Figure III.16 *signal triangulaire choisi de 10 kHz et de 500HZ sinusoïdal choisi*

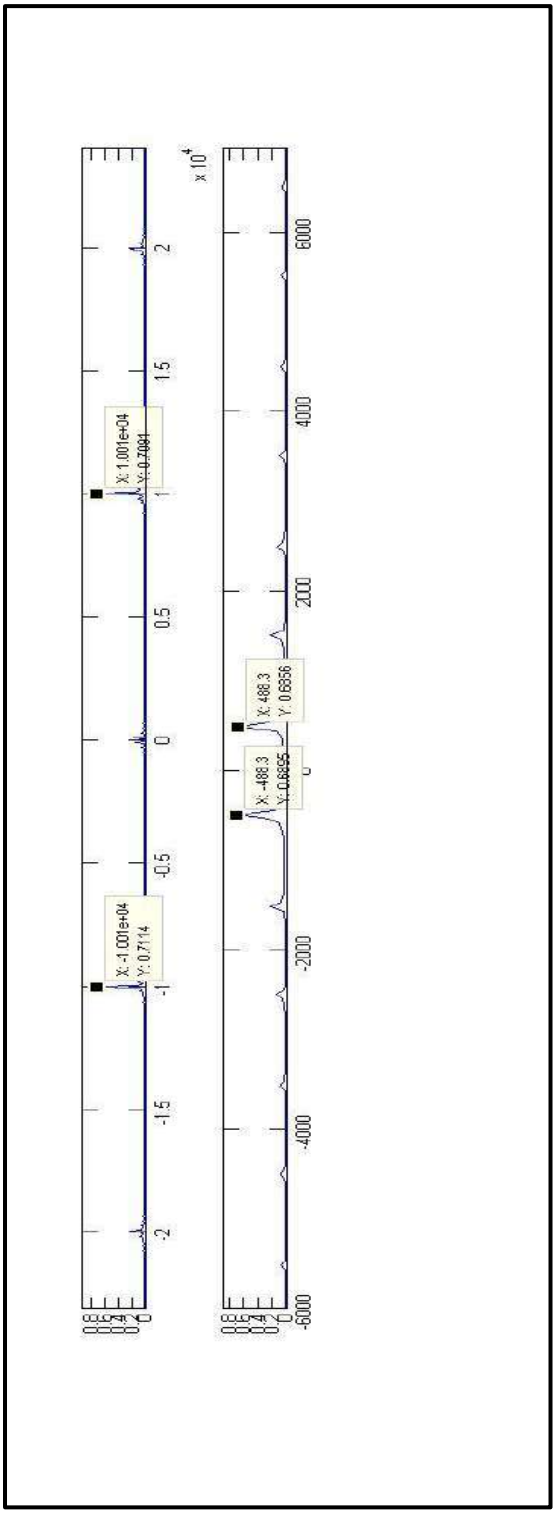


Figure III.17 *les résultats obtenus après les calculs de la FFT du signal triangulaire choisi*

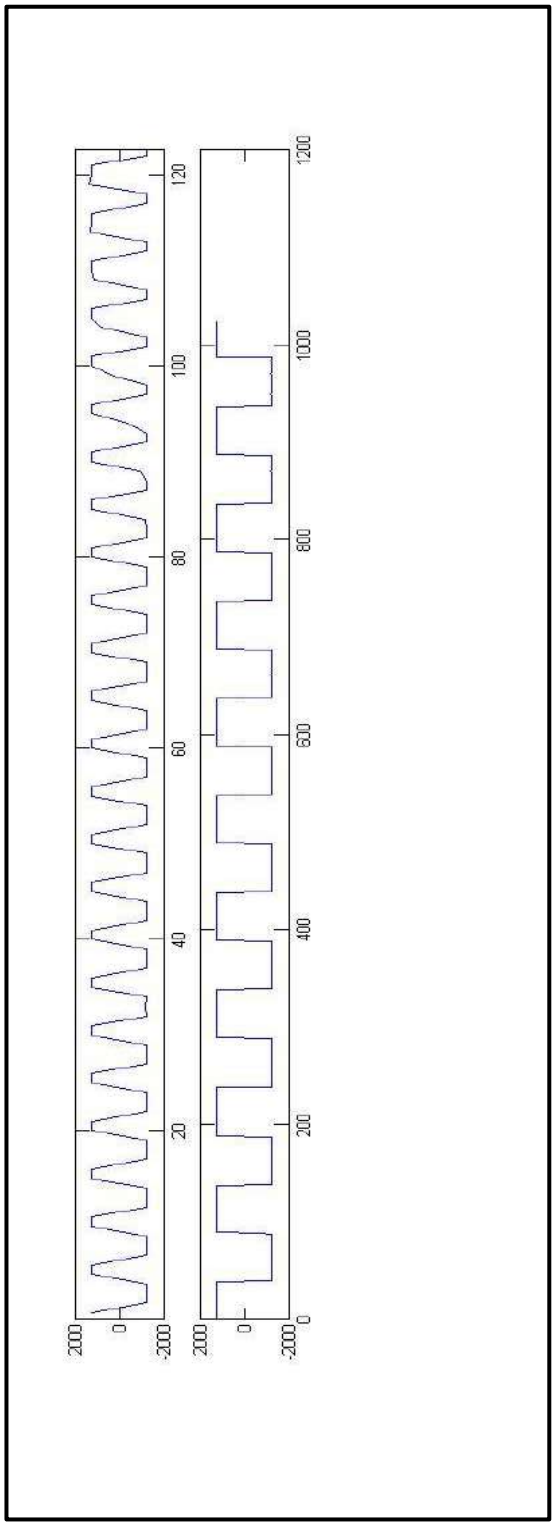


Figure III.18 *signaux caret choisi de 10 kHz et de 500HZ*

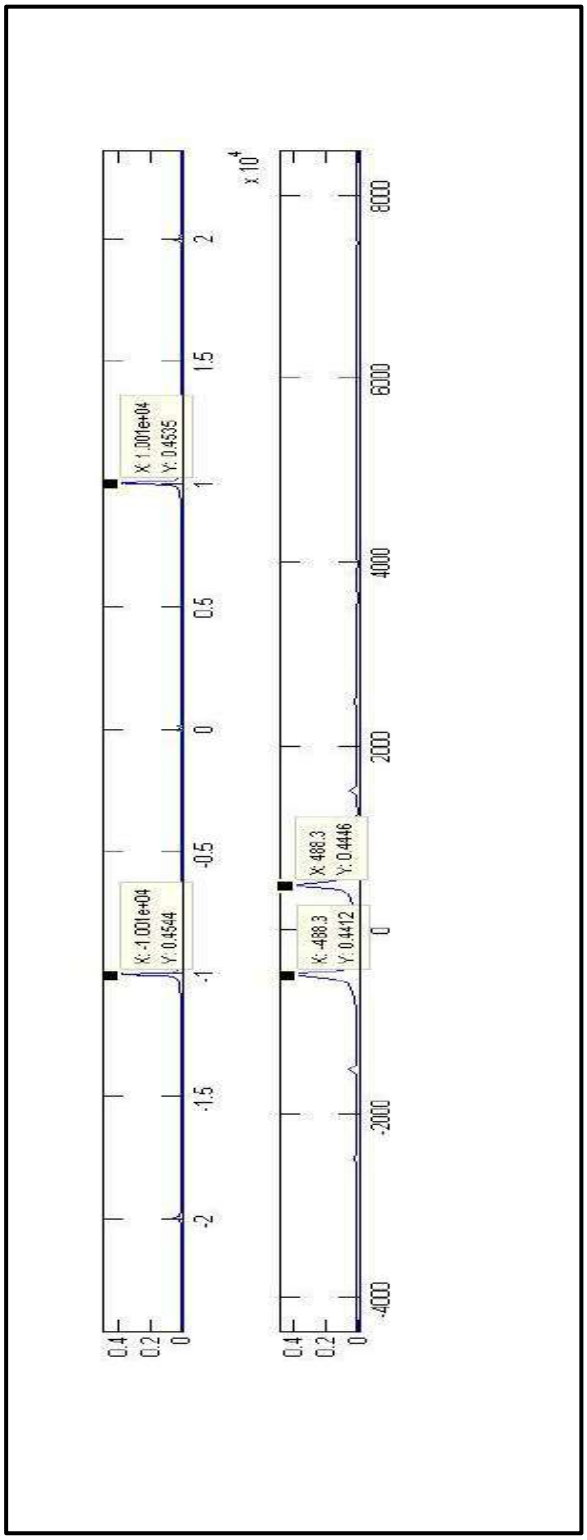


Figure III.19 *les résultats obtenus après les calculs de la FFT du signal caret choisi*

Les résultats de la FFT obtenus sur Matlab on les a illustrés dans le tableau suivant :

Tableau III.5 *Calcul du pourcentage d'erreur*

Signal généré	Valeur utilise pour générer le signal	Valeurs obtenus	Erreur (%)
Signal Sinusoïdal	500 HZ	48830,00%	2.34%
	10 kHZ	1.001 e+04	0,10%
Signal Triangulaire	500 HZ	488.3	2,34%
	10 kHZ	1.001 e+04	0,10%
Signal Carré	500 HZ	488.3	2,34%
	10 kHZ	1.001 e+04	0,10%

Les résultats du calcules de FFT représenté sur **les figures(Figure III.14,Figure III.15,Figure III.16,Figure III.17,Figure III.18,Figure III.19)** montre que le traitement effectuer avec la plateforme proposé dans ce travail est identique avec la théorie de l'FFT (**Annexe C**). On a noté aussi qu'une erreur fixe est obtenu pour chaque fréquence (**Tableau III.5)** où pour une fréquence de 500 Hz une erreur de 2.34 % est obtenu, et une erreur de 0.1% pour une fréquence de signal de 10 kHz, vu que cette erreur est fixe pour les trois signaux proposé on a estimé que l'origine de cette erreur est due au générateur des signaux.

III.5 Conclusion

Dans ce chapitre nous avons présenté d'une façon détaillée les différentes étapes que nous avons suivies pour l'implémentation sur un circuit FPGA pour le calcul de l'FFT.

Pour faire réussir notre conception nous avons commencé par la conception sur le logiciel VIVADO après nous avons passe a l'étape suivante qui est la réalisation sur matériels

La comparaison des calculs obtenu par nos conceptions avec ceux obtenus en utilisant le MATLAB nous a permis la vérification du bon fonctionnement de nos deux conception.

CONCLUSION

Conclusion générale

Le but de ce travail est basé sur la réalisation de la FFT qui est très utile pour le traitement numérique de signal et dans le domaine de communication numérique a base de circuit FPGA généré pas Microblaze une rapide.

La première partie nous avons fait une étude sur les circuits reconfigurables par programmation, ce qui nous a permis d'acquérir les notions de base sur ce type de circuits de point de vue architecturel et méthodologie de conception. L'avantage majeur de ces circuits et leurs reconfigurabilités ainsi que la rapidité et facilité d'implémentation qui est due à l'existence de plusieurs outils et logiciels d'implémentation.

Dans la deuxième nous a permis d'étudier les différentes architectures d'implémentation de l'FFT ainsi que leurs complexités arithmétiques, leurs complexités de contrôle et leur rapidité. Cette étude nous a permis de conclure que les architectures basées sur l'approche Radix-2 et Radix -4.

La dernière partie de ce travail était consacré à la réalisation pratique de l'implémentation sur un circuit FPGA et réaliser la FFT généré par Microblaze.

Pour cela nous avons commencé par la maitrise du langage de développement VHDL. Le profit de cette étape était bien de faire un passage d'un schéma de description d'une architecture vers le programme de description de cette architecture. Puis nous sommes passés à la prise en main de l'outil de développement vivado qui fournit un environnement complet de développement d'application à base de circuits FPGA.

Après avoir maitrisé le langage VHDL et l'outil vivado nous sommes passés au calcule de la FFT ou nous avons choisis des signaux et on les a fait passé dans un convertisseur analogique d'où il va faire l'échantillonnage et les calculs après stocker dans la mémoire et afficher les résultats en Matlab pour vérifier les résultats.

Notations

N **:**la longueur de la transformée.

$\boldsymbol{w_N^{nk}}$**:**Coefficientsde base de la DFT.

4M2A : 4 multiplications réelles et 2 additions réelles.

3M3A : 3 multiplications réelles et 3 additions réelles.

K: le numéro de l'étage.

F échant:Fréquence d'échantillonnage.

ANNEXE

Annexe A

- Datasheet du Basys3 FPGA

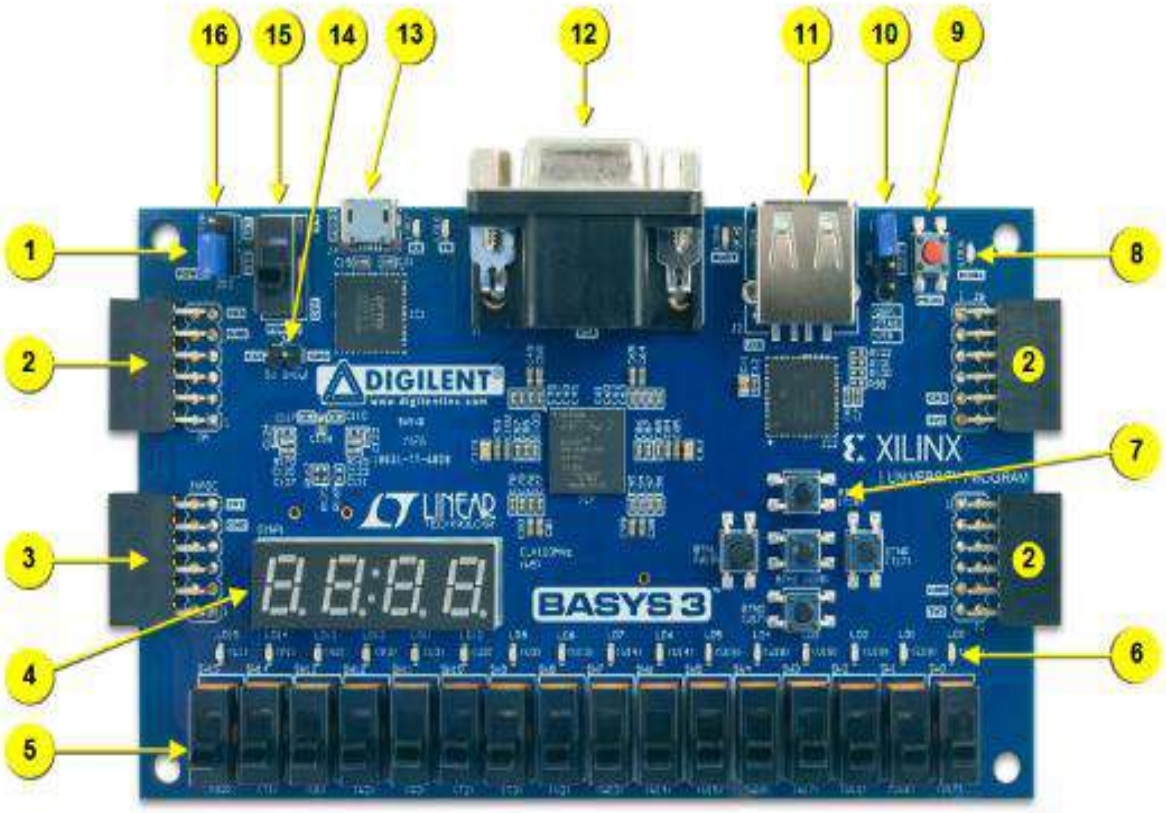


Figure 1. Basys 3 FPGA board with callouts

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Table 1. Basys 3 Callouts and component descriptions.

AnnexeB

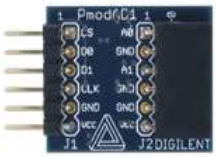
- PmodAD1

PmodAD1: Two 12-bit A/D inputs

Overview

Product Description

The PmodAD1 is an analog-to-digital converter module powered by the Analog Devices AD7476A. Users can simultaneously receive 12 bits of resolution for 2 individual channels at up to 1 MSPS by communicating with the module through an interface similar to SPI.



Key Features & Benefits

- Communicates via an SPI-like interface (see datasheet for pinout)
- Simultaneous A/D conversion at up to 1 MSa per channel
- Two 2-pole Sallen-Key anti-alias filters
- Two Analog Devices AD7476 12-Bit A/D Converters

What's Included

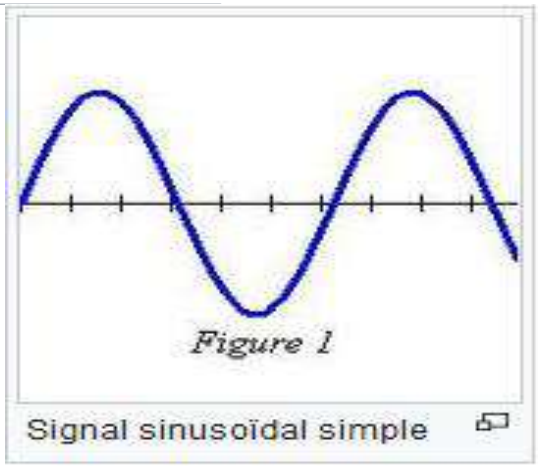
- Custom antistatic Pmod packaging
- PmodAD1

- Un Signal sinusoïdal

Signal sinusoïdal

Un **signal sinusoïdal** est un signal (onde) dont l'amplitude, observée à un endroit précis, est une fonction sinusoïdale du temps.

- La fonction sinus est une fonction qui permet de calculer le sinus d'un angle à partir de la valeur de cet angle.
- Une sinusoïde est la forme que prend cette fonction (voir Figure 1).



Signal sinusoïdal simple

AnnexeC

Caractéristiques d'un signal sinusoïdal [modifier | modifier le code]

Un signal sinusoïdal est caractérisé par son amplitude maximale et sa fréquence. Il peut s'écrire sous la forme :

$$s(t) = A\,\sin(\omega t + \varphi)$$

avec :

- A : amplitude de la grandeur, appelée aussi valeur de crête, dans l'unité de la grandeur mesurée
- ω : pulsation de la grandeur en rad s^{-1}
- $\omega t + \varphi$: phase instantanée en rad
- φ : phase à l'origine en rad (souvent fixée par l'expérimentateur)

La pulsation, la fréquence et la période sont liés par les relations :

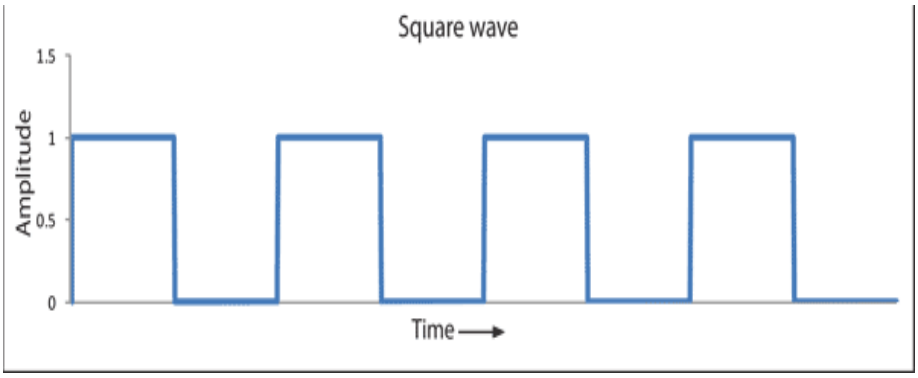
$$\omega = 2\pi f = \frac{2\pi}{T}$$

- Un Signal carré

Signal carré

Pour les articles homonymes, voir Carré (homonymie).

Un **signal carré** est une sorte d'onde non–sinusoïdale que l'on rencontre le plus souvent en électronique ou dans le cas du traitement du signal. Un signal carré idéal alternerait régulièrement et instantanément entre deux niveaux.



AnnexeD

Autres définitions [modifier | modifier le code]

Le signal carré a de nombreuses définitions qui sont toutes équivalentes, sauf dans les discontinuités.

- on peut le définir simplement comme la fonction signe d'une sinusoïde :

$$x(t) = \operatorname{sgn}(\sin(t))$$

qui sera égale à 1 quand la sinusoïde est positive, −1 quand elle est négative, et 0 dans les discontinuités ;

- on peut aussi le définir selon la fonction de Heaviside $u(t)$ ou selon la fonction porte $\sqcap(t)$;

$$x(t) = \sum_{n=-\infty}^{+\infty} \sqcap(t - nT) = \sum_{n=-\infty}^{+\infty} \left(u\left(t - nT + \frac{1}{2}\right) - u\left(t - nT - \frac{1}{2}\right)\right)$$

$T = 2$ pour un rapport cyclique de 50 % ;

- le signal carré peut aussi être défini selon :

$$x(t) = \begin{cases} 1, & 0 < t \leq T \\ 0, & T < t \leq 2T \end{cases}$$

quand

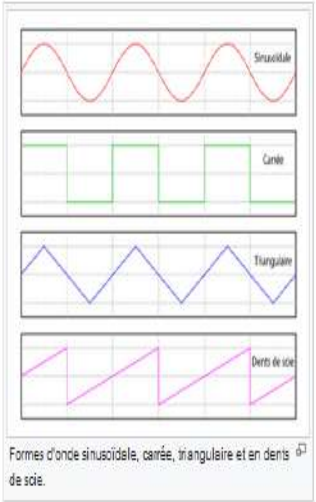
$$x(t + T) = x(t)$$

Un Signal triangulaire

Signal triangulaire

Pour les articles homonymes, voir Triangle (homonymie).

Un **signal triangle** est une sorte d'onde non sinusoïdale que l'on rencontre le plus souvent en électronique ou dans le cas du traitement du signal. Il est périodique, linéaire par morceaux et continu. Tout comme le signal carré, il ne contient que des harmoniques impairs.



Formes d'onde sinusoïdale, carrée, triangulaire et en dents de scie.

Décomposition en série de Fourier [modifier | modifier le code]

Il est possible de décomposer le signal triangle sous la forme d'une série de Fourier infinie :

$$x_{\text{triangle}}(t) = \frac{8}{\pi^2} \sum_{k=0}^{\infty} (-1)^k \frac{\sin\left((2k+1)\omega t\right)}{(2k+1)^2}$$
$$= \frac{8}{\pi^2} \left(\sin(\omega t) - \frac{1}{9}\sin(3\omega t) + \frac{1}{25}\sin(5\omega t) - \cdots \right)$$

où ω est la pulsation du signal d'origine.

Le signal triangle étant continu, la synthèse sonore additive de ses premiers harmoniques obtenus par décomposition en série de Fourier en constitue une bonne approximation.

Autres définitions [modifier | modifier le code]

- On peut aussi définir le signal triangle d'amplitude 1, de valeur moyenne 0 et de période 2a par :

$$x(t) = \frac{2}{a}\left(t - a\left\lfloor \frac{t}{a} + \frac{1}{2} \right\rfloor\right)(-1)^{\left\lfloor \frac{t}{a} + \frac{1}{2} \right\rfloor}$$

où la notation $\lfloor n \rfloor$ désigne l'image de n par la fonction partie entière.

- Le signal triangle peut aussi être vu comme la valeur absolue du signal en dents de scie :

$$x(t) = \left| 2\left(\frac{t}{a} - \left\lfloor \frac{t}{a} + \frac{1}{2} \right\rfloor\right)\right|$$

- On peut aussi exprimer le signal triangle comme l'intégrale du signal carré :

$$\int \operatorname{sgn}(\sin(x))\, dx$$



Animation de la synthèse sonore additive du signal triangle avec un nombre croissant d'harmoniques. Voir série de Fourier pour une description mathématique.

Bibliography

[1] : CONCEPTION DE SYSTÈMES "ON CHIP" (SOC). (n.d.). CentraleSupélec Executive Education, https://www.exed.centralesupelec.fr/fr/formation-continue/er18-17-conception-de-systemes-chip-soc.

[2] **ATAT, Y**. (Le 21 Mai 2007). Conception de haut niveau des MPSoCs a partir d'une specification simulink:Passerelle entre la conception d'algorithmes et la conception d'architectures.

[3] **EPFL, E. S. (n.d.)**. Circuits reconfigurables:Les FPGAs.

[4] **SAHOUR, A**. (n.d.). IMPLEMENTATION SUR FPGA D'UN ALGORITHME DE DEBRUITAGE .

[5] **ABDELGHANI, G**. (n.d.). Conception et Implémentation d'un Corrélateur Numérique sur FPGA.

[6] **BILLEL, C. S**. (n.d.). ETUDE ET REALISATION D'UN NŒUD SANS FIL A BASE D'UN CIRCUIT FPGA .

[7] **BAHRI, I.** (November 13, 2011). Contribution of FPGA-based System-on-Chip controllers for embedded AC drive applications.

[8] **altera. (n.d.)**. Retrieved from https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ab/ab18_soc_fpga.pdf.

[9] **HELLER, D**. (n.d.). Spécification et conception des systèmes temps-réels.

[10]Hayder MRABET, W. B. Intégration d'un FPGA dans un système sur puce.

[11] PEREZ-SEVA, J.-P. (le 24 août 2009). Les optimisations d'algorithmes de traitement de signal sur les architectures modernes parallèles et embarquées.

[12] LLC, G. B. (2010). Transformée Du Signal: Transformée de Fourier Rapide, Transformée en Cosinus Discrčte, Lifting en Ondelettes, Transformée en Z.

[13] lenart, t. (2005). A Hardware Accelerator for Digital Holographic Imaging.

[14]Tzi Dar chiueh, p. y. (2007). OFDM Baseband Receiver Design for Wireless Communications

[15]Fast Fourier Transform.Récupère sur :https://www.omicsonline.com/open-access/fpga-based-implementation-of-fft-processor-using-different-architectures

[16] William Chin Cheng Chung, H. C. (2014). Intelligent Systems and Applications.

[17]Bourennane, S. (2015). Wireless Communication and Sensor Network.

[18]Microblaze. Disponible sur : https://www.xilinx.com/support/documentation/ip_documentation/microblaze

[19] Pillai, A. R. (2007). Implementation of MicroBlaze Processor with Application Specified Peripherals on FPGA.

[20]Fast Fourier Transform, FFT. (1 Mai 2003). Retrieved from https://www.unilim.fr/pages_perso/jean.debord/math/fourier/fft.htm#section5.

[21] Boizumault, P. (1988). Prolog: l'implantation.

[22] COPYRIGHT 2017 XILINX INC. (2017). Récupéré sur https://www.xilinx.com/support.html.

[23] Press, I. C. (1994). Proceedings.

[24]COPYRIGHT 2017 XILINX INC. (2017). Récupéré sur https://www.xilinx.com/support/ SPI .html.

[25] Grasset, J. (1906). Le psychisme inférieur: étude de physiopathologie clinique des centre

[26]COPYRIGHT 2017 XILINX INC. (2017). Récupéré sur https://www.xilinx.com/support/ Interconnect.html.

[27] Stanley Mazor, L. P. (1993). A Guide to VHDL.

[28]COPYRIGHT 2017 XILINX INC. (2017). Récupéré sur https://www.xilinx.com/support/ UART LITE .html.

[29]ADC. Disponible sur : https://www.xilinx.com/support/documentation/ip_documentation/ADC

[30] Dominique Houzet, L. B. (2006). Conception de circuits en VHDL et VHDL-AMS.

[31] Oates, C. (2008). Cardiovascular Haemodynamics and Doppler Waveforms Explained.