République **A**lgérienne **D**émocratique et
Populaire

Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche
Scientifique

Université des **S**ciences et de la **T**echnologie d'**O**ran **M**ohamed **B**OUDIAF
(**U.S.T.O.M.B**)

Faculté de **G**énie **E**lectrique

Département
d'Electronique

MEMOIRE EN VUE DE L'OBTENTION DU DIPLOME DE MAGISTERE

Spécialité : **ELECTRONIQUE**

Option : **INSTRUMENTATION SPATIALE**

PRESENTE PAR :

M^r^ **GUETTAT ABDELGHANI**

SUJET DE MEMOIRE :

Conception et Implémentation d'un Corrélateur Numérique sur FPGA

Soutenu le **09/10 / 2012** devant le jury composé de :

M^r^ **BELBACHIR**	Président	**Professeur**, **USTO**
M^r^ **ALI BACHA**	Examinateur	**Professeur, USTO**
M^r^ **CHALABI**	Examinateur	**Maitre de Conférences, USTO**
M^r^ **N. BOUGHANMI**	Encadreur	**Professeur, USTO**
M^r^ **H. BENDOUMA**	Co-Encadreur	**Maitre de Conférences, USTO**

Remerciements

Comme un tel travail ne s'effectue jamais seul, j'aimerais au début remercier par ces quelques mots tous ceux qui, de prés ou de loin, on contribué à le rendre possible et m'ont aidé à le réaliser.

Je tiens à remercier tout particulièrement *MR. BOUGHANMI,* Professeur en électronique à l'USTO et le Professeur *MR.BENDOUMA* pour avoir accepté l'encadrement scientifique de ce mémoire. Je leur suis reconnaissant pour m'avoir dirigé et assisté tout au long de ce travail. Je les remercie également pour leurs nombreux conseils et remarques qui m'ont été d'une grande utilité.

J'exprime toute ma gratitude à *Mr. BOUTBAL* pour l'intérêt qu'il a apporté à ce mémoire et pour ces précieux conseils et orientations.

Je tiens à remercier aussi le président et les membres de jury, qui m'ont fait l'honneur d'accepter de juger ce travail.

Je remercie également tous mes collègues et amis sans oublier personne.

Que toutes et tous ceux qui ont fait que ce travail aboutisse, trouvent ici l'expression de mes remerciements les plus sincères.

dédicace

Je dédie ce travail à mes très chers parents pour le courage et le soutien pendant mes années d'études

A mes meilleurs amis :

- *BENAIED BRAHIM.*
- *HADDAD ABDELKADER.*
- *GOUJIL MOHAMED AMINE.*
- *BENMAZARI MUSTAPHA*
- *BOUDANE FATIMA*
- *MESSAOUDI AMINA*

Ainsi qu'a mes frères :

ZINEDINE, ABDENOUR, HABIB.

Et ma petite nièce Meriem

SOMMAIRE

Sommaire

CHAPITRE II : Conception des circuits Numériques

CHAPITRE III : Présentation de la réalisation

CHAPITRE IV : L'implémentation et résultats expérimentaux

LISTE DES ACRONYMES ET ABREVIATIONS

ALU: ***A****rithmetic-****L****ogic* ***U****nit*

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

CAO : ***C****onception* ***A****ssistée par* ***O****rdinateur.*

CLB : ***C****onfigurable* ***L****ogique* ***B****loc.*

CPLD : ***C****omplex* ***P****rogrammable* ***L****ogique* ***D****evice.*

CPLD: ***C****omplex* ***P****rogrammable* ***L****ogic* ***D****evice.*

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

EPROM: ***E****rasable* ***P****rogrammable* ***R****ead* ***O****nly* ***M****emory.*

EEPROM: ***E****lectrically* ***E****rasable* ***P****rogrammable* ***R****ead* ***O****nly* ***M****emory.*

EEPLD: ***E****lectrically* ***E****rasable* ***P****rogrammable* ***L****ogic* ***D****evice.*

FPGA: ***F****ieled* ***P****rogrammable* ***G****ate* ***A****rray.*

IOB: ***I****nput* ***O****utput* ***B****loc.*

IP: ***I****ntellectual* ***P****roperties.*

LAB: ***L****ogical* ***A****rray* ***B****loc*

LE: ***L****ogical* ***E****lement*

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

MAC: ***M****ultiply* ***A****ccumulate*

PLL: ***P****hase-****L****ocked* ***L****oop*

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

PROM: ***P****rogrammable* ***R****ead* ***O****nly* ***M****emory.*

PAL: ***P****rogrammable* ***A****rray* ***L****ogic.*

PLD: ***P****rogrammable* ***L****ogic Device.*

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

RTL: ***R****egister* ***T****ransfer* ***L****evel.*

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

SOC: ***S****ystems* ***O****n* ***C****hips.*

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

VLSI: ***V****ery* ***l****arge* ***S****cale* ***I****ntegration*

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

INTRODUCTION **G**ENERALE

Les applications embarquées font appel à des algorithmes pour traiter différents types de données (traitement du signal et des images, compression audio ou vidéo) et utilisent des algorithmes de plus en plus sophistiqués dont la complexité n'a cessé de croître. Du fait de cet accroissement de la complexité, ces applications, et en particulier celles de traitement de signal ou de l'image, demandent toujours de plus en plus de puissance de calcul. Cet aspect s'avère encore plus crucial, lorsque l'implémentation de ces algorithmes nécessite un traitement des données en temps réel. Il s'agit donc de respecter des contraintes temporelles imposées entre deux occurrences successives d'un signal d'entrée (période) ou entre un signal d'entrée et un signal de sortie. Actuellement, les algorithmes de traitement de signal présentant une contrainte temps réel voient leurs implémentations s'orienter vers des architectures numériques dédiées. Les solutions architecturales habituellement utilisées sont des ordinateurs, des cartes électroniques à base de circuits intégrés ou de processeurs spécialisés (DSP) ou à architecture multi-composant (processeurs spécialisés, des circuits de type VLSI et/ou des FPGAs). Le choix de l'architecture est fonction de nombreux critères fortement dépendants de l'application.

Intégrer une grande puissance de traitement dans un système embarqué implique généralement des architectures hétérogènes et parallèles pour rechercher les meilleurs compromis de performances, taille et consommation. Ces architectures peuvent être réalisées avec des composants offrants une grande flexibilité comme les DSPs et/ou des composants offrants de bonnes performances comme les ASICs et les circuits logiques programmables (FPGAs). Pour réaliser de tels systèmes, deux approches sont possibles : répartir le système sur plusieurs composants ou tout intégrer dans un unique composant. Cette dernière approche repose sur le concept de SoC (System On-Chip). **[1]**

Aujourd'hui, les circuits logiques programmables offrent aux concepteurs la possibilité de réaliser des applications numériques relativement complexes dans un unique circuit en fournissant performance et flexibilité. En effet, les dernières générations de circuits configurables disposent, en plus des traditionnelles cellules logiques, de blocs mémoires RAM, de multiplieurs câblés, d'entrées/sorties dédiées aux communications rapides et de cœurs de processeurs embarqués ; utilisés auparavant comme circuit de prototypage pour la

réalisation d'ASIC ou comme circuit d'interface pour des processeurs dédiés aux calculs. L'optimisation des algorithmes à l'architecture du composant (parallélisme, Look Up Tables, etc.) augmente les performances jusqu'à dépasser celles d'implémentation sur DSP et ASICs. De plus, la description d'une architecture optimisée en code synthétisable VHDL avec une approche hiérarchique et modulaire apporte une bonne flexibilité au système. Le FPGA est le circuit logique programmable le plus couramment utilisé de nos jours, de part ses capacités, sa vitesse et sa grande flexibilité. Ces architectures mixent généralement des composants matériels et logiciels travaillant en concurrence afin de répondre le plus efficacement à la contrainte temporelle imposée. Des études de cas montrent en effet que de tels SoCs présentent actuellement une structure architecturale des plus adéquates en terme de performances et d'efficacité d'implantation. **[2]**

Dans notre étude nous avons étudié la fonction de corrélation qui se trouve à la jonction de diverses théories mathématiques. Elle présente des caractéristiques intéressantes pour le traitement des signaux et plus particulièrement dans le domaine des télécommunications, l'astronomie…….etc.

L'algorithme de la fonction de corrélation recommandé par le standard est pris comme modèle par la quasi-majorité des concepteurs de logiciels de la corrélation. Cependant, la forte complexité calculatoire de cet algorithme associée au besoin de répondre à des applications de hautes performances amène les concepteurs à se tourner de plus en plus vers des solutions matérielles.

En effet, si l'on veut encore plus de performance (applications nécessitant des vitesses élevées), ou permettre la corrélation sur des systèmes embarqués, alors une implantation totalement matérielle, voire mixte (matérielle/logicielle), sur des processeurs dédiés est préférable à une implémentation purement logicielle. Cette implantation s'effectue sur des plates-formes matérielles dédiées, à travers des composants programmables (FPGA, DSP,…), qui permettent d'effectuer la validation/vérification du système en des temps très courts.**[3]**

Le travail que nous présentons dans ce mémoire vise la transposition de l'architecture de la fonction de corrélation vers une architecture implantable sur FPGA du type CYCLONE III de ALTERA, dans un but de mettre en place une plate-forme de prototypage rapide et évolutive.

Cette transposition consistera à décrire en langage VHDL l'architecture et à l'adapter aux contraintes physiques de notre prototype. Dans le cadre de ce travail, nous utiliserons une carte de développement FPGA pour simplifier la conception et le développement du système.

Organisation du mémoire :

Ce mémoire se découpe en quatre parties, organisées en chapitres.

Le chapitre I présente l'aspect théorique de la corrélation, nous y exposons les bases théoriques de la fonction de corrélation continue et numérique, ces applications, ainsi que les différentes architectures utilisées et enfin quelques applications de la fonction de corrélation.

Le chapitre II sera consacré à la conception des circuits numériques. Quelques généralités sur les circuits logiques programmables et sur les circuits FPGA sont préalablement données, et enfin nous clôturons par la description de l'architecture générale d'un composant FPGA type CYCLONE III en détail et ainsi une présentation de la carte START-KIT du circuit FPGA.

Le chapitre III décrit la réalisation matérielle de notre architecture de la fonction de corrélation. Nous débutons ce chapitre par des notions générales sur la conception synchrone de la machine d'état. Une vue du circuit et les séquenceurs (Logique de contrôle) avec des signaux de contrôle ainsi les équations logiques sont présentées.

Le chapitre IV pilote l'implantation matérielle de notre architecture du corrélateur numérique. Nous présentons les résultats de la simulation de notre système. Et nous avons établi une comparaison entre les résultats de MATLAB et ceux issus de l'implémentation VHDL de notre architecture. Nous terminons par une conclusion générale qui rassemble les différentes évaluations et perspectives que l'on peut tirer de ce travail.

CHAPITRE I

ASPECT THEORIQUE DU CALCUL DE CORRELATION

I)- INTRODUCTION

L'application des méthodes de corrélation à divers problèmes de traitement de signal et d'exploitation de mesure n'est pas nouvelle dans son principe. Il y a déjà plusieurs décennies que de nombreux chercheurs ; dans le monde entier ; ont montré la richesse de cette méthode et les résultats remarquables que l'on était en droit d'en attendre, dans beaucoup de cas ou, précisément, les méthodes classiques étaient en défaut. **[3]**

Ce chapitre présente une chaine de définitions et d'explications sur l'opération de corrélation qui est proche sous certains aspects mais qui a un objectif bien différent. Il s'agit ici de disposer d'une mesure de comparaison entre deux signaux en adoptant comme critère que des signaux corrélés ne diffèrent que par un terme de retard, Il introduit, dans un premier temps, la fonction d'inter/autocorrélation continue, puis la fonction discrète et d'en dégager les principales propriétés, les différents types et architectures des corrélateurs numériques, dans ce chapitre ; nous nous limitons que pour les signaux déterministes.

I.1)- Historique [4]

La corrélation est un concept issu de la biologie. C'est par le biais des travaux de « FRANCIS GALTON » que la corrélation devient un concept statistique. Toutefois pour Galton, la notion de corrélation n'est pas définie précisément et il l'assimile dans un premier temps à la droite de régression d'un modèle de régression linéaire.

C'est ensuite « KARL PEARSON » qui propose en 1896 une formule mathématique pour la notion de corrélation et un estimateur de cette grandeur. La corrélation est introduite en économie avec l'ouvrage de « BOWLEY » Elements of Statistics en 1902 et l'intervention de « GEORGE UDNY YULE » en 1909. « YULE » introduit notamment la notion de corrélation partielle.L'arrivée de récepteurs numériques dans les années 1980, on assiste à un nouveau regain d'intérêt pour les radars passifs. Pour la première fois les ingénieurs ont pu utiliser des techniques numériques du traitement du signal pour se servir de différents signaux radioélectriques et des techniques de la corrélation pour obtenir des informations suffisamment précises pour détecter des cibles et estimer leur distance et leur décalage.

I.2)- Etude théorique de la corrélation [5] [6]

I.2.1)- Fonctions de corrélation des signaux à énergie finie

I.2.1.1)- Définition intercorrélation

La technique de corrélation est un algorithme de calcul utilisé dans beaucoup de domaines d'application. Vu son importance pratique ainsi que les opérations de calcul qui le constituent.

La fonction de corrélation est l'une des méthodes les plus appréciées pour comparer deux signaux. Son principe se base sur le décalage d'un des signaux par rapport à l'autre et de mesurer, à l'aide d'un produit scalaire, leur similitude en fonction du décalage.

*Soient $x(t)$ et $y(t)$ deux signaux à énergie finie, on appelle fonction d'intercorrélation entre $x(t)$ et $y(t)$ la fonction notée $\varphi_{xy}(\tau)$ définie par :

$$\varphi_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t).y^*(t+\tau)\,dt \qquad (I.1)$$

On peut remarquer que $\varphi_{xy}(\tau)$ est égal au produit scalaire :

$$\varphi_{xy}(\tau) = < x, y_\tau > \qquad (I.2)$$

I.2.1.2)- Définition Autocorrélation

L'autocorrélation est un cas particulier de la fonction d'intercorrélation, puisqu'elle est appliquée sur un seul signal. Elle consiste donc à comparer un signal $x(t)$ avec lui-même, mais décalé de τ .

*Soient $x(t)$ un signal à énergie finie, on appelle fonction d'autocorrélation de $x(t)$ la fonction notée $\varphi_{xx}(\tau)$ définie par :

$$\varphi_{xx}(\tau) = \int_{-\infty}^{+\infty} x(t).x^*(t+\tau)dt \qquad (I.3)$$

I.2.2)- Corrélation des signaux à puissance moyenne finie [7]

Le produit scalaire, défini pour les signaux à énergie finie, n'est pas applicable pour les signaux à puissance moyenne finie, car l'intégrale ne converge pas ! Pour les signaux à puissance moyenne finie, on définit donc la corrélation d'une façon un peut différente :

$$\varphi_{xy}(\tau) = \lim_{T\to+\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t).y^*(t+\tau)\, dt \qquad (I.4)$$

I.2.3)- Fonction de corrélation de signaux périodiques de même période

Pour des signaux périodiques de période T, le calcul de la limite est inutile. Il suffit d'intégrer sur une seule période :

$$\varphi_{xy}(\tau) = \frac{1}{T} \int_{-T/2}^{+T/2} x(t).y(t+\tau)dt \qquad (I.5)$$

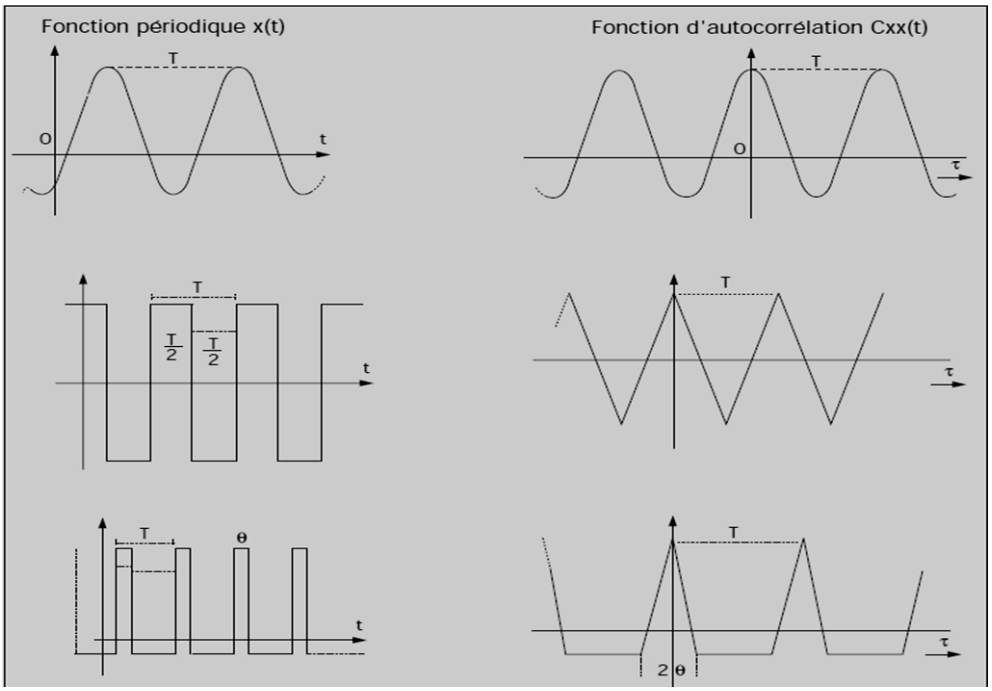


Figure (I.1) : *Exemples de fonctions d'autocorrélation des signaux périodiques*

I.3)- Fonction de corrélation des signaux numériques [8]

I.3.1)- Définitions

Pour des signaux complexes continus x (t) et y (t), la fonction de corrélation $\varphi_{xy}(\tau)$ est définie par :

$$\varphi_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t).y^*(t+\tau)\ dt$$

Le passage au domaine numérique est direct par substitution de l'intégrale par une somme discrète. La fonction de corrélation pour des signaux numériques est donc définie par :

Autocorrélation : corrélation entre signal *x(n)* et lui-même :

$$\varphi_{xx}(\tau) = \sum_{n=-\infty}^{+\infty} x(nT).x^*\big((n+\tau)T\big) \qquad (I.6)$$

Intercorrélation : corrélation entre le signal discret *x(n)* et le signal discret *y(n)* est défini par la relation suivante :

$$\varphi_{xy}(\tau) = \sum_{n=-\infty}^{+\infty} x(nT).y^*((n+\tau)T) \qquad (I.7)$$

De manière similaire, pour des signaux discrets à puissance moyenne finie, on définit l'autocorrélation de la manière suivante :

Autocorrélation : corrélation entre signal *x(n)* et lui-même :

$$\varphi_{xx}(\tau) = \lim_{N\to+\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} x(nT).x^*((n+\tau)T) \qquad (I.8)$$

Intercorrélation : corrélation entre le signal discret *x(n)* et le signal discret *y(n)*

$$\varphi_{xy}(\tau) = \lim_{N\to+\infty} \frac{1}{2N+1} \sum_{n=-\infty}^{+\infty} x(nT).y^*((n+\tau)T) \qquad (I.9)$$

Remarque :

L'auto ou l'intercorrélation sont des signaux temporels et nT est le décalage qui peut intervenir entre les deux signaux comparés. Ce décalage peut être aussi bien positif que négatif et donc les sommes sont bien à priori de $-\infty$ à $+\infty$.

Pour un signal causal qui n'existe donc que pour t compris entre 0 et $+\infty$, son autocorrélation elle existera bien de $-\infty$ à $+\infty$.

Exemple

Considérons comme exemple la fonction de corrélation de deux signaux $x(n)$ et $y(n)$ donnés par : (on prend dans ce cas T=1)

$x(n) = rect_N\ (n)$

$y(n) = |a|^n.u(n)\ \ avec\ \ |a| < 1$

Où $rect_N\ (n)$ est la fonction rectangle échantillonnée de durée N et $u(n)$ est la fonction échelon unité.

Ces signaux sont illustrés sur la figure (I.2). Il faut remarquer d'abord que les valeurs positives de τ dans l'équation correspondent à un décalage de $y\ (n)$ vers la gauche, et que les valeurs négatives de τ correspondent à un décalage de $y\ (n)$ vers la droite. On peut alors distinguer trois cas selon la valeur de τ :

- Pour $-\infty \leq \tau \leq -N$, il n'y a aucun produit non nul $x(n).y(n+\tau)$ donc

  $\varphi_{xy}(\tau) = 0$ Pour $\tau \leq -N$

- Pour $-N \leq \tau \leq 0$ on a

$$\varphi_{xy}(\tau) = \sum_{n=-\tau}^{N-1} a^{\tau+n} = \sum_{m=0}^{N+\tau-1} a^m = \frac{1-a^{N+\tau}}{1-a} \qquad (I.10)$$

En posant $m = n + \tau$.

- Et finalement, pour $\tau > 0$, on obtient

$$\varphi_{xy}(\tau) = \sum_{n=0}^{N-1} a^{\tau+n} = a^\tau \sum_{m=0}^{N-1} a^n = a^\tau \frac{1-a^N}{1-a} \qquad (I.11)$$

La fonction $\varphi_{xy}(\tau)$ complète est représentée dans la figure (I.2).

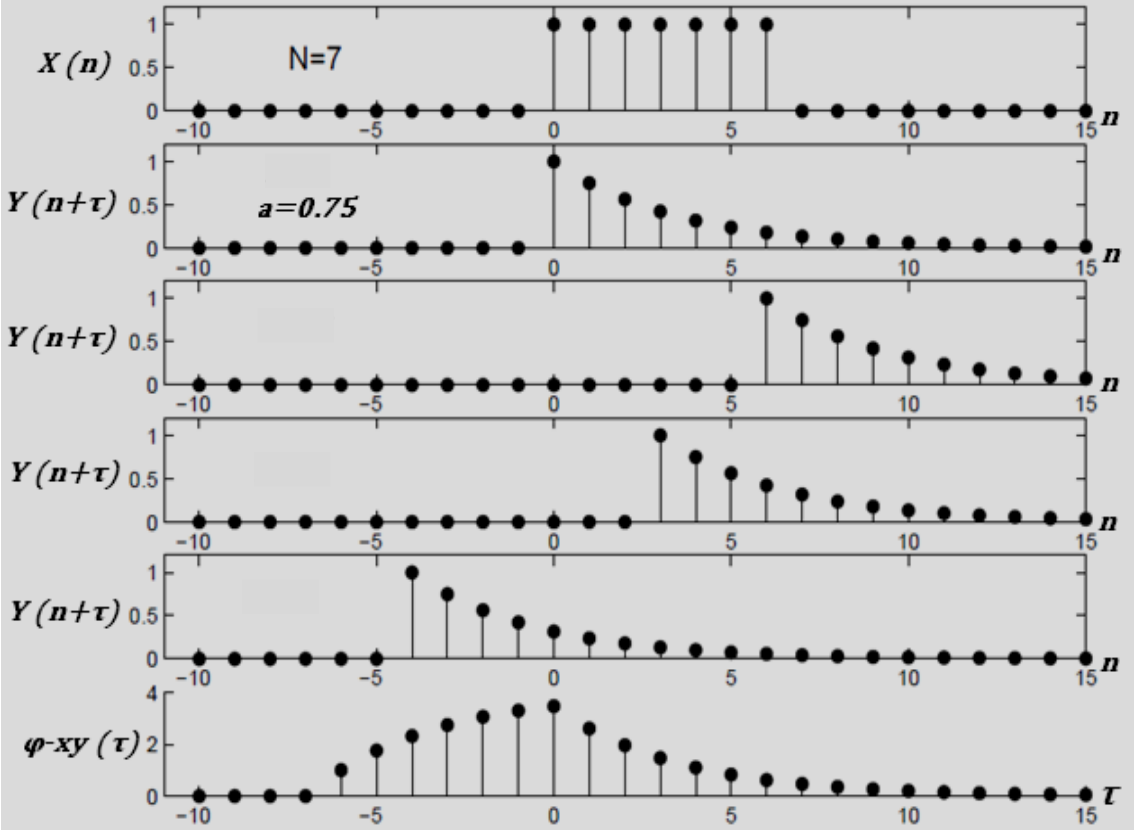


Figure (I.2) : *signaux x(n) et y(n) ainsi la fonction d'intercorrélation* $\varphi_{xy}(\tau)$**[9]**

I.4)- Propriétés [10]

I.4.1)- Parité pour des signaux réels

En utilisant l'égalité $\varphi_{xx}(\tau) = \varphi_{xx}{}^{*}(-\tau)$, et en notant la fonction d'autocorrélation sous la forme $\varphi_{xx}(\tau) = R(\tau) + jI(\tau)$, l'égalité devient :

$$Re(\tau) + jIm(\tau) = Re(-\tau) - jIm(-\tau)$$

Si *x(t)* est un signal réel comme cela est le cas de beaucoup de signaux usuels, sa fonction d'autocorrélation est nécessairement réelle elle est donc paire alors que sa partie imaginaire est impaire, et cette remarque simplifie le calcul de celle-ci grâce à cette symétrie.

$$\varphi_{xx}(\tau) = \varphi_{xx}(-\tau) \quad \text{Et} \quad \varphi_{xy}(\tau) = \varphi_{yx}(-\tau) \qquad (\text{I.12})$$

Pour $\tau = 0$, on a en particulier $\varphi_{xx}(0) = \varphi_{xx}{}^{*}(0)$, donc $\varphi_{xx}(\tau) \in \mathbb{R}$ et vaut :

$$\varphi_{xx}(0) = \sum_{n=-\infty}^{+\infty} x(nT).x^*(nT) = \sum_{n=-\infty}^{+\infty} |x(nT)|^2 > 0 \qquad (I.13)$$

Remarque :

- Pour un signal réel la fonction de corrélation est réelle et paire et sa DSP aussi.

I.4.2)- Valeur maximale

Un signal ressemblera d'autant plus à un autre que cet autre est le même signal non décalé; La fonction d'autocorrélation est donc maximale pour $\tau = 0$.

I.4.3)- Borne supérieure de la fonction de corrélation

Inégalité de Schwartz

Bien connue dans le cas des intégrales elle a aussi son équivalent pour les séries.

Démonstration: Pour tout 'a' réel non négatif, 1a série $|\sum(a_i - \alpha b_i)|^2$ est non négative.

RAPPEL :

$$\left|\sum(a_i - \alpha b_i)\right|^2 \leq \sum|(a_i - \alpha b_i)|^2 \leq \sum|a_i|^2 - 2\alpha\sum|a_i.b_i| + \alpha^2\sum|b_i|^2 = (\alpha - \alpha_1).(\alpha - \alpha_2) \geq 0$$

Le trinôme en α etant non négatif, ses deux racines sont obligatoirement complexes conjuguées : $\alpha_1;\alpha_2 \in \mathbb{C}$

$\alpha_1 = \beta + j\gamma$ et $\alpha_2 = \beta - j\gamma \Rightarrow (\alpha - \alpha_1).(\alpha - \alpha_2) = (\alpha - \beta)^2 + \gamma^2$ toujours positif ou nul.

Pour cela le discriminant doit être négatif ou nul ce qui établie l'inégalité de Schwartz :

$$\left|\sum(a_i b_i)\right|^2 \leq \sum|a_i b_i|^2 \leq \sum|a_i|^2 . \sum|b_i|^2 \qquad (I.14)$$

Application aux fonctions de corrélation :

$$\left|\boldsymbol{\varphi_{xy}(\tau)}\right|^2 = \lim_{N\to+\infty}\frac{1}{(2N+1)^2}\left|\sum_{n=-N}^{+N} x(nT).y^*((n+\tau)T)\right|^2 \leq \lim_{N\to+\infty}\frac{1}{(2N+1)^2}\left[\sum_{n=-N}^{+N}|x(nT)|^2.\sum_{n=-N}^{+N}\left|y^*\left((n+\tau)T\right)\right|^2\right]$$

$$\lim_{N\to+\infty}\frac{1}{(2N+1)^2}\left[\sum_{n=-N}^{+N}|x(nT)|^2.\sum_{n=-N}^{+N}\left|y^*\big((n+\tau)T\big)\right|^2\right]$$

$$=\left[\lim_{N\to+\infty}\frac{1}{(2N+1)^2}\sum_{n=-N}^{+N}|x(nT)|^2\right].\left[\lim_{N\to+\infty}\frac{1}{(2N+1)^2}\sum_{n=-N}^{+N}\left|y^*\big((n+\tau)T\big)\right|^2\right]$$

$$\Rightarrow\left|\varphi_{xy}(\tau)\right|^2\leq\varphi_{xx}(0).\varphi_{yy}(0)\qquad(\text{I.}15)$$

Le carré du module de la fonction d'intercorrélation est borné supérieurement par le produit des puissances moyennes des deux signaux.

Cas d'autocorrélation :

$$\left|\varphi_{xy}(n)\right|\leq\varphi_{xx}(0)\qquad(\text{I.}16)$$

Le module de la fonction de corrélation est borné supérieurement par sa valeur en zéro c'est-à-dire par la puissance moyenne du signal.

I.4.4)- Coefficient de corrélation [12]

D'après la relation (I.15) l'inégalité de shwartz l'intercorrélation normalisée par la moyenne géométrique des autocorrélations est la fonction :

$$C_{corxy}(\tau)=\frac{\varphi_{xy}(\tau)}{\sqrt{\varphi_{xx}(0).\varphi_{yy}(0)}}\qquad(\text{I.}17)$$

Le module de $C_{corxy}(\tau)$est toujours inférieur ou égal à 1 (dans le cas d'une corrélation normalisée). Un module de $C_{corxy}(\tau)$ proche de 1 traduira une bonne identité entre *x* et *y* (à un facteur de retard ou de signe ou d'amplitude près). Pour des petites valeurs, on dira que les signaux ne sont pas corrélés.

On peut définir le temps de corrélation d'un signal, le temps τ tel que :

$$C_{corxy}(\tau)=\frac{\varphi_{xx}(\tau)}{\varphi_{xx}(0)}=\frac{1}{2}\qquad(\text{I.}18)$$

C'est une mesure de la prédictibilité d'un signal.

I.4.5)- Corrélation par Transformée de Fourier [11]

Soient $x(t)$ un signal à énergie finie et $\varphi_{xx}(\tau)$ sa fonction d'autocorrélation, On appelle densité spectrale d'énergie, la fonction notée $S_{xy}(f)$, transformée de Fourier de $\varphi_{xx}(\tau)$:

En utilisant l'expression de l'autocorrélation, on a :

$$\varphi_{xy}(\tau) = \boldsymbol{TF^{-1}}\{TF(x(n) \times TF(y(n)\} \qquad \text{(I. 19)}$$

Demonstration:

$$TF\{\varphi_{xy}(\tau)\} = \sum_{-\infty}^{+\infty} \varphi_{xy}(\tau).exp\,(-2j\pi f\tau) = S_{xy}(f)$$

$$S_{xy}(f) = TF\{x(\tau) * y^*(-\tau)\}$$

$$S_{xy}(f) = TF\{x(\tau)\}.TF\{y^*(-\tau)\}$$

$$\Rightarrow S_{xy}(f) = X(f).Y^*(f) \qquad \text{(I. 20)}$$

Cas d'autocorrélation : $\boldsymbol{S_{xx}(f) = |X(f)|^2}$

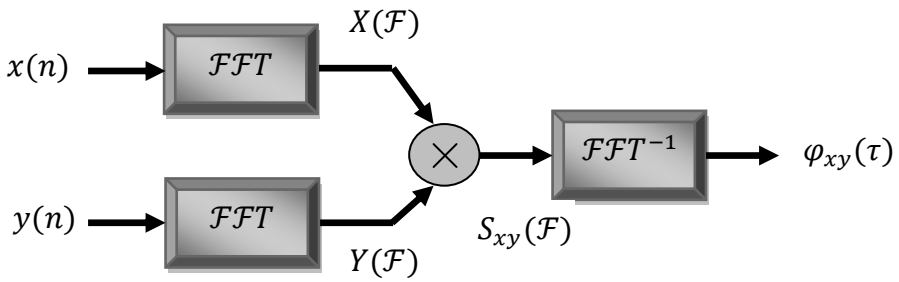


Figure (I.3) : *Opération de corrélation par FFT entre deux signaux*

Remarque :

La transformée de Fourier pour des signaux à énergie finie, sont multipliés dans le domaine fréquentiel.

Pour des signaux à énergie finie, la transformée de Fourier de la fonction de corrélation est égale à la densité spectrale d'énergie.

I.4.6)- Corrélation d'un bruit blanc

L'autocorrélation d'un bruit blanc aura un pic important à $\tau = 0$ et sera proche de 0 pour tout autre τ.

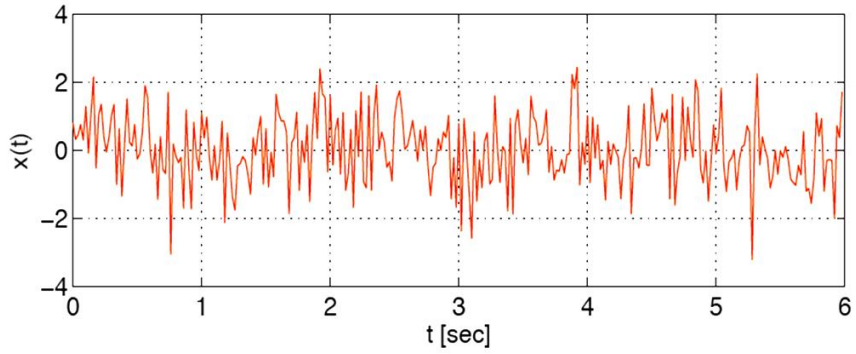


Figure (I.4) : *bruit blanc*

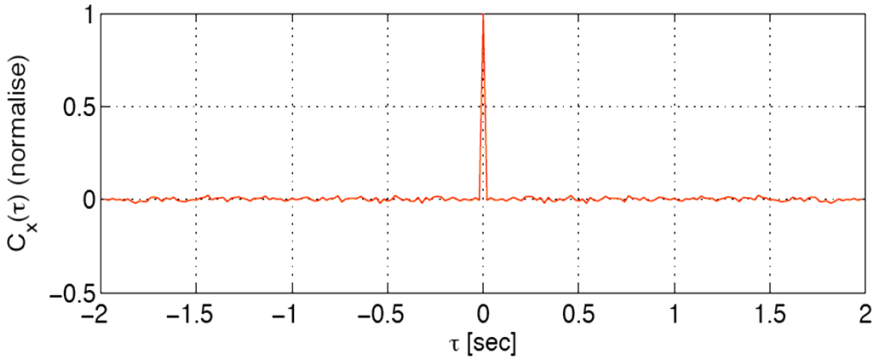


Figure (I.5) : *fonction d'autocorrélation d'un bruit blanc* **[12]**

I.5)-Relation entre corrélation et convolution [13]

Dans la convolution, n reste la variable et une des fonctions est renversée $(y(n-u))$ par rapport à la variable d'intégration u. Dans la corrélation, le décalage τ est la variable de la fonction résultat.

La corrélation mesure la ressemblance entre les fonctions $x(t)$ et $y(t)$ selon le décalage τ, la convolution formalise l'interaction (filtrage) entre les fonctions $x(n)$ et $y(n)$.

Mathématiquement, on peut écrire une relation qui permet d'exprimer la fonction de corrélation comme un produit de convolution (et réciproquement).

En effet,

$$\varphi_{xy}(\tau) = \sum_{-\infty}^{+\infty} x(u).y^*(u-\tau) = \sum_{-\infty}^{+\infty} x(u).y^*(-(\tau-u)) = \; x(\tau) * y^*(-\tau)$$

C'est –à-dire: $\varphi_{xy}(\tau) = \; x(\tau) * y^*(-\tau)$ (I. 21)

I.6)- Erreur quadratique de la fonction de corrélation [14]

La fonction $\varphi\,(\tau)$ est une mesure de la ressemblance entre $x(n)$ et $y\,(n)$ au sens de l'erreur quadratique moyenne $\varepsilon\,(\tau)$. en effet, celle-ci peut s'écrire de la façon suivante :

$$\varepsilon\,(\tau) = \sum_{-\infty}^{+\infty}[x(n) - y(n+\tau)]^2 = \sum_{-\infty}^{+\infty}[x(n)]^2 + \sum_{-\infty}^{+\infty}[y(n+\tau)]^2 - 2\sum_{-\infty}^{+\infty}[x(n).y(n+\tau)]$$

$$\Longrightarrow \varepsilon\,(\tau) = E_x + E_y - 2\varphi\,(\tau) \qquad (I.22)$$

Où E_x et E_y sont les énergies des signaux $x(n)$ et $y(n)$.

On voit donc que $\varepsilon\,(\tau)$ est une fonction linéaire de $\varphi\,(\tau)$, en particulier si les deux signaux sont identiques à un décalage temporel τ_0 près, la fonction $\varepsilon\,(\tau)$ s'annule pour $\tau = \tau_0$ et donc $\varphi\,(\tau)$ passe par un maximum pour cette valeur.

Le calcul $\varphi\,(\tau)$ peut s'effectuer en passant par l'intermédiaire des spectres de $x(n)$ et $y\,(n)$

I.7)- Dérivation de la fonction de corrélation

A partir du résultat :

$$\varphi_{xy}(\tau) \rightleftharpoons X(f).Y^*(f) \qquad (I.23)$$

Et en utilisant les propriétés de dérivation de la TF par rapport à la variable temporelle, on a :

$$\frac{d\varphi_{xy}(\tau)}{d\tau} \rightleftharpoons X(f).Y^*(f)(j2\pi f). \qquad (I.24)$$

En associant le terme $(j2\pi f)$ à l'une ou l'autre des TF, on peut donc écrire :

$$\frac{d\varphi_{xy}(\tau)}{d\tau} = \frac{dx(\tau)}{d\tau} * y^*(-\tau) = \varphi_{x'y}(\tau) \qquad (I.25)$$

Ou

$$\frac{d\varphi_{xy}(\tau)}{d\tau} = x(\tau) * \left(\frac{dy^*(-\tau)}{d\tau}\right) = x(\tau) * \left(-\frac{dy^*(-\tau)}{d(-\tau)}\frac{d(-\tau)}{d\tau}\right)$$

$$= x(\tau) * y'^*(-\tau) = -\varphi_{xy'}(\tau) \qquad (I.26)$$

$$\varphi_{xy}{}'(\tau) = \varphi_{x'y}(\tau) = -\varphi_{xy'}(\tau) \qquad (I.27)$$

On peut généraliser ces deux expressions à des dérivées d'ordre quelconque.

I.8)- Limite de la fonction de corrélation [11]

Le fait que le signal soit à puissance moyenne finie implique que la transformée de Fourier de la fonction de corrélation, qui est la DSP du signal, existe. Pour que cette transformée existe il faut que la fonction de corrélation soit absolument sommable :

$$\lim_{N\to+\infty} \sum_{n=-N}^{+N} \left|\varphi_{xy}(n)\right| \leq \infty \qquad (I.28)$$

Cette condition implique deux contraintes :

- La fonction de corrélation doit être bornée. (Ceci a été montré dans la partie précédente).
- Les limites de la fonction de corrélation ne peuvent être que nulles. Ceci nous amène à la seconde propriété des fonctions de corrélation : $\lim_{N\to+\infty} \boldsymbol{\varphi_{xy}(n) = 0}$

La seconde propriété peut être légèrement étendue car elle est induite par la condition d'existence d'une transformée de Fourier au sens des fonctions. Si on étend ceci aux distributions on peut ajouter à la fonction de corrélation une constante.

I.9)- Principe de la corrélation numérique [15]

L'opération de corrélation est un élément très important dans le traitement des signaux. Elle permet en particulier de déterminer le degré de similitude entre deux signaux. En général, un des deux signaux est connu et l'autre non. Cependant, la corrélation peut également être utilisée pour la recherche de périodicité dans un signal.

Le principe est de faire le produit moyenné des signaux pour différents décalages temporels.

$$\varphi_{xy}(\tau) = \sum_{-\infty}^{+\infty} x(n).y(n+\tau)$$

Pour un décalage τ donné, chaque échantillon des deux signaux sont multipliés deux à deux. Le résultat de la corrélation est la somme des résultats du produit des échantillons. Dans le cas d'un signal inconnu, il est nécessaire de faire la corrélation pour tous les décalages possibles afin de pouvoir déterminer dans quelle position le signal est le plus proche du signal connu. La corrélation la plus élevée correspond à la meilleure similitude.

La formule de la corrélation a l'avantage d'être relativement modulable. La plupart du temps, une corrélation normalisée est utilisée, c'est-à-dire que le résultat obtenu varie entre 0 (pour le cas où il n'y a aucune dépendance entre les deux signaux) et 1 (pour le cas où les deux signaux sont complètement similaires).

Elle peut également être utilisée pour différentes sortes de signaux. Ceux-ci peuvent être autant réels que complexe. La plupart du temps, les signaux sont des signaux réels échantillonnés. La corrélation est également utilisable pour les signaux numériques.

I.10)- Corrélation binaire

Dans le cas de signaux binaires, chaque échantillon vaut zéro ou un. Il est possible, dans le cas où un signal serait échantillonné à fréquence élevée, de considérer qu'il n'y a pas plus d'un changement de valeur par cycle.

Donc le produit de deux échantillons binaires donnerait dans ce cas un résultat de zéro ou un. Il est possible alors de réaliser simplement le produit de deux signaux binaires à l'aide d'une porte AND. La somme consiste également en une opération relativement simple, car il suffit de compter le nombre total de un (1) dans un signal de plusieurs bits.

I.11)- Différentes types des corrélateurs [17]

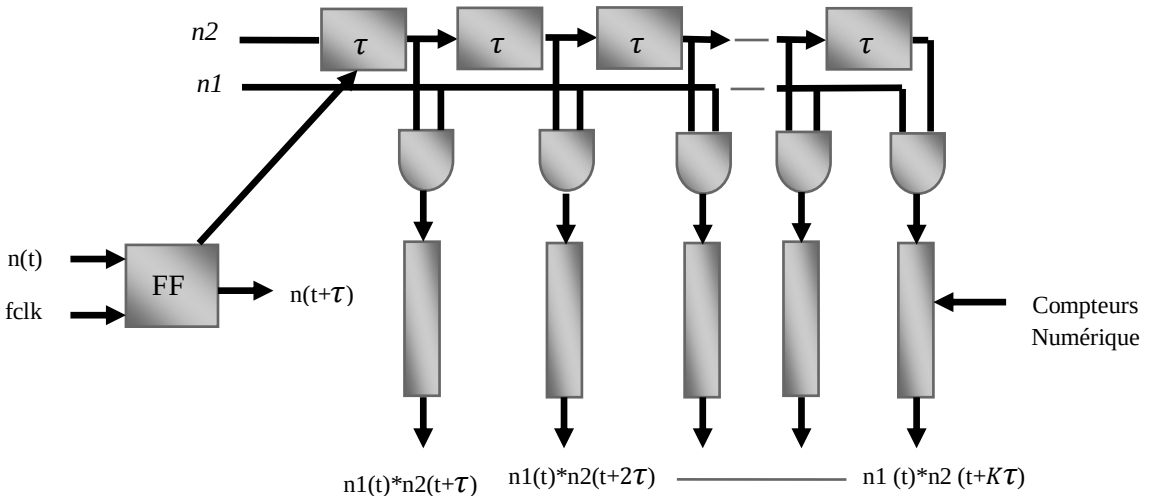


Figure (I.6) : *Architecture standard d'un corrélateur*

I.11.1)- Corrélateur linéaire (architecture standard)

Le corrélateur linéaire à l'architecture la plus simple. Il possède deux entrées pour les signaux. L'une d'elles possède une série de retard afin de créer le décalage entre les deux signaux (voir figure I.6). Ce signal est ensuite multiplié à chaque décalage à l'aide d'une simple porte AND avec le signal non retardé.

Cela permet d'obtenir alors les produits utiles pour la corrélation. En effet, la première porte AND aura à ses entrées les signaux avec un décalage de τ entre eux, la seconde aura un décalage de ($2.\tau$), La suivante ($3.\tau$) et ainsi de suite. La somme des produits se fait alors par un simple compteur qui, à chaque coup d'horloge, s'incrémente ou non en fonction de la sortie de la porte AND.

Une fois que les signaux sont complètement passés, on obtient en sortie des compteurs les résultats des corrélations pour chaque décalage, de n à ($n\times \tau$) où n est le nombre de canal et de bit de chaque signal.

Les deux principaux paramètres d'un tel corrélateur sont la fréquence d'horloge f_{clk} et la plage dynamique n. Cependant, le nombre de compteur linéaire correspond au nombre de bit. Il est donc difficile de réaliser des corrélateurs ayant un nombre de bit très élevé.

I.11.2)- Corrélateur exponentiel

Il existe alors une possibilité d'augmenter la plage dynamique de résolution en utilisant des intervalles d'échantillonnage avec une durée allant en s'accroissant. C'est le concept du corrélateur exponentiel. Pour chaque canal, la durée de l'horloge est augmentée légèrement (c'est-à-dire la fréquence est diminuée). Cependant cette méthode est relativement complexe à implémenter et les résultats obtenus n'ont pas un degré de résolution exacte.

I.11.3)- Corrélateur multiple-tau

Le corrélateur multiple-tau est une combinaison utilisant les avantages d'une architecture linéaire et d'une architecture exponentielle. Il consiste à mettre en cascade une succession de corrélateurs linéaires.

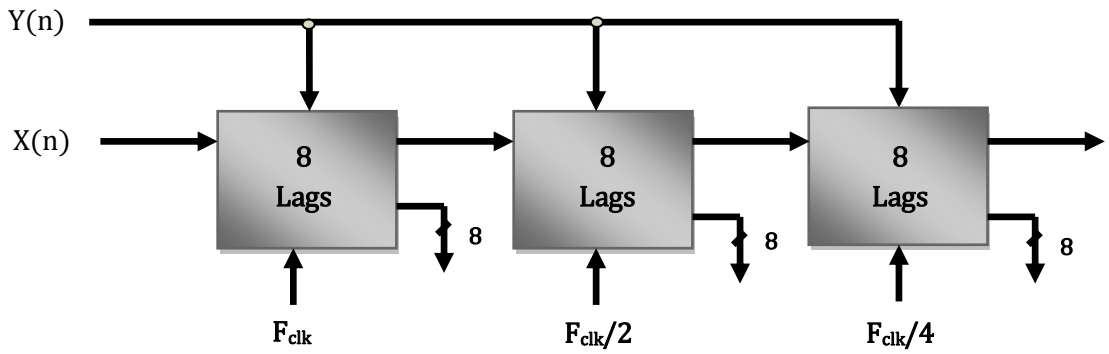


Figure (I.7): *Corrélateur multiple tau*

Typiquement, plusieurs canaux sont regroupés ensemble pour une même fréquence et celle-ci est divisée par deux pour le bloc de canaux suivant. Cette architecture a pour conséquence une diminution de l'erreur engendrée par la structure exponentielle de l'horloge.

I.12)- Algorithme de la fonction intercorrélation

L'algorithme mis en jeu ici peut être détaillé en trois étapes :

- Le signal $y(nT)$ est décalé d'une certaine quantité τ ;
- Le produit $x(nT)$ et $y^*((n+\tau)T)$ est effectué échantillon par échantillon pour toute les valeurs de τ ;
- Les valeurs obtenues sont additionnées pour obtenir la valeur de $\varphi_{xy}(\tau)$

Ces étapes sont répétées autant de fois que nécessaire (figure I.10).

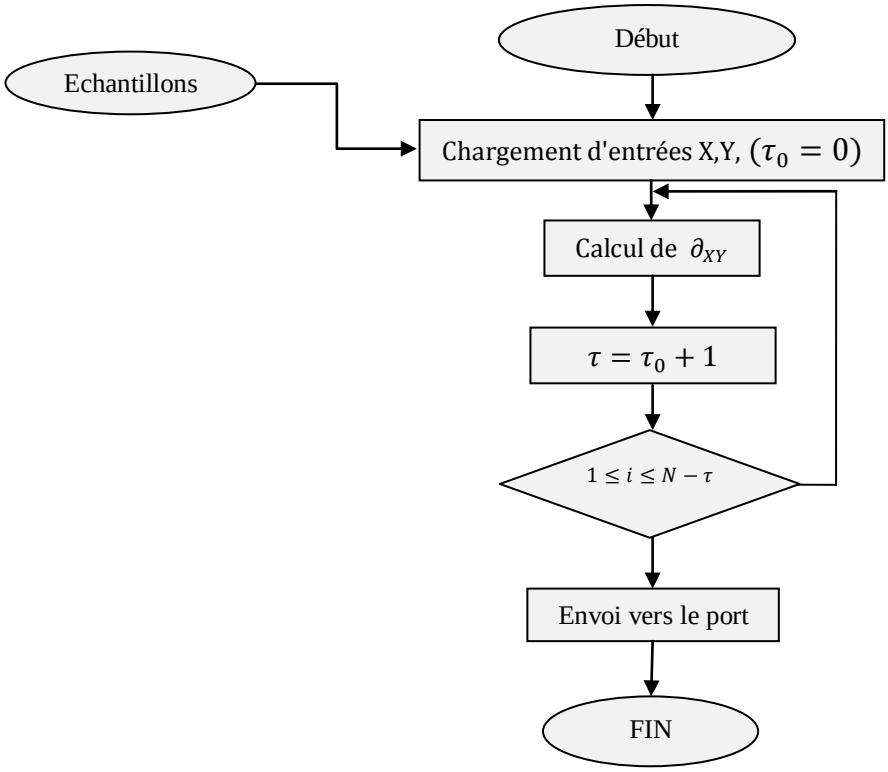


Figure (I.08) : *algorithme de la fonction de corrélation*

I.13)- Domaine d'utilisation des corrélateurs [17]

Les domaines où les corrélateurs sont utiles sont multiples. Un de ces domaines est le domaine des **télécommunications**.

Les corrélateurs sont utilisés ici surtout pour faire de la synchronisation ou bien de la correction d'erreur. Le meilleur exemple d'appareil utilisant des corrélateurs est le GPS (Global Positioning System). Le corrélateur permet de déterminer précisément le moment de synchronisation entre un signal reçu (provenant d'un satellite) et le signal attendu (généré par le GPS).

- L'astronomie est aussi un domaine d'utilisation des corrélateurs. Ils sont utilisés pour des mesures d'interférométrie ou encore pour la spectrométrie.
- La recherche sur les biotechnologies utilise aussi des corrélateurs. Ils sont utiles pour des mesures de spectrométrie sur l'ADN.
- En optique, l'autocorrélation normalisée et la corrélation croisée donnent le degré de cohérence d'un champ électromagnétique

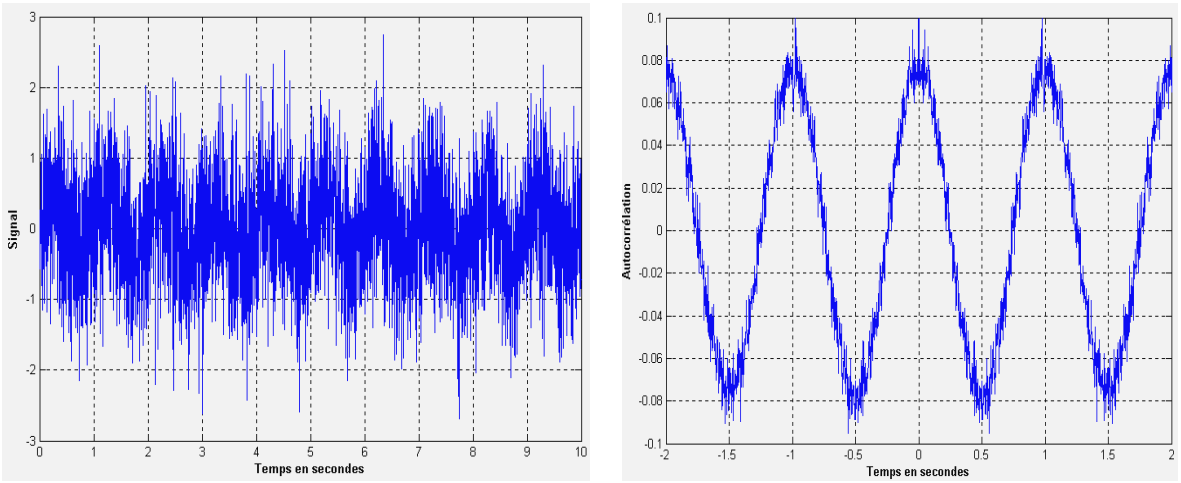
I.14)- Quelques applications [11]

Les applications de la corrélation sont nombreuses en filtrage et en identification.
Elles impliquent souvent une approche stochastique et la modélisation des signaux aléatoires. Nous nous contenterons de décrire ci-dessous les deux applications élémentaires de l'autocorrélation et de l'intercorrélation dont le but est d'extraire un signal du bruit.

I.14.1)- Extraction d'un signal du bruit par autocorrélation

Nous faisons l'acquisition d'un signal qui a été bruité lors de sa transmission par exemple. Le bruit est souvent à caractère aléatoire et la seule hypothèse que l'on est amené à formuler est que son spectre de fréquence est très étendu et ne peut donc être supprimé par un filtrage classique réjecteur de bande.
Si, dans l'espace des fréquences, le spectre de puissance du bruit est étendu, sa transformée de Fourier inverse sera très localisée dans le domaine temporel.

On pourra donc séparer la partie corrélation du bruit de celle du signal si l'auto-corrélation du signal est étendue. Les signaux périodiques bruités sont de bons candidats pour ce type de traitement.



(a) **(b)**

Figure (I.09) :*(a) :* *sinusoïde noyée dans un bruit blanc ;**(b)** autocorrélation du signal*

I.14.2)- Mesure d'un temps par intercorrélation

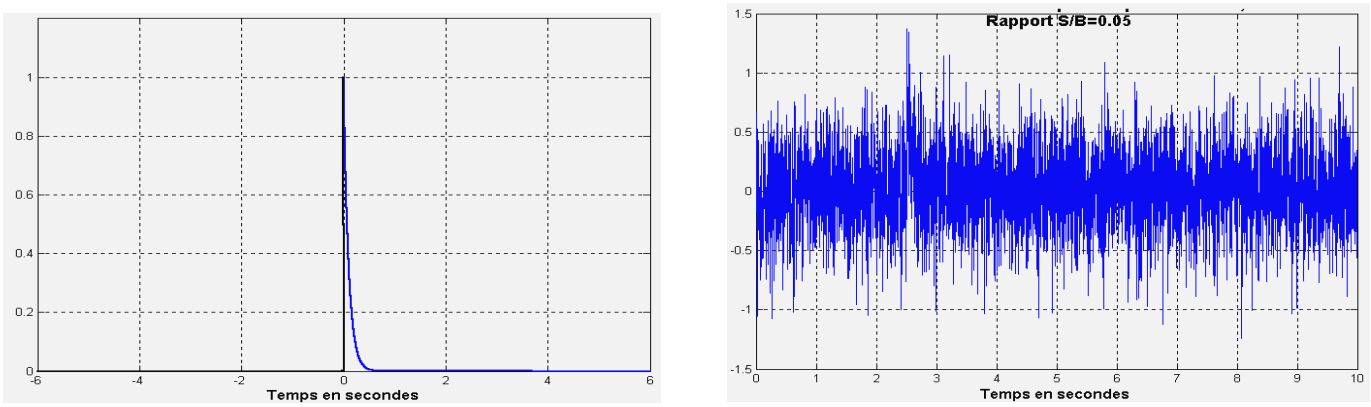
Un signal $x(t)$ est envoyé vers une cible et un écho $y(t)$ revient vers l'émetteur. Le temps de parcours θ du signal permet de déterminer la distance à laquelle se situe la cible mais l'écho de retour est en général un signal fortement bruité ce qui ne permet pas une mesure directe du temps de parcours. Cependant nous disposons du signal émis qui est indépendant du bruit qui s'ajoute lors du parcours et l'intercorrélation permettra de minimisé l'effet de ce bruit.

$$\boldsymbol{y(t) = x(t-\theta) + b(t)} \qquad (\text{I.31})$$

$$\boldsymbol{\varphi_{yx} = \varphi_{x(t-\theta)x(t)} + \varphi_{b(t)}} \qquad (\text{I.32})$$

$$\boldsymbol{\varphi_{b(t)} \simeq 0 \Rightarrow \varphi_{yx} \simeq \varphi_{x(t-\theta)x(t)}} \qquad (\text{I.33})$$

Si le signal $x(t)$ est choisi tel que sa fonction d'autocorrélation φ_{xx} soit maximale en $t = 0$, la fonction d'intercorrélation φ_{xy} sera maximale en $t.T = \theta$, l'effet du bruit devenant quasiment négligeable.



(a) ***(b)***

Figure (I.10) : ***(a)*** *Signal émis* ***;(b)*** *signal reçu bruit après un parcours de 2.5s*

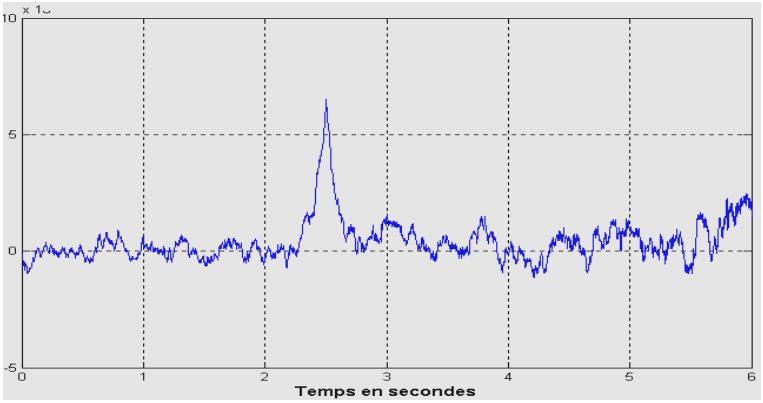


Figure (I.11) : *intercorrélation entre le signal reçu et le signal émis*

I.15)- Conclusion

La fonction de corrélation se trouve à la jonction de diverses théories mathématiques. Elle offre la possibilité de mesurer le degré de ressemblance entre les signaux, nous avons présenté un aperçu rapide sur les différents types des corrélateurs numériques ainsi que leurs avantages et leurs applications.

Il est à noter que le chapitre suivant sera la conception des circuits numériques.

CHAPITRE II

CONCEPTION DES CIRCUITS NUMERIQUES

Objectif

L'objectif de ce chapitre, avant d'aborder le vif du sujet, est de présenter les circuits FPGA (**F**ield-**P**rogrammable **G**ate **A**rray) afin de se familiariser avec ce type de circuits ainsi la méthodologie de conception et la programmation de ces circuits et en fin, nous présentons l'architecture générale du composant ALTERA TYPE CYCLONE III en détail ainsi la carte STARTER KIT-CYCLONE III.

II.1)- Introduction

Au cours des quinze dernières années, les méthodes de conception des fonctions numériques ont subi une évolution importante. Dans les années soixante-dix, la majorité des applications de la logique câblée étaient construites autour de circuits intégrés standard, souvent pris dans la famille TTL. Au début des années quatre-vingt apparurent, parallèlement, les premiers circuits programmables par l'utilisateur du côté des circuits simples et les circuits intégrés spécifiques (ASICs) pour les fonctions complexes fabriquées en grande série. La complexité de ces derniers a nécessité la création d'outils logiciels de haut niveau qui sont à la description structurelle (schémas au niveau des portes élémentaires) ce que les langages évolués sont au langage machine dans le domaine de la programmation. A l'heure actuelle, l'écart de complexité entre circuits programmables et ASICs s'est restreint : on trouve une gamme continue de circuits qui vont des héritiers des premiers PALs (programmable array logic), équivalents de quelques centaines de portes. **[18]**

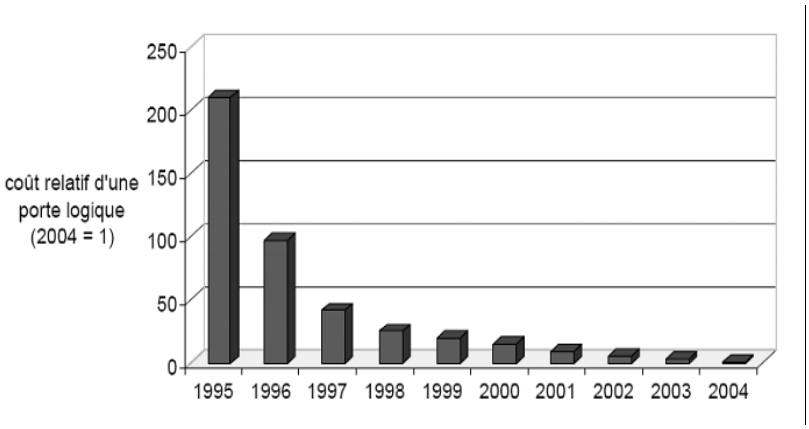
Aujourd'hui, l'avènement des dernières générations d'FPGAs à permis de mettre la technologie SOC à la portée d'un public nettement plus large. Ceci est particulièrement depuis que les FPGA sont proposés à un prix très faible et raisonnable. Ce prodigieux essor à été rendu possible grâce aux progrès concernant les technologies de fabrication des transistors et les méthodes de conception assistée par ordinateur (CAO). Le rôle des FPGA est d'intégrer des circuits logiques complexes. Ces circuits sont susceptibles d'être reconfigurés (Architecture programmée modifiable) partiellement ou entièrement suivant l'application.

Les outils d'aide à la conception se sont unifiés ; un même langage, VHDL par exemple, peut être employé quels que soient les circuits utilisés PALs, ASICs……ect. **[19]**

II.2)- Facteurs d'évolution des circuits Numériques [20]

Les circuiteries numériques (où la porte logique représente l'unité de base) reposent sur les trois aspects des circuits logiques qui sont : le combinatoire, le séquentiel et l'hybride des deux. Des statistiques qui ont été faites ces dernières années ont montré qu'au bout de 10 ans, le prix d'une porte logique a été divisé par 200 où l'intérêt économique du numérique.

A cet effet, le prix d'une porte logique se réduit de 40% par an. Le diagramme suivant montre ces statistiques d'évolution des coûts durant ces dernières années:



Figure(II.01) : *Diagramme d'évolution des coûts ces dernières années*

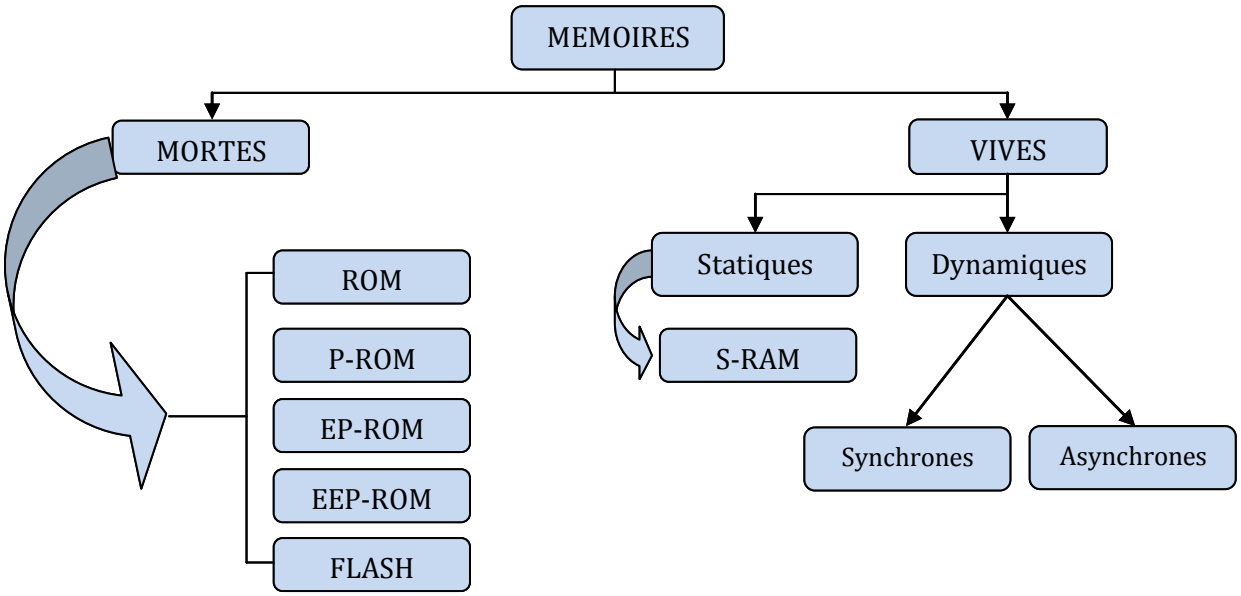
Cette évolution est le fruit d'un ensemble de facteurs qui sont récapitulés dans le tableau suivant :

Besoins croissants en circuits spécialisés	Evolution rapide de la Technologie	Evolution des outils de conception "haut-niveau"
✚ Produits de plus en plus complexes. ✚ Contraintes-(performance, coût…).	✚ Généralisation des « Systèmes sur Puce ». ✚ Espace de conception trop grand	✚ Produire une architecture à partir d'un algorithme. ✚ Exploration automatique de l'espace de conception. ✚ Outils d'estimations (performances, surface, etc.).

Tableau(II.01): Ensemble de facteurs d'évolution des circuits numériques.

II.3)- Les technologies de Mémorisation [21]

Voici d'une manière générale, l'arborescence des mémoires disponibles à nos jours :



Figure(II.02) : *Les différents types de mémoires*

L'ensemble des caractéristiques de ces mémoires sont récapitulées comme suit :

- **Les ROM (R**ead **O**nly **M**emory**):** Mémoires figées par le concepteur à lecture seule et non modifiables.
- **Les P-ROM (P**rogrammable **R**ead **O**nly **M**emory**):** Mémoires programmables une fois par l'utilisateur avec un équipement spécialiser (tableau de fusibles).
- **Les EP-ROM (E**rasable **P**rogrammable **R**ead **O**nly **M**emory**):** Mémoires programmables électriquement et effacement par des rayons ultra-violets au bout d'un certain temps (Quelques minutes).
- **Les EEP-ROM (E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory**):** Mémoires programmables électriquement à lecture seule, effaçables électriquement (Quelques milli-secondes).
- **Les mémoires FLASH:** Elles sont une version plus évolué des EEP-ROM avec avantage d'être plus facile à programmer et à effacer.

- **Les S-RAM (S**tatic **R**andom **M**émory**):**Mémoires volatiles avec cellule de base à plusieurs transistors (accès rapide, consommation plus, coûteux). La volatilité correspond au non disponibilité de l'information lorsqu'il n'y a pas d'alimentation.
- **Les RAM dynamiques:** Mémoires volatiles qui nécessitent rafraichissement périodique de l'information afin de la conserver avec cellule de base à un transistor (densité forte, accès lent).

II.4)- Les circuits Programmables [22]

Le principe de la logique programmable remonte au début des années 1960, le concept ayant été proposé par « G. Estrin ». Il a cependant fallu attendre les années 1980 pour que les premières réalisations matérielles apparaissent sur le marché. L'apparition de ce type de circuit s'est d'abord faite au travers de circuits logiques programmables simples de type PAL (Programmable Array Logic), qui se programment comme des mémoires non volatiles de type ROM et sont utilisés pour implémenter des fonctions combinatoires simples, telles des décodeurs d'adresse, ou des contrôleurs de bus.
Avec les évolutions en micro-électronique, différentes familles de circuits programmables ont commencé à apparaître : Les CPLD (Complex Logic Programmable Device), puis les FPGA (Field Programmable Gate Arrays), introduits par la société Xilinx en 1985. L'industrialisation de ce type de circuits s'est faite à grande échelle avec l'apparition de circuits de plus en plus performants et reprogrammables à volonté. Les circuits de type FPGA les plus récents offrent désormais l'équivalent de dix millions de portes logiques programmables, à des fréquences de fonctionnement atteignant les 200MHz. À l'heure actuelle, on compte une dizaine de fabricants, le marché étant nettement dominé par les sociétés Xilinx, Altera (circuits reprogrammables) et Actel (circuits non reprogrammables).

D'une manière générale, il existe deux alternatives ou solutions qui sont:

- *Une solution logicielle*: Elle est nommée aussi solutions programmables du type « Processeur » où un traitement séquentiel relativement lent et programmation dépendante du composants (DSP, Microprocesseur et Microcontrôleur…).

- *Une solution matérielle*: Elle est nommée aussi solutions programmables du type «Logique» où un traitement parallèle en temps réel et une programmation architecturale avec un langage de description matériel HDL (Méthodologie de conception CAO) indépendante du composant (ASIC et FPGA…).

Les caractéristiques de ces circuits sont :

1. *Les circuits du type DSP/Microprocesseurs* : Un rapport performance/coût faible, un temps de conception très court et une grande souplesse d'utilisation.

2. *Les circuits du type spécialisé ASIC* : Très performants mais avec un cycle de conception long et une architecture figée.

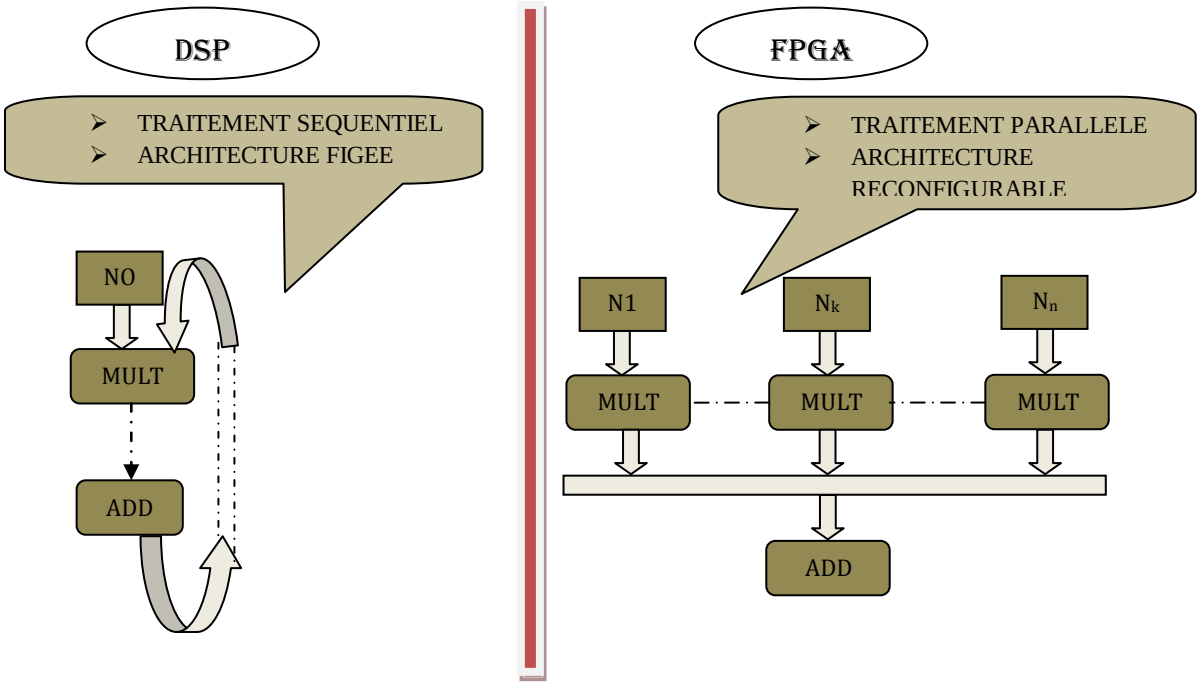
3. *Les circuits du type FPGA :* Des performances proches des ASIC**,** un coût unitaire intermédiaire et un cycle de conception moyen et une architecture modifiable. Voici un tableau récapitulatif de comparaison des différentes solutions numériques :

	ASIC	**FPGA**	**DSP**
Performance	Très élevées	élevées	faibles
Taille	faibles	moyenne	élevées
Consommation	faibles	moyenne	Très élevées
Intégration	Système sur puce	Système sur puce	Composants supplémentaires
souplesse	Fonctions figées	reconfigurable	Programmable
Mise en œuvre	Complexe	Complexité moyenne	Complexité moyenne
Coût du composant	Très élevées	Moyen	faibles

Tableau(II.01): *Comparaison des différentes solutions numériques*

Le tableau présente les avantages et inconvénients respectifs des architectures les plus couramment utilisées pour le traitement numérique du signal. Les circuits FPGA apparaissent comme un bon compromis, tant du point de vue de leur prix de revient limité que de leurs performances leur permettant d'être intégrés dans les architectures de traitement du signal à haut-débit.

Les fréquences de fonctionnement du mode séquentiel dépassent aujourd'hui 2 GHz, la réduction du temps de cycle ne suffira pas à compenser les insuffisances de ce mode de fonctionnement. La fréquence d'horloge ou de fonctionnement des FPGA est relativement faible devant les microprocesseurs et les DSP et ne dépasse pas quelques centaines de Mégahertz, mais cette faiblesse est largement compensée et même surpassée grâce au parallélisme de traitement. L'exploitation du parallélisme est une technique en pleine expansion dans les circuits numériques FPGA qui sont une alternative des DSP.

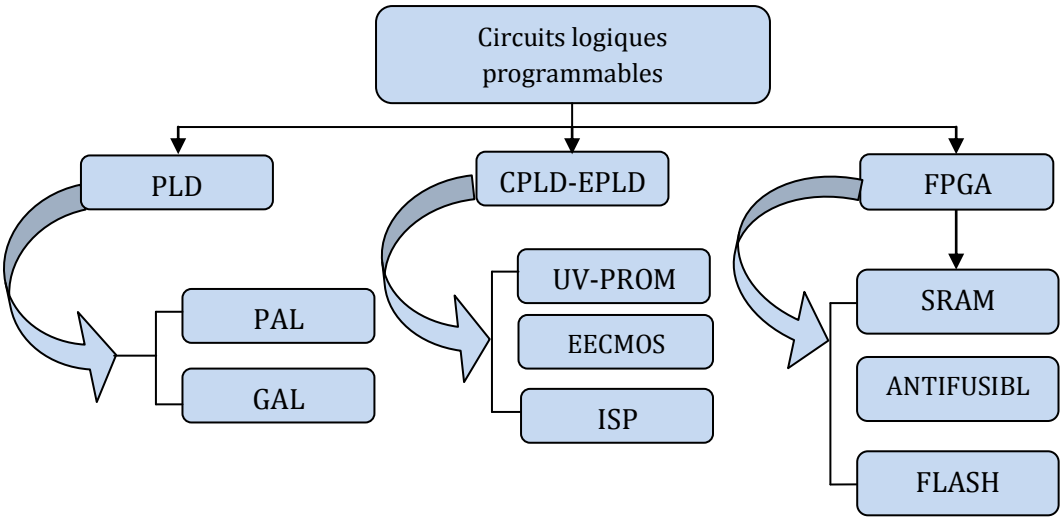


Figure(II.03): *Schéma comparatif d'un **DSP** et d'un **FPGA*** **[23]**

Les circuits séquentiels ont une architecture matérielle figée et inversement pour les circuits logiques reconfigurables FPGA, on fait une adaptation de l'architecture du composant en fonction des l'algorithme

II.4.1)- Les circuits Logiques Programmables [21]

Alors qu'auparavant la distinction était nette entre le logiciel et le matériel, le circuit logique programmable FPGA est venu s'introduire comme un hybride entre les deux approches. Les circuits logiques programmables et reprogrammables architecturalement sont classifiés en trois grandes familles les PLD, CPLD et FPGA. La figure suivante illustre les différents types suivant la technologie utilisée.



Figure(II.04) : *Diagramme des différents types de circuits logiques programmables*

- *Les PLD (Programmable Logic Device)* : Famille des circuits programmables qui comprend les PAL, GAL.
- *PAL (Programmable Array Logic)* : Circuits logiques programmables dans lesquels seules les fonctions ET sont programmables, les fonctions OU ne le sont pas.
- *GAL (Generic Array Logic)* : Circuits logiques PAL reprogrammables à technologie CMOS.
- *Les CPLD ou EPLD (Erasable Programmable Logic Device)* : Circuits logiques reprogrammables.
- *ISP (In System Programmable)* : Circuit que l'on peut programmer même lorsqu'il est en place sur l'application.
- *Les FPGA (Field Programmable Gate Array)* : Ces circuits sont une évolution des CPLD. Récemment, ils intègrent également des mémoires entières, des multiplieurs et même des noyaux de processeur.

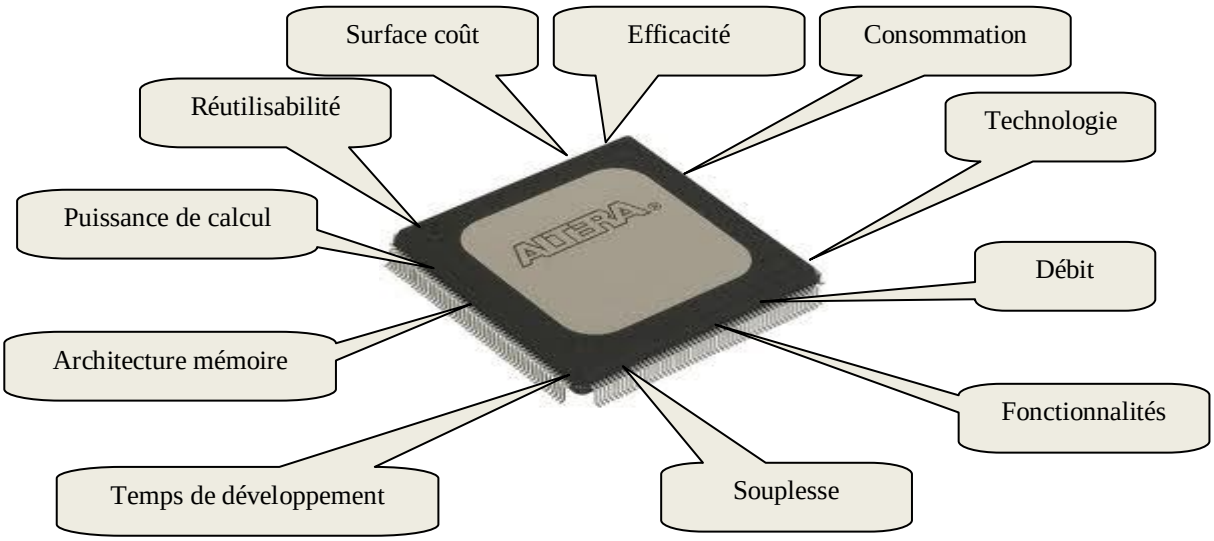
II.4.1.1)- Les Circuits Logiques Programmables du type FPGA

II.4.2)- Critères de choix du circuit programmable FPGA [24]

Les FPGA sont développés récemment grâce aux progrès de la technologie VLSI, l'apparition de ce type de circuits est une révolution des systèmes digitaux et ouvrants des perspectives de traitement numérique inaccessibles auparavant. La fin des années 80 a vu l'apparition des premiers circuits FPGA qui sont des circuits intégrés que l'on peut configurer en un temps relativement court pour réaliser n'importe quelle fonction logique « câblée » à bas coût par une programmation de ses cellules logiques et ses interconnexions avec une restriction de ne pas épuiser les ressources du FPGA. Typiquement, un circuit FPGA haute densité peut contenir jusqu'à plusieurs millions d'éléments programmables. Pour réussir une application à base de FPGA et afin d'obtenir un système plus performant, consommant un minimum de puissance, il est nécessaire de respecter un certain nombre de règles comme :

- Bien connaître les caractéristiques du FPGA ciblé pour assurer son adéquation avec les besoins du projet.
- Elaborer une méthodologie de conception.
- Maîtriser les outils d'implémentation et de choisir des outils de synthèse de qualité.

La conception sur les circuits FPGA est un challenge dans lequel l'objectif est de trouver le bon compromis entre densité, flexibilité et performances temporelles.



Figure(II.05) : *Critères de choix du circuit logique programmable **FPGA***

II.4.3)- Différent domaines d'application des FPGA

Les FPGA on fait révolutionner certain domaines de contrôle numérique et de plus en plus utilisés pour intégrer des architectures numériques complexes. Ils sont devenus les plus populaires en matière d'implantation et de prototypage des circuits numériques après leur apparition sur le marché en 1984**.** La clé maîtresse de leurs réussites est l'aspect de programmation de ces derniers. Leurs utilisations actuelles couvrent les deux domaines : civil et militaire. Parmi ces applications nous citons :

1- Informatique : Périphériques spécialisés.

2- Machinerie industrielle : Contrôleur pour machines.

3- Télécommunications : Traitement d'images, Filtrage.

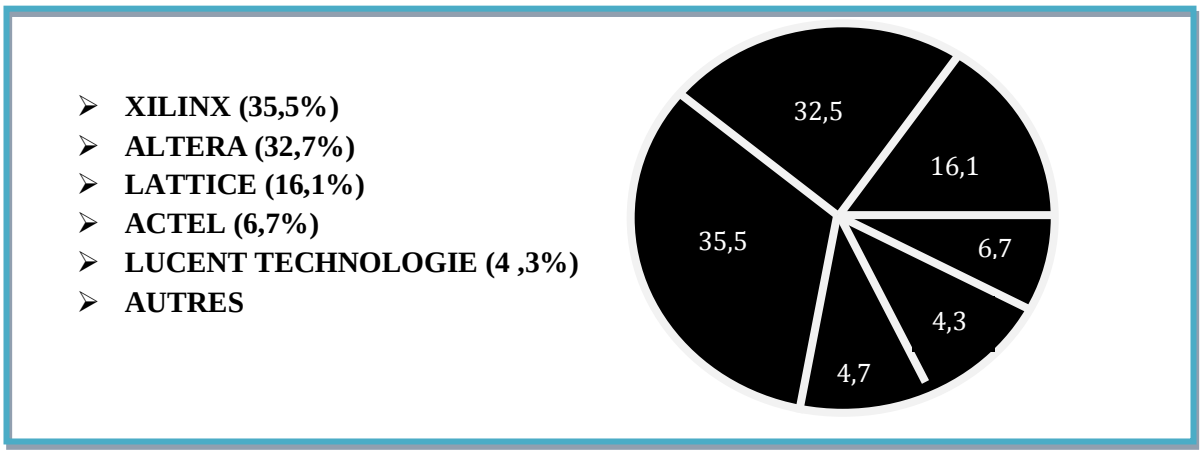
4- Instrumentation : Équipement médical, Prototypage.

5- Transport : Contrôle d'avions et métros.

6- Aérospatiale: Satellites, Radar, Communication protégée, la détection ou la surveillance……etc.

II.4.4)-Principaux fondeurs d'FPGA

Les fabricants des FPGA ne cessent pas d'améliorer leurs produits par l'efficacité et la puissance. L'ensemble des firmes (Principaux fondeurs) qui conçoivent ce type de circuits sont : **Actel, Altera, Atmel, Cypress, Lattice, Minc, QuicLogic, Xilinx** et d'autres.

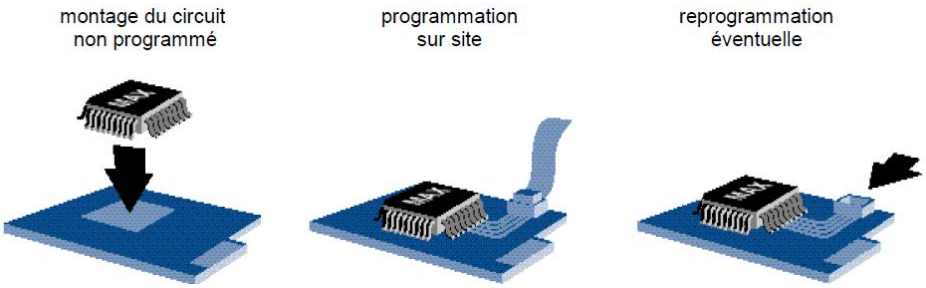


Figure(II.06) : *Statistiques du marché occupé par les vendeurs d'**FPGA*** **[25]**

II.4.5)- Configuration et reconfiguration des FPGA [26]

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 système reconfigurable 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 reconfigurabilité est une propriété nécessaire face aux systèmes à charges et contraintes variables. Le FPGA est une abréviation anglaise qui signifie (réseau des portes programmables) ce qui est décrit dans la figure suivante.



Figure(II.07): *Reprogrammabilité sur site d'un **FPGA***

II.4.6)- Technologies de programmation des FPGA [27]

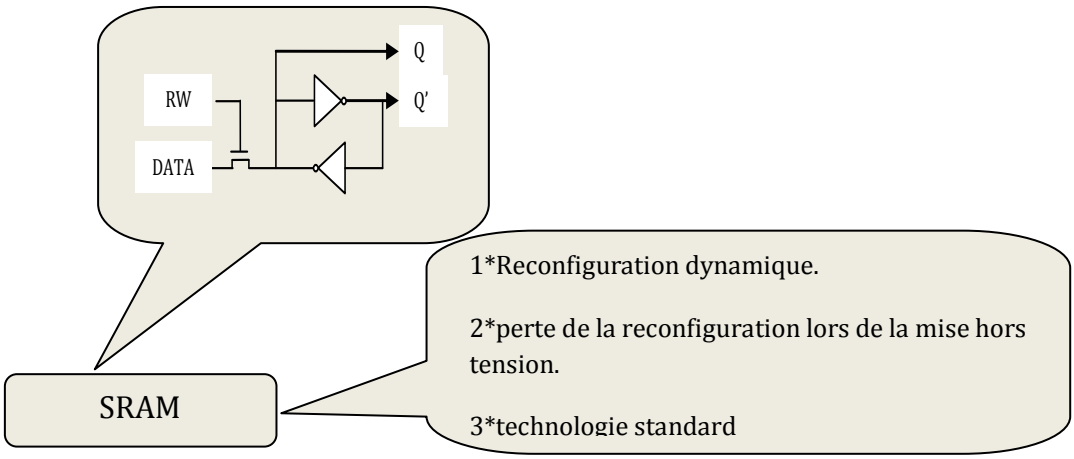
Pour franchir les inconvénients susmentionnés des mémoires, et dans le but de faire un ensemble de technologies complémentaire adaptable suivant l'environnement des cahiers de charges, il existe trois types d'FPGAs reprogrammables suivant la technologie de mémorisation pour répondre aux différentes applications.

Les trois principales technologies d'FPGA sont :

- **T**echnologie de programmation par **RAM**.
- **T**echnologie de programmation par **EEPROM** ou **FLASH**.
- **T**echnologie de programmation par **ANTI-FUSIBLE.**

II.4.6.1)-Technologie à base de RAM (XILINX et ALTERA)

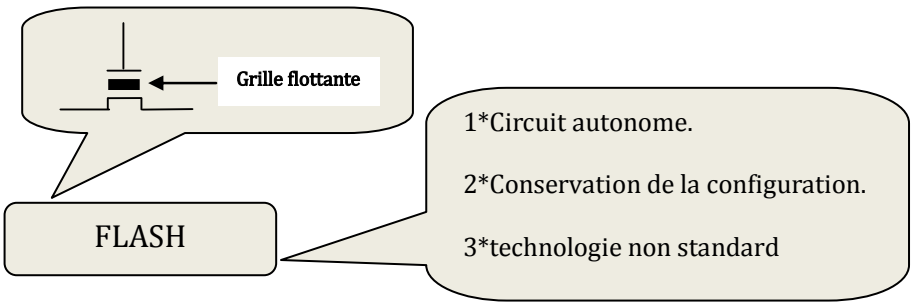
Cette technologie permet d'avoir une reconfiguration rapide des FPGA. Les points de connexions sont des ensembles de transistors commandés. L'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du FPGA dans une autre mémoire Flash.



Figure(I.08): *Caractéristiques des technologies **SRAM***

II.4.6.2)- Technologie à base d'EEPROM ou FLASH (LATTICE et ACTEL)

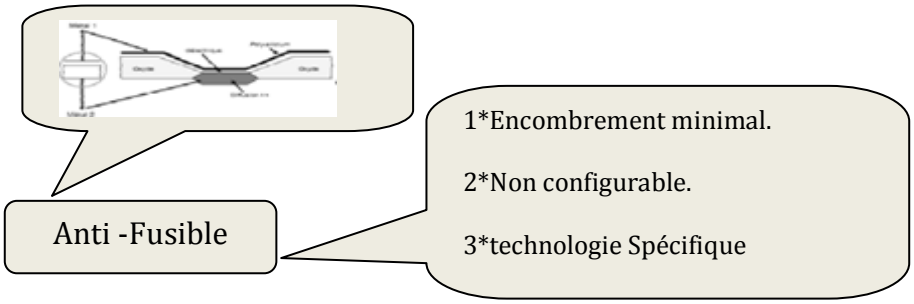
Cette technologie garde sa configuration mais un nombre limité de configuration avec une configuration plus lente par rapport à SRAM.



Figure(II.09): *Caractéristiques des technologies **FLASH***

II.4.6.3)- Technologie à base d'anti-fusibles (ACTEL)

Les points de connexions sont du type ROM, c'est-à-dire que la modification du point est irréversible. Pour comprendre le mécanisme de connexion sans rentrer dans les détails des semi-conducteurs, on considère que le point de connexion est le point de rencontre de deux segments conducteurs ou lignes conductrices. Le nom anti-fusible vient du fait que l'état initial du fusible ou la couche isolante est présente et il n'y a pas de contact pour l'établir, il faut détruire le fusible ce qui est contradictoire au fonctionnement habituel d'un fusible. Des composants moins génériques mais plus petits et plus rapides ont été développés.



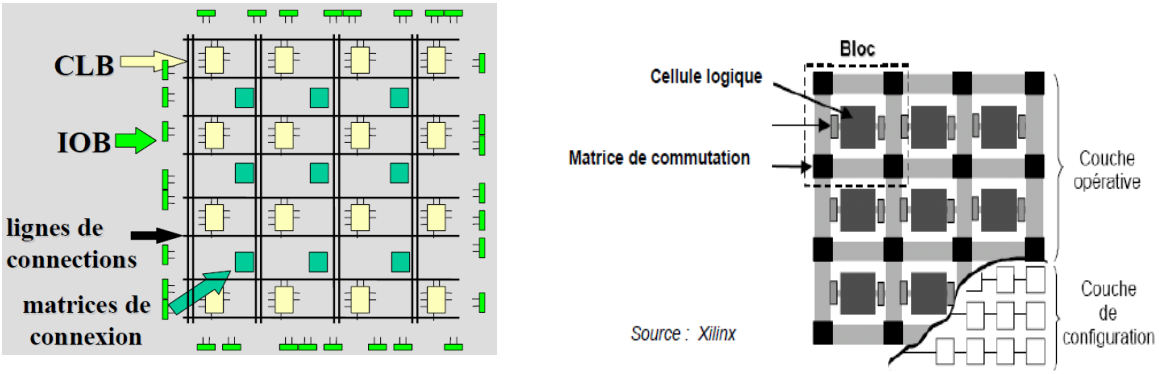
Figure(II.10): *Caractéristiques des technologies **ANTI-FUSIBLES***

II.5)- Architecture interne des FPGA [28]

On appelle les FPGA quelques fois LCA, abréviation anglaise de « **L**ogic **C**ell **A**rray » signifiant réseau de cellules logiques. Pour réussir à implanter un système dans un FPGA de manière efficace, il est indispensable de bien connaître sa structure interne et ses limites du point de vue performances. Les composants logiques programmables sont des circuits composés de nombreuses cellules logiques élémentaires librement assemblables. Celles-ci sont connectées de manière définitive ou réversible par programmation afin de réaliser les ou les fonctions numériques désirées. Un FPGA (**F**ield-**P**rogrammable **G**ate **A**rray) est un circuit intégré avec une structure adaptable par l'utilisateur et composée d'un réseau de cellules élémentaires ou d'éléments logiques programmables CLB et IOB répartis régulièrement et reliés entre eux grâce à des connections qui forment une matrice de routage programmable pour obtenir un comportement spécialisé du circuit dans sa globalité. Puisque tous les éléments sont programmables.

L'ensemble des systèmes reconfigurables FPGA est subdivisé en trois catégories suivant les fonctions préexistantes et des possibilités de les interconnectées. Ces catégories sont : des systèmes reconfigurables nommés "**grain fin**", des systèmes reconfigurables nommés "**grain moyen**" et des systèmes reconfigurables nommés " **grain large**".

L'architecture interne des FPGA est différente d'un fondeur à un autre et même entre les différentes gammes du même constructeur mais rien n'empêche que leurs ressemblances peuvent être rassemblées dans le schéma représentatif de la figure suivante:

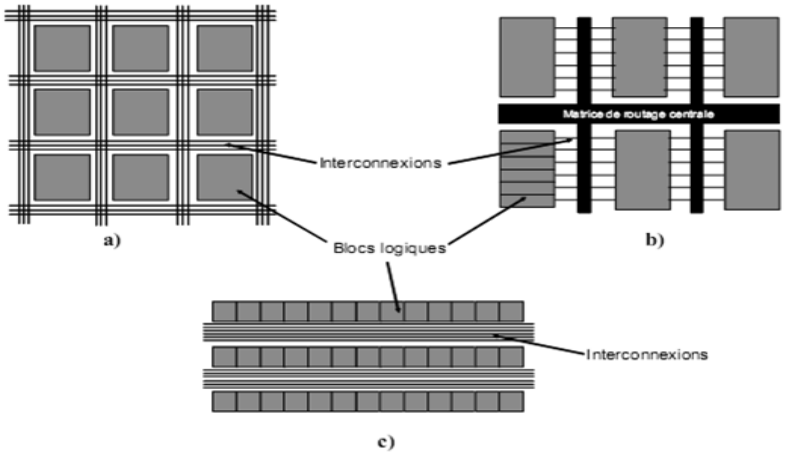


Figure(II.11): *Architecture interne d'un **FPGA***

- Les macro-cellules internes sont appelées :
  - **S**oit **CLB** qui est la dénomination adoptée par Xilinx et abréviation anglaise de « **C**onfigurable **L**ogic **B**lock », signifiant bloc logique configurable.
  - **S**oit **LC** qui est le nom choisi par Cyprès et abréviation anglaise de « **L**ogic **C**ell », signifiant cellule logique.
  - **S**oit **LE** qui c'est l'appellation d'Altéra abréviation anglaise de « **L**ogic **E**lement» signifiant élément logique.
- Les macro-cellules sur la périphérie sont appelées : **IOB** abréviation anglaise de « **I**nput **O**utput **B**lock », signifiant bloc logique d'entrées sorties.
- L'ensemble des points de connexion est appelé **PIP**, abréviation anglaise de « **P**rogramme **I**nterconnect **P**oints ». La granularité des FPGA par les macro-cellules CLB nous permet d'implémenter des fonctions logiques «combinatoires ou séquentielles » complexes car chaque CLB est constitué d'une partie combinatoire et d'une partie séquentielle. Chaque fonction est décomposée en petites fonctions booliennes qui peuvent être contenues par de petites cellules élémentaires SLICES.

Ces dernières comportent des LUT pour la partie combinatoire et une ou des bascules (généralement de type D) pour la partie séquentielle.

Les architectures existantes peuvent être regroupées en trois grandes catégories suivant la manière dont les blocs logiques sont organisés comme le montre la figure suivante:



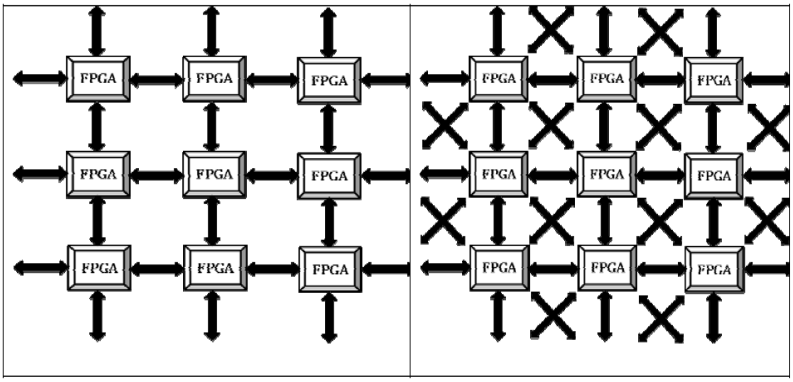
Figure(II.12): *Différentes architectures des **FPGA***

II.5.1)- Architecture MULTI-COMPOSANTS [29]

Dans un environnement multi-composant, plusieurs structures sont possibles pour un certain nombre de FPGAs qui admettent l'association et comme exemple:

II.5.1.1)- Association de plusieurs FPGA

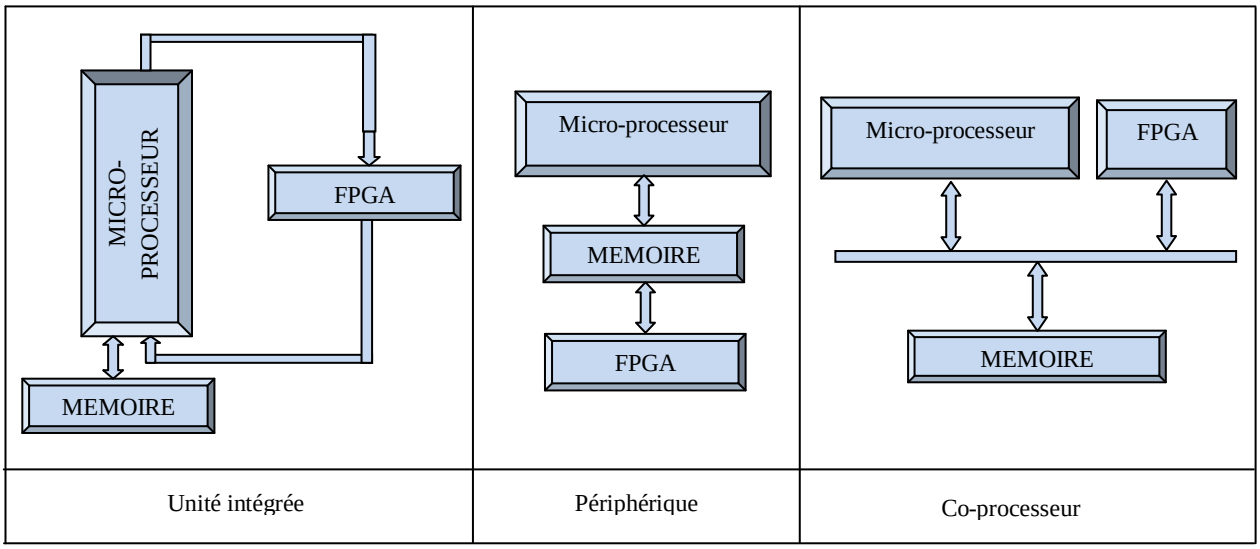
L'association de plusieurs FPGA prend différentes architectures suivant le besoin voici deux exemples d'association



Figure(II.13): *Exemples d'association entre **FPGA***

II.5.1.2)- Association d'un microprocesseur et d'un FPGA

La combinaison entre un **FPGA** et un processeur est possible dans certains cas comme accélérateurs matériels où le système reconfigurable est directement couplé à un processeur ce qui constitue un SOC**.** Le microcontrôleur ou le microprocesseur est généralement le hôte (organise, initialise, charge les programmes…..).



Figure(II.14): *Exemples d'association entre **FPGA et Microprocesseur**.*

II.5.2)-Avantages et inconvénients des FPGA [30]

Les avantages et les inconvénients des FPGA sont multiples on trouve :

AVANTAGES	INCONVÉNIENTS
Technologie « facile » à maîtriser. **T**emps de développement réduit. **R**eprogrammable. **I**déal pour le prototypage. **C**oût peu élevé. **P**arallélisme de traitement. **F**lexibilité et la possibilité de réduire **F**ortement les délais de développement et commercialisation. **L**a reconfiguration, parfois en temps réel.	**P**erformances non optimisées. **T**emps de réponse long par rapport aux **ASIC**.

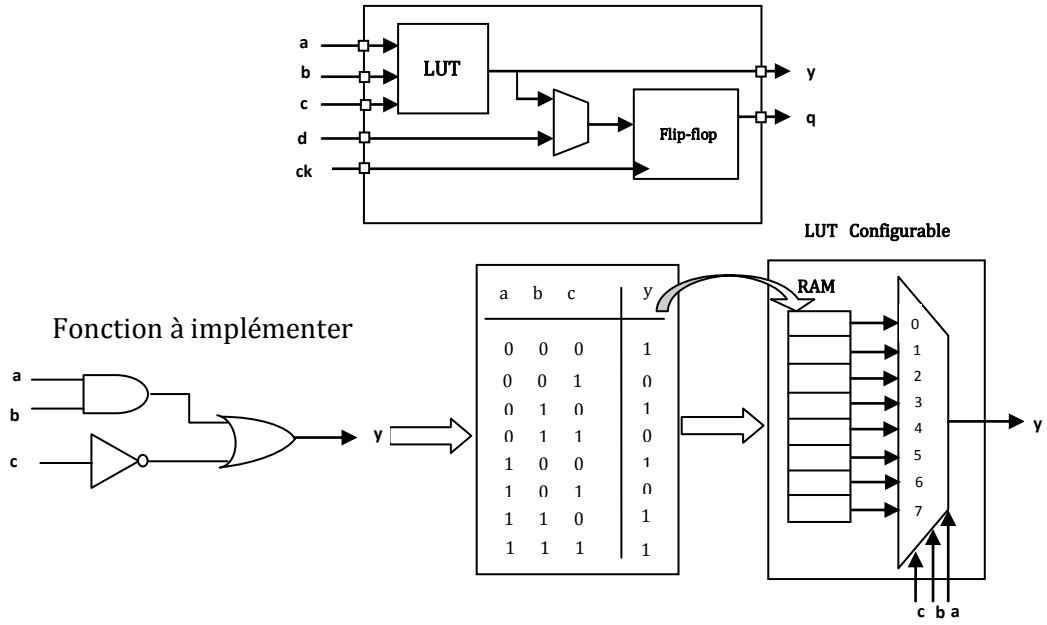
Tableau (II.03) : *avantages et inconvénients des FPGAs*

II.5.3)-Les deux grandes familles architecturales d'FPGA

Les familles des FPGA peuvent se regroupées en deux groupes :

II.5.3.1)-Les circuits FPGA à base de « LUT » (Look Up Tables)

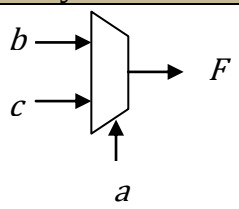
Les LUT (**L**ook **U**p **T**ables) ressemblent aux tables de vérité des fonctions logiques et réalisables par des mémoires de type SRAM. Aujourd'hui, la structure la plus utilisée est basée sur ce type (Look-Up Table). Les possibilités offertes par les circuits programmables FPGA a SRAM permettent par ailleurs de mettre en œuvre le concept de prototypage (ou maquette) pour la vérification fonctionnelle de systèmes sur puce pour certaines applications. La fonction de la LUT est de stocker la table de vérité de la fonction combinatoire à implémenter comme le montre la figure suivante.

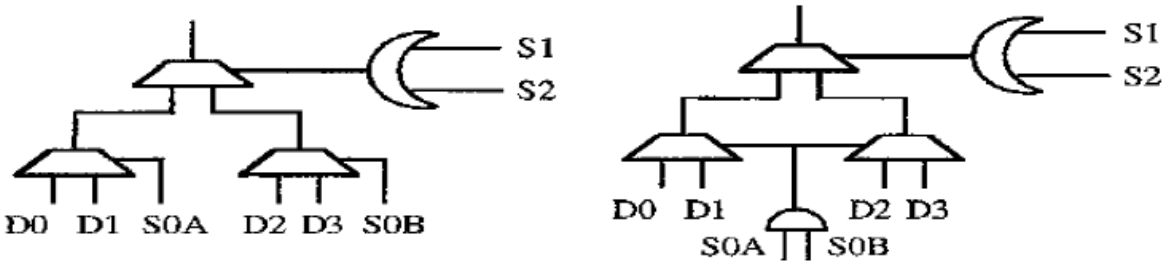


Figure(II.15): *Exemple d'implémentation sur* ***LUT*** **[31]**

II.5.3.2)-Les circuits FPGA a base de multiplexeurs « MUX »

Les FPGA à base de multiplexeurs qui sont des microcellules à trois entrées capable de réaliser la fonction suivante :

Equation logique	Symbole
$F = (a\ AND\ b) OR (\overline{a}\ AND\ b)$	



Figure(II.16): *Exemple d'implémentation sur des multiplexeurs.*

II.6)- LA CONFIGURATION DES FPGA PAR LES OUTILS CAO

II.6.1)- De l'algorithmique à la conception CAO [32]

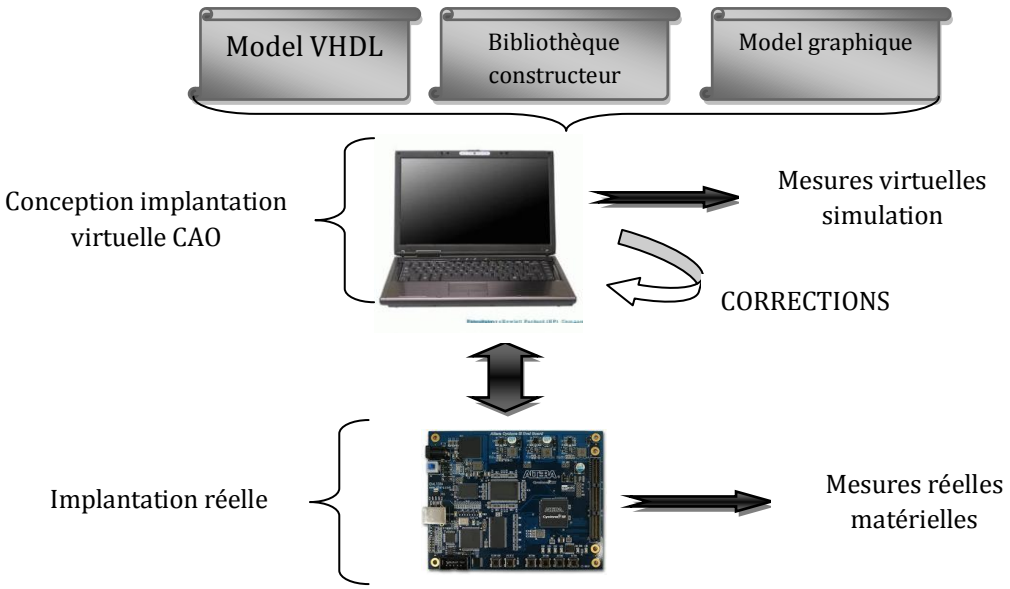
L'origine du mot algorithme provient du nom latinisé d'Al-Khawarizmi (ABOU JAFAR MUHAMMED IBN MUSA AL-KHAWARIZMI médecin arabe du moyen âge). Un algorithme est une suite ou séquence de raisonnements réalisés par un nombre fini d'opérations en termes de temps et de support matériel afin de fournir une solution à certains problèmes.

Un algorithme est présenté sous forme d'une prescription qui peut avoir différentes formes (textuelle, graphique, formule mathématique, diagramme de séquence…..etc.). La plupart des algorithmes existants sont orientés vers une implémentation logicielle ce qui est contraire à notre application qui s'agit d'une implantation matérielle. L'objectif des algorithmes sont récapitulés comme suit:

- **T**ransmettre un savoir faire.
- **D**écrire les étapes à suivre pour réaliser un travail.
- **E**xpliciter clairement les idées.

Les techniques de conception CAO (**C**onception **A**ssistée par **O**rdinateur**)** sont aujourd'hui très éprouvées et largement employées afin de concevoir des circuits électroniques nécessaires à mettre en pratique les connaissances algorithmiques.

L'approche moderne pour la conception des circuits (logiques) électriques et la manière d'introduire une fonctionnalité sur un support physique sont confiées aux outils CAO**.** Les outils CAO sont utilisés pour générer le fichier de configuration des FPGA qui s'appelle (bit-Stream) à partir d'une description de haut niveau. Les principaux rôles confiés aux outils CAO sont : la description, la simulation, la synthèse, le placement et le routage. Un design peut être conçu à l'aide d'un éditeur schématique ou d'un outil de traitement de textes.



Figure(II.17): *Mode d'exécution matériel de la* ***CAO***

La conception de circuit simple peut se faire par l'approche schématique mais dans les circuits complexes elle cède le champ de conception à l'approche textuelle.

II.6.2)- Méthodologie de conception [33]

Le flot de conception d'un système sur puce regroupe plusieurs niveaux d'abstraction.

Dans chaque niveau, le concepteur s'intéresse à la résolution d'un problème. Les outils de CAO sont utilisés intensivement et assurent la transition entre les différents niveaux d'abstraction.

Nous pouvons traiter un système complexe de deux manières qui sont :

- **L**'approche dite « descendante » (ou « **top-down** » en anglais).
- **L**'approche dite « ascendante » (ou « **bottom-up** » en anglais).

REMARQUE : Il y a une grande similitude de conception pour les FPGA, les CPLD et les ASIC.

Le développement d'une application sur FPGA par des outils CAO suit l'enchainement des étapes suivantes:

1)- Spécification du design

- **L**e nombre de broches d'entrée-sortie et leur localisation dans la puce FPGA.
- **L**a spécification de la fréquence d'horloge du système.
- **L**a spécification de la mémoire requise pour l'application.

2)- Développement du design

- **S**pécification de la méthodologie de design (Outil de développement utilisé).
- **L**a saisie du circuit Codage RTL (VHDL, Verilog...)
  - **G**raphique (Machine à états).
  - **S**aisie HDL (**H**ardware **D**escription **L**anguage).
- **L**a simulation (Pré et Post synthèse).

3)- Synthèse

La synthèse est le processus qui convertit la représentation du design à partir du code HDL fourni pour produire une représentation au niveau porte logique. Elle s'occupe de déterminer quelles sont les structures susceptibles de répondre au cahier des charges étudié et de produire un code booléen unique sous forme d'un fichier.

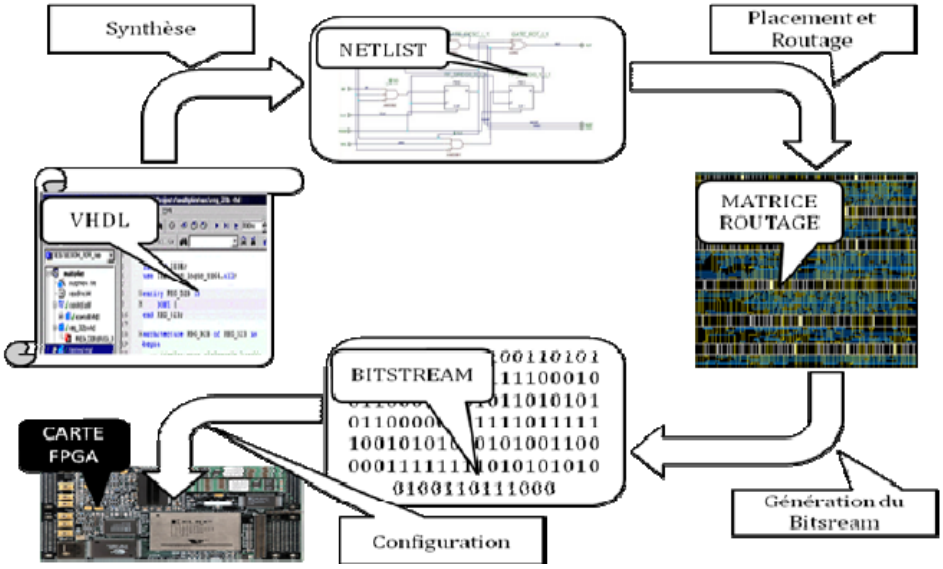
4)- Placement et routage

A partir des fichiers de synthèse, l'outil de conception procède au placement et routage. Un algorithme de routage est sensé de faire l'aiguillage des données qu'il reçoit vers leurs destination par action sur les nœuds de routage ce qui est équivalent à définir les chemins qui relient l'ensemble des **CLB** contenus dans la fonction désirée. Ces algorithmes de routage sont différents d'un concepteur à un autre. Plusieurs traitements sont nécessaires pour obtenir un fichier de configuration :

- **Partitionnement :** Les équations logiques sont partitionnées en un autre ensemble équivalent d'équations. Chaque équation de ce nouvel ensemble peut être implantée dans un seul bloc logique du composant cible **FPGA**.
- **Placement :** Des blocs logiques sont sélectionnés dans la matrice et affectés au calcul des nœuds du réseau booléen.
- **Routage :** Les ressources d'interconnexion sont affectées à la communication de l'état des nœuds du réseau vers les différents blocs logiques qui en ont besoin.
- **Génération des données numériques de configuration :** Les informations abstraites de routage, de placement et les équations implantées dans les blocs sont transformées en un ensemble de valeurs numériques, qui seront chargées sur le composant **FPGA**.

5)- Intégration et implémentation [34]

L'implémentation est la réalisation proprement dite qui consiste à mettre en œuvre l'algorithme sur l'architecture du circuit configurable cible, c'est-a-dire à compiler, charger, puis lancer l'exécution sur un ordinateur ou calculateur. C'est une étape de programmation physique et de tests électriques qui clôture la réalisation du circuit. La figure suivante résume l'ensemble de ces étapes.



Figure(II.18): *Etapes de conception sur **FPGA***

6)- Etapes d'implantation

La Figure (II.19) montre les différentes phases pour l'implantation de fichiers VHDL sur un FPGA de la famille Altéra.

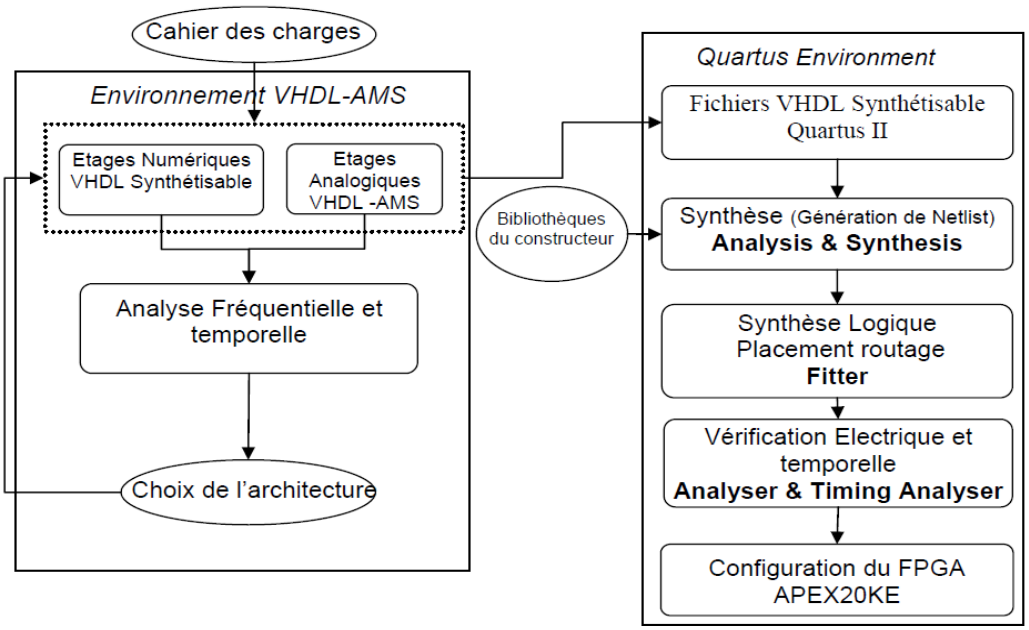


Figure (II.19): *Méthodologie pour un FPGA de la famille ALTERA*

Nous trouvons 4 phases :

- L'outil « Analysis & Synthesis » vérifie la syntaxe et la sémantique du code et réduit au minimum la logique de conception. Il génère ensuite une netlist qui détermine le nombre d'éléments logiques utiles à la réalisation du système et la connexion entre ces éléments logiques.

- La deuxième phase est le placement/routage qui est défini à partir de la netlist des éléments logiques. C'est la phase qui consomme le plus de temps car c'est lors de cette étape que le logiciel choisit les positions physiques des cellules et les chemins des signaux pour une utilisation optimale et réduite.
- La phase suivante correspond à la vérification des choix faits lors de la phase précédente et la génération des fichiers vers le FPGA. Elle permet d'analyser les différents chemins et les délais de propagation. Après cette étape, nous pourrons connaître le délai pour le chemin le plus critique.

  Les résultats de cette phase sont établis sous forme de rapport et déterminent les performances du composant (occupation du FPGA, Puissance, température de fonctionnement, schémas des connexions …).
- La dernière phase implante le circuit conçu par programmation et configure les interrupteurs en fonction des résultats du placement/routage.

II.7)- Le FPGA CYCLONE III [35]

La famille CYCLONE III du constructeur ALTERA ; offre grandes performances ; constituée de 8 modèles, ceux-ci fournissent une haute densité d'intégration se situant entre 5000 et 120000 de porte logiques (tableau II.03). Ils possèdent des interfaces rapides pour la connexion à des mémoires externes.

Device	Logic Elements	Number of M9K blocks	Total RAM Bits	18*18 Multipliers	PLLs	Global Clock networks	Maximum User I/Os
EP3C5	5,136	46	423,936	23	2	10	182
EP3C10	10,320	46	426,936	23	2	10	182
EP3C16	15,408	56	516,096	56	4	20	346
EP3C25	24,624	66	608,256	66	4	20	215
EP3C40	39,600	126	1, 161,216	126	4	20	535
EP3C55	55,856	260	2, 396,160	156	4	20	377
EP3C80	81.264	305	2, 810,880	244	4	20	429
EP3C120	119,088	432	3, 981,312	288	4	20	531

Tableau (II.03) : *La famille CYCLONE III*

Les fonctionnalités offertes par ces composants sont les suivantes :

- Environ 120 000 portes logiques (entre 5 000 et 120 000)
- Nombre de cellules IOB supérieur à 531
- Supporte standards d'entrée/sortie : LVDS, HSTL, PCI ….
- Disponibilité de mémoire vive sous forme de block RAM (RAM distribuée)
- Possède des multiplieurs 18 × 18 bits incorporés.
- 20 lignes principales d'horloges….

II.7.1)- Architecture générale

L'architecture du CYCLONE III est constituée d'une structure régulière et flexible de cellules LAB programmables, entourée par un périmètre de cellules IOB (figure II.20), il y a quatre PLL (Phase-Locked Loops) à chacune des extrémités de la matrice. Selon le modèle, il peut y avoir jusqu'à deux colonnes de block RAMs d'une capacité de 9K chacune. Tous ces éléments sont interconnectés à l'aide d'une puissante hiérarchie de segments métalliques (routage).

Cette structure est composée de deux types d'élément de base, ainsi que de structures d'interconnexion et de cellules gérant les entrées/sorties (IOB). Les deux éléments de bases sont :

- Les éléments logiques élémentaires, notés LE_s ;
- Les matrices d'éléments logiques élémentaires, notés LAB

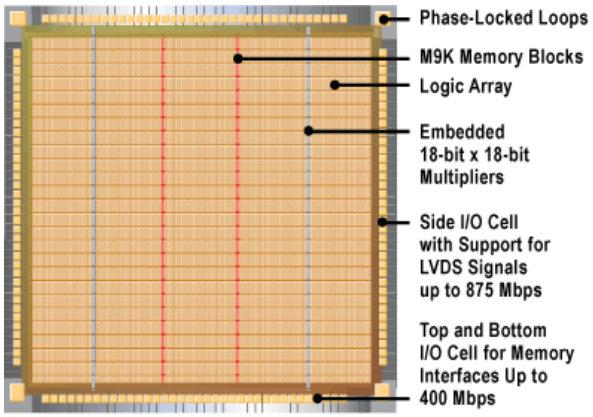


Figure (II.20): *Architecture générale du CYCLONE III* **[35]**

II.7.2)- Les éléments logiques LEs [35]

Les éléments logiques LE_s correspondent à la plus fine granularité pour le type de FPGA décrit ici. Leur structure est présentée sur la figure (II.21). Ils sont composés d'une table LUT à quatre entrées permettant de réaliser des fonctions combinatoires à quatre variables.

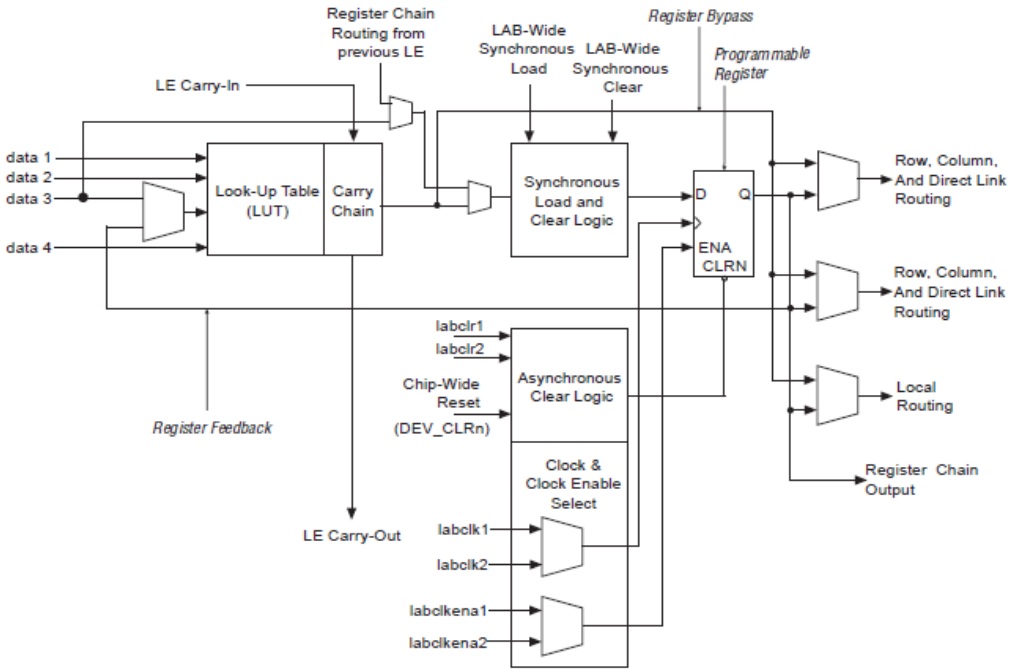


Figure (II.21): *Architecture d'une cellule logique*

Le second élément important est un registre programmable, autorisant avec la table LUT la gestion de fonctions de logique synchrone, on peut le configuré par des bascules D, T, JK, ou RS, chaque registre contient des entrées de données (data), l'horloge (CLK), le signal de validation de l'horloge (Clock Enable), Les signaux qui utilisent le réseau d'horloge globale, les différentes broches (I/O).

Enfin, des chemins spécifiques sont intégrés. Ils sont dédiés à la propagation des retenues des opérations arithmétiques de base et à l'utilisation en cascade de plusieurs LEs pour réaliser des fonctions comportant plus de quatre entrées.

Il faut remarquer que ces cellules ont deux modes distincts de fonctionnement :

- Mode normal ;
- Mode arithmétique.

Ces deux modes de fonctionnement sont les plus intéressants pour les algorithmes de traitement du signal.

II.7.2.1)- Mode de fonctionnement normal

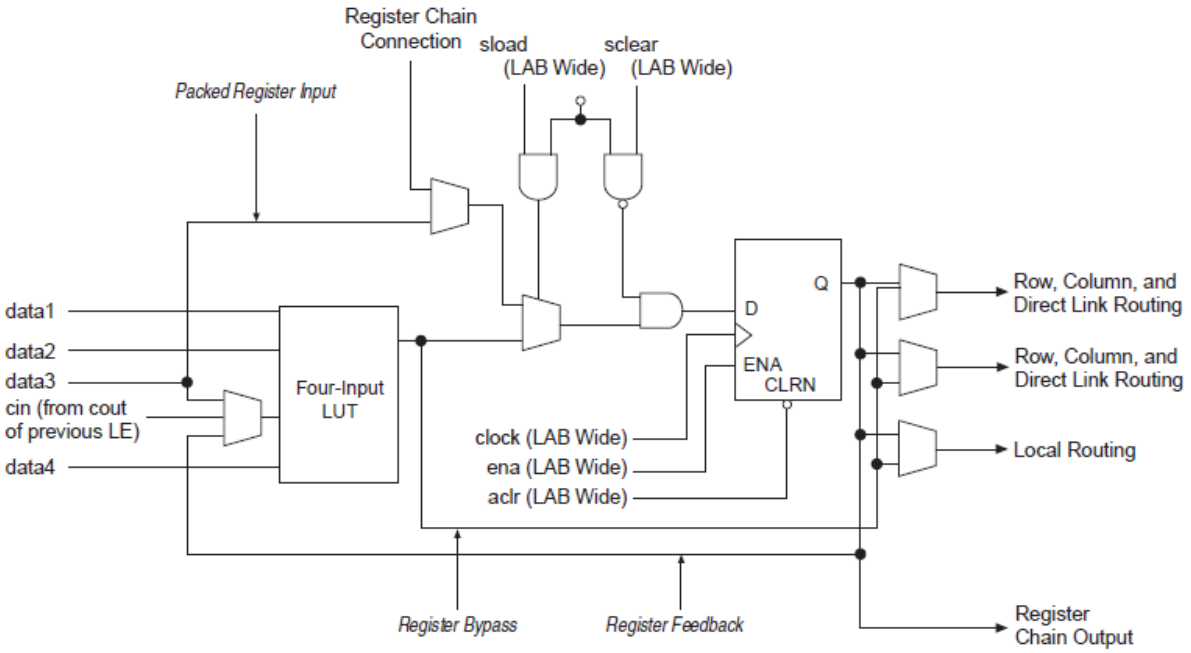


Figure (II.22): *Élément logique en mode de fonctionnement « Normal »*

Ce mode propose une table LUT à quatre entrées permettant de créer des fonctions combinatoires classiques. Le schéma structurel d'un élément logique fonctionnant dans ce mode est présenté sur la figure (II.22). La présence d'un chemin d'interconnexion dédié et rapide entre les LE_s d'un même LAB permet de réaliser des fonctions combinatoires plus complexes utilisant plus de quatre entrées. Le compilateur sélectionne automatiquement le carry-in (CIN) ou le signal 'data3' comme l'une des entrées de la LUT.

Le mode «normal» ne propose pas de chemin optimisé pour les propagations de retenues. Il est donc plus spécifiquement utilisé dans le cas de fonctions combinatoires simples.

II.7.2.2)- Le mode arithmétique

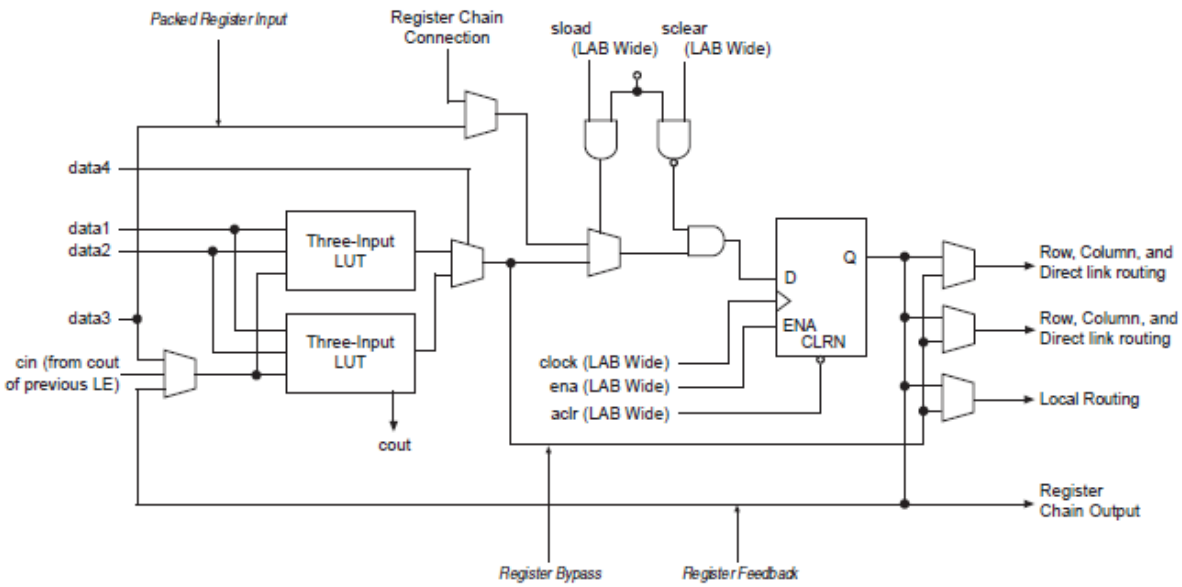


Figure (II.23): *Élément logique en mode de fonctionnement arithmétique* **[35]**

Propose deux LUT à trois entrées au lieu de la seule LUT à quatre entrées du mode normal. Cette architecture est particulièrement indiquée pour la réalisation d'additionneurs, compteurs, accumulateurs, et comparateurs, ou de fonctions équivalentes. En effet, dans ce cas l'une des LUT peut effectuer la somme de trois signaux (en tenant compte de la retenue éventuelle provenant d'un autre étage du compteur), pendant que la seconde table est utilisée pour calculer la valeur de la retenue qui sera propagée aux étages suivants du compteur.

Comme dans le mode normal, différents LE peuvent être cascadés pour réaliser des fonctions combinatoires plus complexes.

Ce mode comble l'une des lacunes du mode « Normal » en proposant un chemin adapté à la propagation rapide de retenues. En contrepartie, les fonctions logiques que le LE_s est capable de traiter sont limitées à deux entrées de données. De plus, les chemins optimisés pour les retenues sont cantonnés à un même LAB. Dans le cas d'une fonction complexe s'étendant sur plusieurs LAB, le bénéfice est perdu.

II.7.3)- Les blocs logiques « LAB »

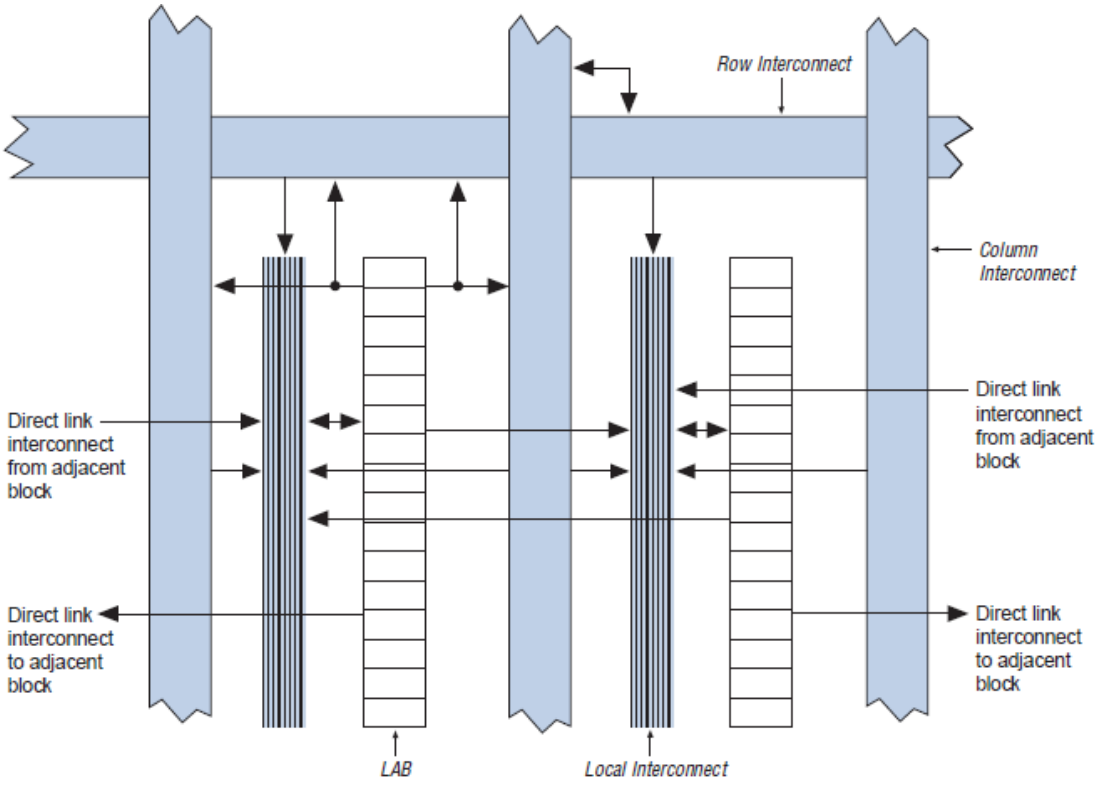


Figure (II.24) : *Architecture d'un bloc logique* **[35]**

Ils sont composés de 16 blocs logiques élémentaires de type LE. La figure (II.24) présente l'architecture interne d'un LAB. Ils comportent également des chemins optimisés pour la propagation de retenues, utiles en particulier dans le cadre de l'implantation de fonctions arithmétiques comme les additionneurs. De plus, des chemins d'interconnexion rapides et simplifiés entre les différents LE d'un même LAB sont disponibles. Des signaux de contrôle programmables sont fournis et peuvent servir à acheminer un signal d'horloge ou de type reset asynchrone.

Les LAB facilitent la combinaison de plusieurs LE et donc la réalisation de fonctions combinatoires ou séquentielles plus complexes. En profitant des chemins de propagation intégrés, les fonctions implantées dans un seul LAB bénéficient de délais de propagation réduits entre les différents LEs intégrés. Mais un LAB ne regroupe qu'un nombre restreint de LE (16 sur l'architecture CYCLONE III). Les fonctions trop complexes, ou qui nécessitent un trop grand nombre d'entrées/sorties sont donc implantées sur plusieurs LAB distincts, et les délais de propagation entre cellules logiques s'en trouvent dégradés.

II.8)- Présentation de la carte CYCLONE III [35]

- La carte : STARTER KIT, CYCLONE III, FPGA
- Silicon Fabricant: Altéra
- Core Architecture: FPGA
- Nombre de noyau de silicium: EP3C
- Régulateurs abaisseurs LTC3413, LT1959, LT1117
- Applications Outils / Carte: Processeur / Microcontrôleur intégré

- **Mémoires**

  - 32 Mégabits (Mb) de mémoire DDR SDRAM
  - 1 Mégabit (Mb) de mémoire SSRAM
  - 16 Mo de mémoire flash Intel P30/P33

- **Horloge**

  - Un oscillateur 50 MHz

- **Interrupteurs et Indicateurs**

  - 6 Boutons poussoirs en total
  - 7 LEDs en total

- **Outils / carte d'application:** Conception embarqué
- 216 broches (I/O)
- Le composant FPGA CYCLONEIII EP3C25.

- **Connecteurs**
  - Connecteur HSMC.
  - USB type B
- Entrée alimentation 12V.

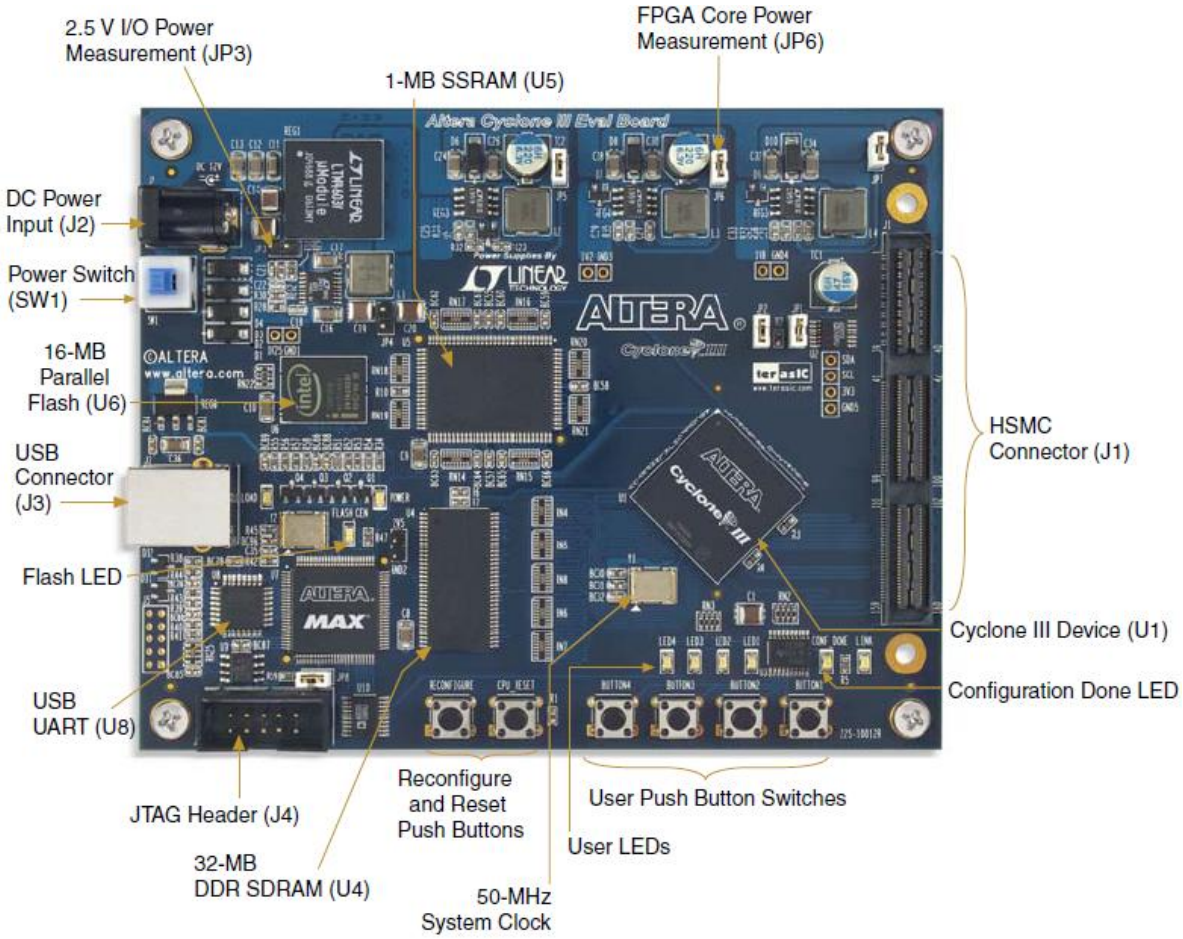


Figure (II.25) : Carte de développement ALTERA - STARTER KIT, CYCLONE III, FPGA

II.9)- Limitations et avantages pour le traitement numérique du signal

La structure générale d'un FPGA en fait un outil de premier plan pour implanter des algorithmes de traitement du signal et notamment, les algorithmes de la fonction de corrélation qui nous intéressent, Cependant, nous avons dans notre architecture un grand nombre de multiplieurs et additionneurs.

Un multiplieur peut être implanté en utilisant par exemple le *mode arithmétique* des éléments logiques. La complexité dans le domaine des circuits FPGA est d'abord mesurée par le nombre d'élément logiques, qui constituent la partie élémentaire du circuit. Ceci est différent dans les circuits de type ASIC, où cette complexité est évaluée en nombre de portes. Une seule cellule logique peut être équivalente à une architecture ASIC comprenant de (1) à (12) portes.

De plus pour les FPGA, le nombre de cellules logiques élémentaires disponible est limité dès la conception du circuit. Lorsque le taux de remplissage du FPGA approche de sa capacité limite, la vitesse de fonctionnement se trouve très affectée par les délais d'interconnexion entre les différents éléments logiques.

Par conséquent, lors de la conception d'une architecture numérique dont la cible est un circuit FPGA, il est nécessaire de limiter la complexité des fonctions arithmétiques et logiques dans les limites imposées par la taille des structures élémentaires disponibles.
Autant que possible, les fonctions ne doivent pas s'étendre sur plusieurs LAB, ni dépasser la taille des CLB (Configurable Logic Bloc). Les cellules élémentaires de ces circuits sont composées d'une fonction combinatoire simple suivie d'un registre. Cela favorise les architectures tendant à reproduire cette structure, et ce sont celles-ci qui présenteront les meilleures performances sur ce type de circuit.

Les fonctions arithmétiques mises en œuvre dans les architectures analogiques décrites plus haut sont trop complexes pour permettre une utilisation optimale d'un FPGA.
Il est donc nécessaire de définir de nouvelles fonctions équivalentes, mais mieux adaptées à la structure interne de ces circuits. **[36]**

II.10)- Conclusion

Au départ, nous avons présenté un survol des circuits programmables puis nous avons étudié l'état d'art des FPGA ce qui nous a permis de conclure que la technologie FPGA s'inscrit au sommet de l'évolution des composants logiques. Le besoin croissant de composants plus performants, plus économiques et disponibles en grandes quantités, et les grands axes du progrès sont disponibles dans les FPGA. Enfin nous avons présenté l'architecture de notre circuit CYCLONE III et une présentation globale de notre carte.

CHAPITRE III

PRESENTATION DE LA REALISATION

III.1)- Introduction

La conception et l'implantation du corrélateur numérique sur circuit FPGA consiste à trouver une architecture en parallèle basée sur les registres à décalage, les MACs et une description de haut niveau VHDL. Nous allons donc décrire le circuit sous forme d'algorithme, et ce après avoir choisi l'architecture du corrélateur.

III.2)- La conception synchrone [37]

Les circuits FPGA sont des circuits numériques basés sur la logique programmable, ce sont des circuits séquentiels synchrones. La logique synchrone exige que toutes les entrées horloge des bascules, des registres, des compteurs soient attaquées par un seul et même signal d'horloge. Entre deux fronts actifs de cette horloge, il n'y a aucune évolution possible du système logique, même si des variables asynchrones changent d'état.

III.2.1)- Les différentes parties d'un système synchrone

Un système synchrone ou logique est constitué de deux grandes parties :

- *La partie opérative* : qui rassemble tous les composants et sous ensembles nécessaires à l'exécution des fonctions (ou opérations) que doit effectuer le système synchrone : bascules, registres, compteurs, bus,…
- *La partie commande* : appelée aussi partie contrôle, rassemble tous les circuits nécessaires à la génération des signaux de commande de la partie opérative : bits de sélection d'un multiplexeur, signaux d'écriture ou de lecture de mémoires…, la partie commande contient une machine d'états ainsi que des circuits secondaires tels que des compteurs. Elle reçoit des signaux d'information de la partie opérative à laquelle elle renvoit des ordres. Elle reçoit également des commandes de l'environnement externe auquel elle retourne des signaux d'état :

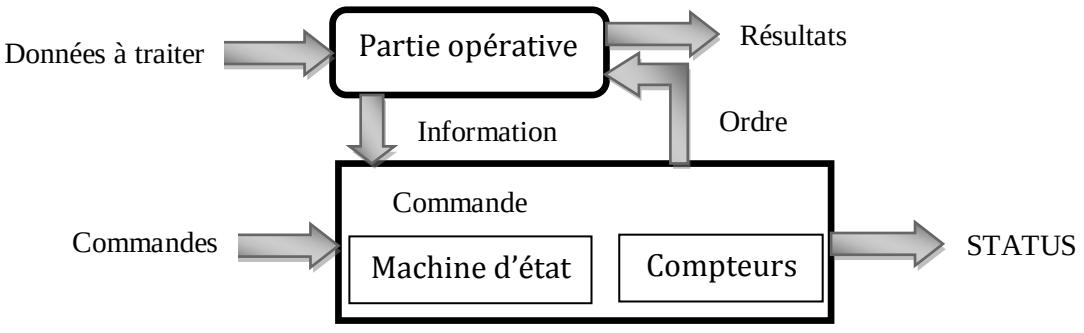


Figure (III.1) : *Modèle de GLUSHKOV d'une machine de traitement*

III.2.2)- Les machines d'état [38]

Tout système logique doit être placé dans un état connu à l'initialisation, ce qui rend cette dernière indispensable.

Une machine d'états présente les principales caractéristiques suivantes:

- L'automate est toujours dans un seul des états possibles : l'état courant. Cet état est stocké dans une mémoire particulière, en générale appelée « registre d'états ».
- L'état suivant peut être calculé à partir de l'état courant et/ou de la valeur des entrées.
- La valeur des sorties est calculée à partir de l'état courant et/ou de la valeur des entrées.
- A chaque cycle d'horloge (l'automate est dit synchrone), le registre d'états est mis à jour avec l'état préalablement calculé (l'état suivant).

Il existe deux types de machine d'état : la *machine de Moore* et la *machine de Mealy* comme l'illustre la figure (III.2).

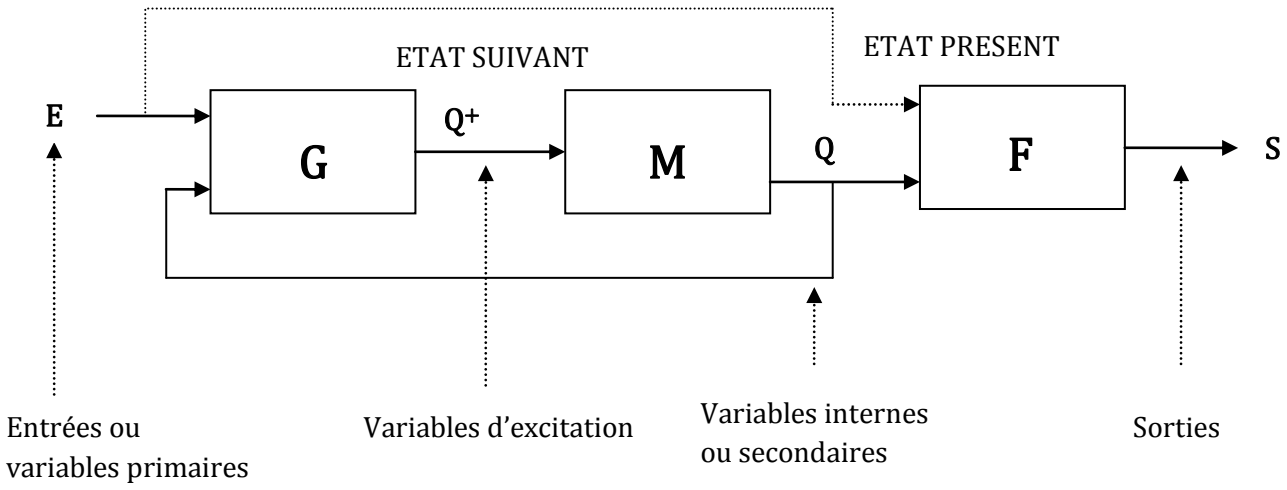


Figure (III.2) : *Machine d'état : si la liaison en pointillé existe, on a une machine de Mealy. si la liaison en pointillé n'existe pas, on a une machine de Moore*

Pour chaque front actif de l'horloge, l'état suivant du système logique est calculé à partir des valeurs des entrées et de l'état présent. Ce calcul est la fonction du bloc combinatoire G, le seul bloc qui ne soit pas simple. Le bloc M est constitué de registres de synchronisation (également appelé registre d'états), se réduisant à des bascules D pour un système synchrone, toutes pilotées par l'horloge. Le bloc F permet la génération des signaux de sortie ; pour une machine de Moore ; ces valeurs ne dépendent que de l'état présent ; pour une machine de *Mealy*, elles dépendent de l'état suivant mais aussi des entrées.

III.3)- Schéma de calcul

Etant donnés 2 vecteurs X et Y à 256 éléments (indicés de 0 à 255), il s'agit de calculer les expressions suivantes :

$$\boldsymbol{\varphi(0) = x_0.y_0 + x_1.y_1 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{255}.y_{255}}$$

$$\boldsymbol{\varphi(1) = x_0.y_1 + x_1.y_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{254}.y_{255}}$$

$$\boldsymbol{\varphi(2) = x_0.y_2 + x_1.y_3 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{253}.y_{255}}$$

⋮

$$\boldsymbol{\varphi(254) = x_0.y_{254} + x_1.y_{255}}$$

$$\boldsymbol{\varphi(255) = x_0.y_{255}.}$$

Ces calculs nécessitent 32 640 multiplications et autant d'additions. En utilisant 1 seul MAC et une horloge cadencée à 10 ns, ces calculs requièrent 326,40 μs. De plus, la recherche en mémoire des termes X(i) et Y(k) réclame au minimum 2 impulsions d'horloge ; ce qui porte la durée des calculs aux environs de 1 ms.

Le schéma de calcul que nous proposons vise à réduire la durée des calculs à quelques dizaines de μs.

Pour celà, nous utilisons un ensemble de 64 MAC fonctionnant en parallèle et allons agencer les données dans des registres à décalage. En principe, la durée minimum des calculs devrait être aux environs de 32 640 $\times 30ns$ /64 #15 μs.

Nous subdivisons les 256 expressions $\varphi(0)$ ... ... $\varphi(255)$ en 4 blocs à 64 éléments :

***Bloc 1: phase P =1**

$$\boldsymbol{\varphi(0) = x_0.y_0 + x_1.y_1 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{255}.y_{255}}$$

$$\boldsymbol{\varphi(1) = x_0.y_1 + x_1.y_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{254}.y_{255}}$$

⋮

$$\boldsymbol{\varphi(63) = x_0.y_{63} + x_1.y_{64} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{192}.y_{255}}$$

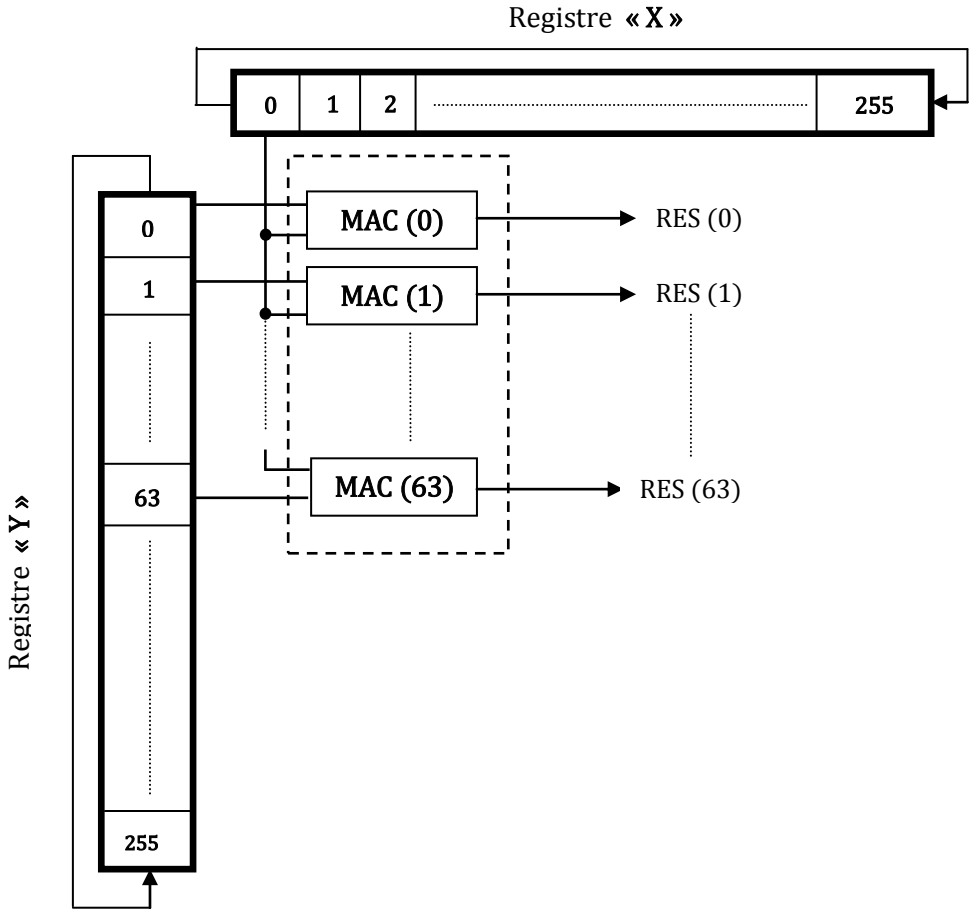


Figure (III.3) : *Schéma de calcul*

Nous disposons X et Y dans 2 registres à décalage de la façon indiquée dans le schéma.

Ainsi $RES(0) \dots \dots RES(63)$ contiendront les résultats de la 1^{ere} colonne du bloc $P = 1$.

Un décalage **simultané** de X et Y placera dans $RES(0) \dots \dots RES(63)$, les résultats des $1^{ère}$ et $2^{ème}$ colonnes du bloc $P = 1$.

Le processus de décalage se poursuit donc de cette façon jusqu'à la dernière colonne du bloc $P = 1$.

Le nombre de décalages nécessaires pour la phase $P = 1$ est donné par le nombre de termes contenus dans $\varphi(0)$; donc 255.

Mais en fait, les expressions $\varphi(k)$ d'ordre $K > 0$ ne nécessitent pas autant de décalages. Ainsi, pour $\varphi(63)$, seuls 192 décalages sont nécessaires. Au delà, l'accumulation au niveau de MAC63 doit être bloquée. $\varphi(62)$ Nécessite 193 décalages. Pour $\varphi(1)$, 254 décalages.

En définitive, l'accumulation dans le MAC d'indice $\mathbf{0 \leq i \leq 63}$ est bloquée si :

$$i + cx > 255 \qquad \text{(III.1)}$$

cx : Étant le nombre de décalages effectués par le registre X.

Quand l'ensemble des décalages auront été effectués et comme il s'agit de décalages par rotation,

i.e. $\begin{cases} x(255) \leftarrow x(0) \\ y(255) \leftarrow y(0) \end{cases}$

Nous retrouvons dans les registres les vecteurs X et Y tels qu'ils y étaient disposés en début de phase.

***Bloc 2: phase P = 2**

Nous devons calculer :

$$\boldsymbol{\varphi(64) = x_0 . y_{64} + x_1 . y_{65} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{191} . y_{255}}$$

$$\boldsymbol{\varphi(65) = x_0 . y_{65} + x_1 . y_{66} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{190} . y_{255}}$$

⋮

$$\boldsymbol{\varphi(127) = x_0 . y_{127} + x_1 . y_{128} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{128} . y_{255}}$$

L'étape P = 1 s'est achevée avec $y(0)$ dans la cellule 0 du registre Y et $x(0)$ dans la cellule 0 du registre X. Or, la présente phase commence avec $y(64)$ en cellule 0 du registre Y.

Donc, avant de démarrer les accumulations, nous devons effectuer 64 décalages de positionnement du registre Y.

Les décalages d'accumulation sont au nombre de termes contenus dans $\varphi(64)$, i.e 191.

L'accumulation dans le MAC d'indice *i* est bloquée quand :

$$i + cx + 64 > 255 \qquad \text{(III.2)}$$

***Bloc 3: phase P = 3**

Calcul de :

$\boldsymbol{\varphi(128) = x_0.y_{128} + x_1.y_{129} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{127}.y_{255}}$

$\boldsymbol{\varphi(129) = x_0.y_{129} + x_1.y_{130} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{126}.y_{255}}$

⋮

$\boldsymbol{\varphi(191) = x_0.y_{191} + x_1.y_{192} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{64}.y_{255}}$

Le nombre de décalages :

- Positionnement **: 128**
- Accumulation **: 127**

L'accumulation dans le MAC d'indice i est bloquée par : $i + cx + 128 > 255$ (III.3)

***Bloc 4: phase P = 4**

Calcul de :

$\boldsymbol{\varphi(192) = x_0.y_{192} + x_1.y_{193} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{63}.y_{255}}$

$\boldsymbol{\varphi(193) = x_0.y_{193} + x_1.y_{194} + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + x_{62}.y_{255}}$

⋮

$\boldsymbol{\varphi(255) = x_0.y_{255}}$.

- Décalages de Positionnement **: 192**
- Décalages Accumulation **: 63**

Blocage d'accumulation dans le MAC d'indice i :

$$i + cx + 192 > 255 \qquad \text{(III.4)}$$

Conclusion

Les 4 phases $P = 1,2,3$ et 4 nécessitent :

$64 + 128 + 192 = 384$ décalages de positionnement.

$63 + 127 + 191 + 255 = 636$ décalages d'accumulation.

Nous remarquons donc que les décalages de positionnement consomment plus du tiers ($\frac{1}{3}$) des décalages nécessaires pour l'ensemble des calculs.

Pour éliminer ces décalages de positionnement nous procédons de la façon suivante :

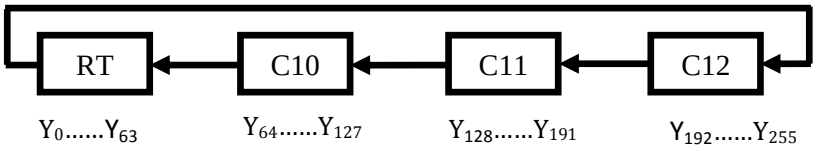
Vecteur Y :

- Les éléments de ce vecteur seront logés dans 4 registres à décalage à 64 cellules chacun.
- Un seul registre, le registre de traitement (RT) sera relié aux MAC.
- Les 3 autres registres tirent vers RT de manière à ce que les décalages de positionnement coïncident avec les décalages d'accumulation.

***Phase P=1 :**

Au cours des 192 premiers décalages, les 4 registres seront reliés ainsi :

(1)



Au départ, ces 4 registres contiennent les éléments de Y tels qu'indiqués.

Au 192eme décalage nous aurons :

$$y_{192} \dots\dots\dots y_{255} \quad \text{Dans } RT(0) \dots\dots\dots RT(63)$$

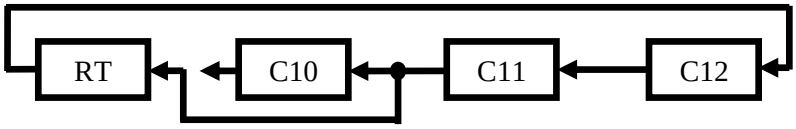
$$y_0 \dots\dots\dots y_{63} \quad \text{Dans } C_{10}(0) \dots\dots\dots C_{10}(63)$$

$$y_{64} \dots\dots\dots y_{127} \quad \text{Dans } C_{11}(0) \dots\dots\dots C_{11}(63)$$

$$y_{128} \dots\dots\dots y_{191} \quad \text{Dans } C_{12}(0) \dots\dots\dots C_{12}(63)$$

A partir de maintenant, i.e quand le décalage 192 à été effectué et jusqu'au décalage 255, nous enchainons les 4 registres ainsi :

(2)



Ainsi, un décalage d'accumulation coïncidera avec un décalage de positionnement pour la phase suivante.

A la fin de cette phase, nous aurons :

$y_{64} \dots \dots \dots y_{127}$ Dans $RT\ et\ C_{10}$

$y_{128} \dots \dots \dots y_{191}$ Dans C_{11}

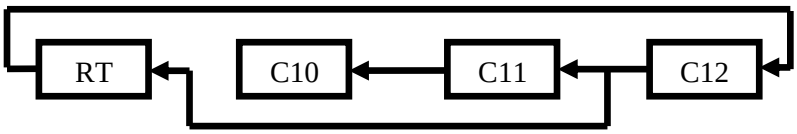
$y_{192} \dots \dots \dots y_{255}$ Dans C_{12}

Nous avons effectué 256 décalages.

***Phase P=2 :**

Au cours des 128 premiers décalages, nous enchainons les registres comme en **(2)**. Ensuite, nous enchainons les registres ainsi :

(3)

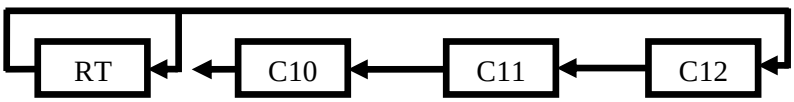


Total : 128 décalages

***Phase P=3 :**

Au cours des 64 premiers décalages, l'enchainement des registres est conforme à **(3)**. Ensuite, nous enchainons ces registres ainsi :

(4)



Total : 64 décalages.

***Phase P=4 :**

L'enchainement sera conforme à **(4)**.

Total : 64 décalages.

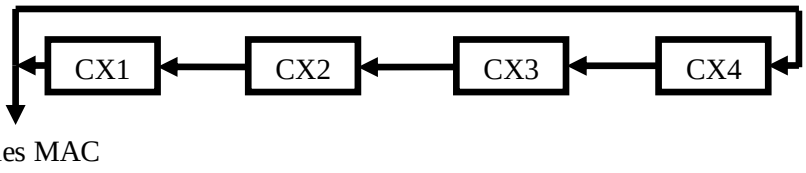
En définitive, nous aurons effectués : $256+192+128+64 = 640$ décalages.

Vecteur X :

- Les éléments de ce vecteur seront logés dans 4 registres à décalage à 64 cellules pour chacun : $CX1, CX2, CX3, CX4$.
- Un seul registre $CX1$, verra sa cellule 0 relié au MAC.
- Les 3 autres registres tirent vers $CX1$, de manière à ce que les décalages de positionnement coincident avec les décalages d'accumulation.

***Phase P=1:** 256^* décalages :

(5)



A la fin de cette phase, nous aurons :

$X_0 \dots \dots \dots X_{63}$ Dans CX1

$X_{64} \dots \dots \dots X_{127}$ Dans CX2

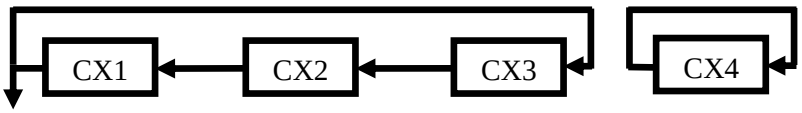
$X_{128} \dots \dots \dots X_{191}$ Dans CX3

$X_{192} \dots \dots \dots X_{255}$ Dans CX4

* : En fait seul 255 décalages sont nécessaires le $256^{ème}$ sera un décalage de positionnement.

***Phase P=2 :** 192 décalages :

(6)



Vers les MAC

A la fin de cette phase, nous aurons :

$X_0 \dots \dots \dots X_{63}$ Dans CX1

$X_{64} \dots \dots \dots X_{127}$ Dans CX2

$X_{128} \dots \dots \dots X_{191}$ Dans CX3

$X_{192} \dots \dots \dots X_{255}$ Dans CX4

Si nous avions gardé l'enchainement **(5)** pour cette phase, nous aurions eu au bout de ces 192 décalages.

$X_{192} \dots \dots \dots X_{255}$ Dans CX1

$X_0 \dots \dots \dots X_{63}$ Dans CX2

$X_{64} \dots \dots \dots X_{127}$ Dans CX3

$X_{128} \dots \dots \dots X_{191}$ Dans CX4

Et bien sûr, la phase P=3 n'aurait pas pu démarrer l'accumulation immédiatement après la phase P=2.

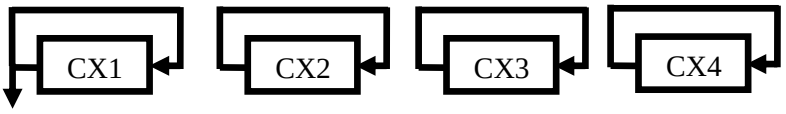
***Phase P=3 :** 128 décalages

(7)

Vers les MAC

***Phase P=4 :**

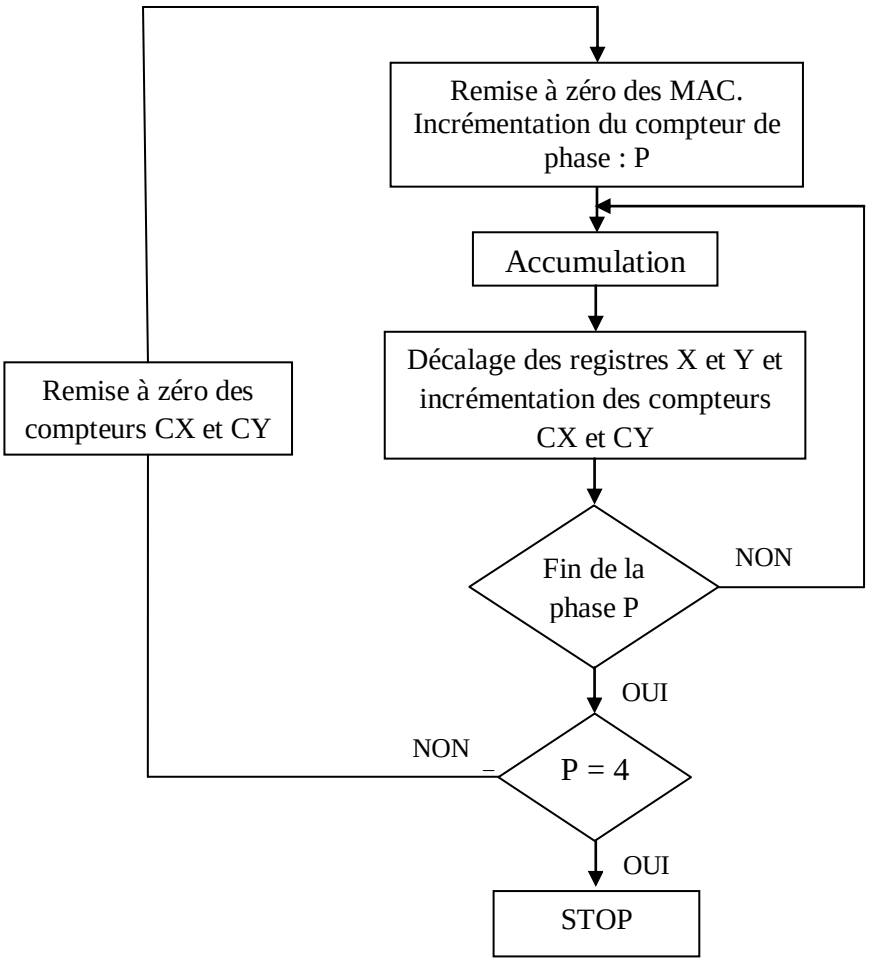
(8)



Vers les MAC

III.4)- Flow-chart :

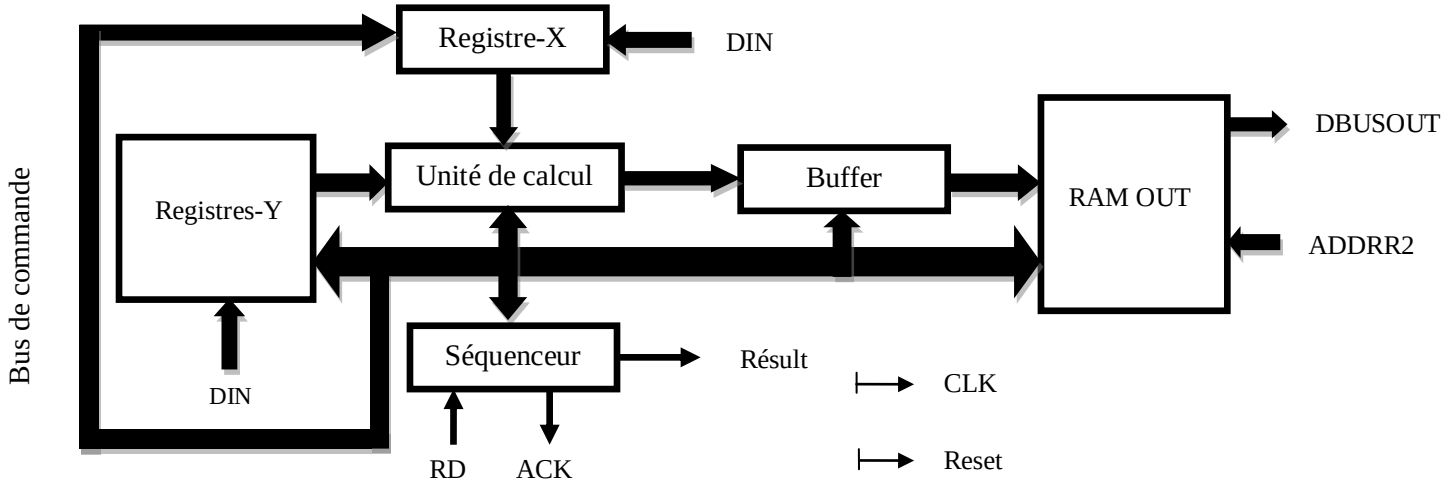
Les ressources matérielles ayant été établies dans leurs grandes lignes, nous pouvons établir le flow-chart, du moins une esquisse :



Figure(III.4) : *Organigramme globale du corrélateur*

III.5)- Organisation de la machine

Nous donnons à présent une description plus détaillée du chemin de données et du séquenceur servant à l'implémentation du schéma de calcul précédent.



Figure(III.5) : *Schéma de la machine de calcul*

Il s'agit d'une machine synchrone à horloge CLK monophasée.

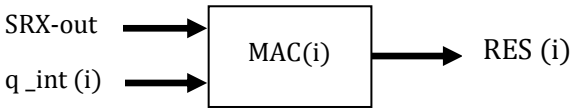
- **DIN** sert à l'introduction des vecteurs X et Y dans les registres à décalage.
- **L'unité de calcul** sert aux accumulations.
- **Le buffer** sert à un chargement parallèle des résultats d'accumulation en fin de phase avant leur stockage mot par mot en RAM.

Cette dernière est à double port. Elle fournit sur DBUSOUT le contenu du mot mémoire pointé par ADDR2.

- Les signaux RD et ACK servent à la lecture de X et Y.
- Le signal RESULT signale la fin du traitement et donc la disponibilité des résultats en RAM.

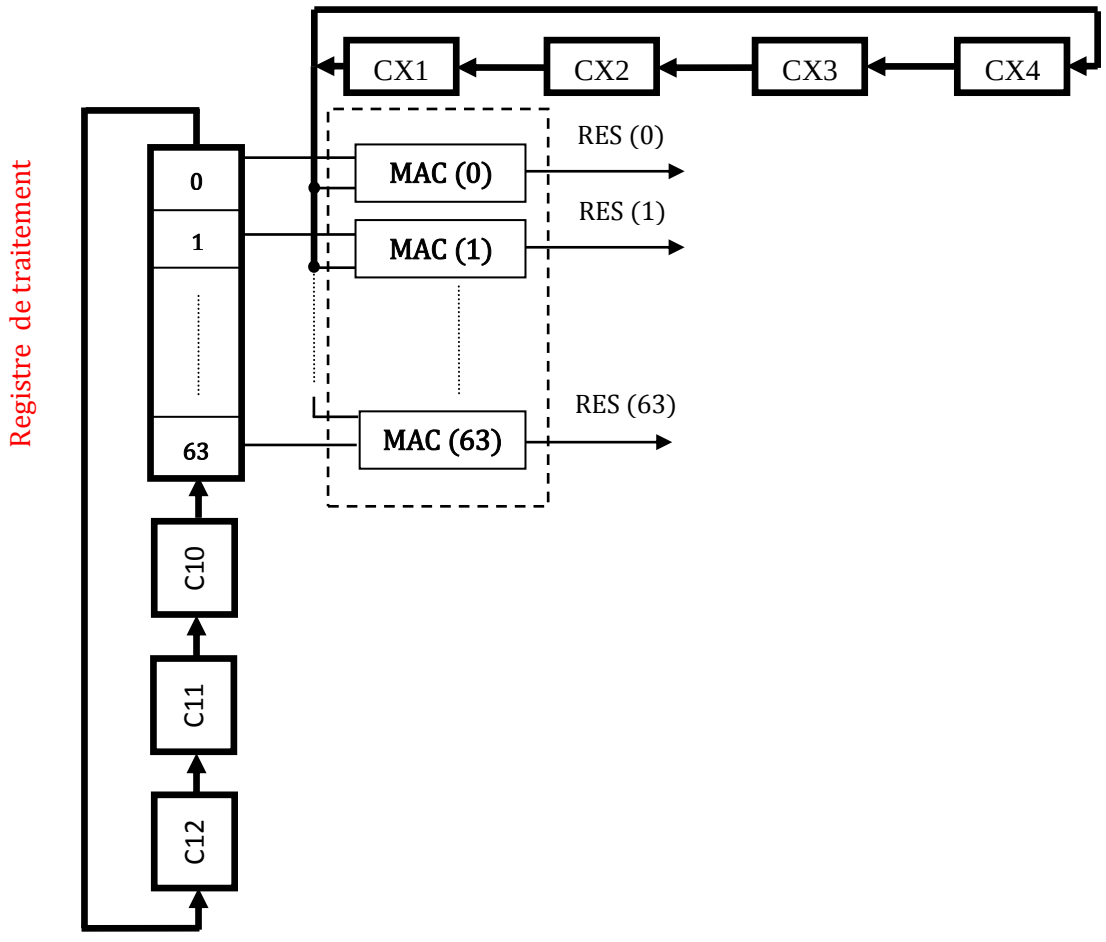
III.5.1)- Unité de calcul

Elle est constituée de 64 MAC numérotés de 0 à 63 :



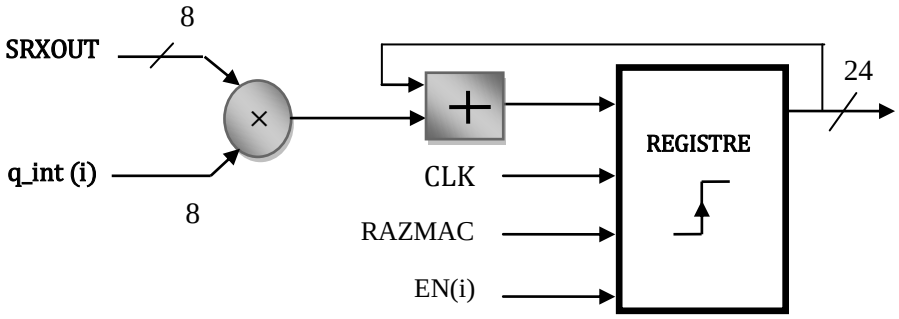
Le MAC d'indice *i* reçoit en entrée l'élément q_int(i) du registre de traitement, celui-ci fait partie du registre Y.

Sur la seconde entrée, nous injectons le signal SRXOUT en provenance du registre à décalage X



Figure(III.6) : *Schéma de calcul dans la 1ère phase*

La structure interne du MAC est :



Figure(III.7) : *Structure interne du MAC*

Compte tenu que les entrées sont à 8 bits et comme le nombre maximum d'addition est de 256, le vecteur de sortie doit être de taille au minimum égale à 24 bits.

Cette structure MAC utilise des multiplieurs 18 bits. L'ensemble est câblé (Embedded structure) dans le corps du FPGA. Elle ne nécessite aucune description VHDL sauf son invocation (component) une fois créee avec l'outil MEGAWIZARD figure (III.8), fonction ALMULT-ACCUM.

L'invocation:

```
COMPONENT MAC1
        PORT (
                ACLR0       : IN STD_LOGIC:= '0';
                CLOCK0      : IN STD_LOGIC:= '1';
                DATAA       : IN STD_LOGIC_VECTOR (7 DOWNTO 0):= (OTHERS => '0');
                DATAB       : IN STD_LOGIC_VECTOR (7 DOWNTO 0):= (OTHERS => '0');
                ENA0        : IN STD_LOGIC:= '1';
                RESULT      : OUT STD_LOGIC_VECTOR (23 DOWNTO 0));
END Component;
```

- **ACLR0 :** signal de RAZ du registre de sortie du MAC.
- **CLOCK0 :** entrée horloge.
- **DATAA, DATAB :** entrées données.
- **ENA0 :** signal de validation d'horloge.
- **RESULT :** signal de sortie.

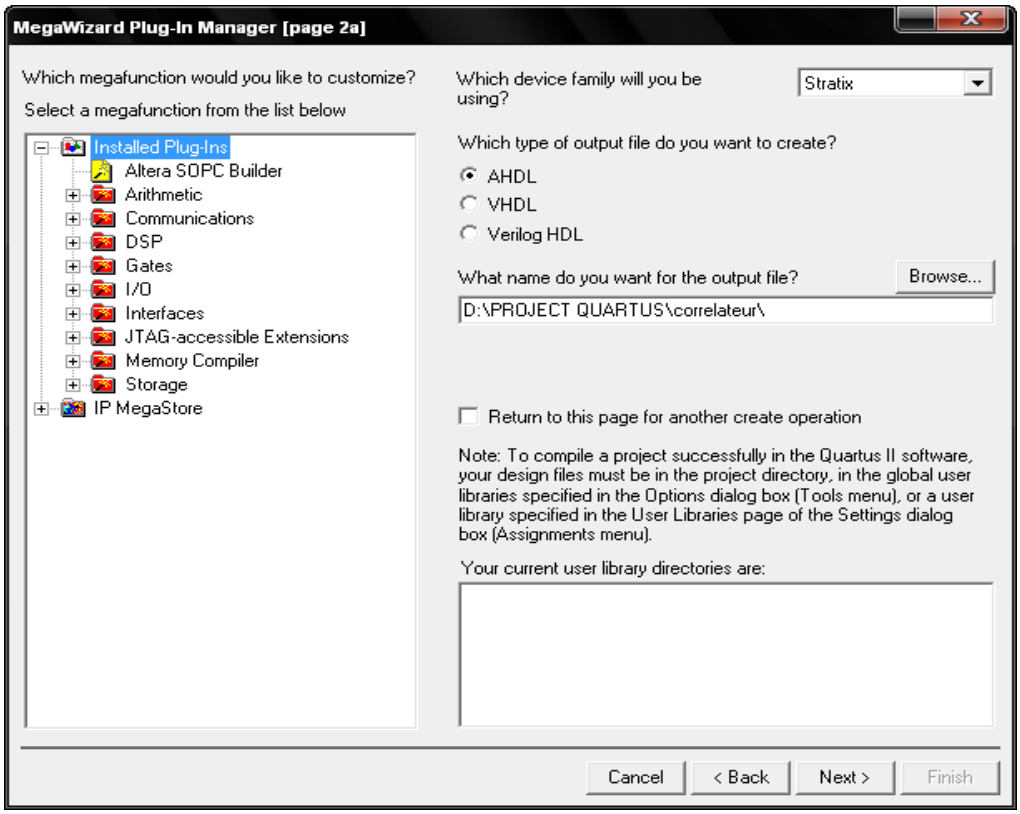


Figure (III.8) : *Instantiation d'IP dans QUARTUS : (MEGAWIZARD)*

Pour l'instantiation des MAC, plutôt que ceci soit fait élément par élément, nous avons utilisé l'instruction GENERATE pour créer l'ensemble des 64 MAC :

```
LES_MAC: FOR i IN 255 DOWNTO 192 GENERATE
MACS : mac1 PORT MAP (RAZMAC,CLK, SRXOUT,Q_INT (255-i),EN (255-i), RES (255-i));
End GENERATE;
```

De cette façon :

- **RAZMAC :** est l'équivalent du signal formel ACLR0. Il permet la remise à zéro du MAC à la fin de la phase P.
- **EN(i) :** est l'équivalent du signal formel.
- **ENA0 :** il permet ou non le chargement du résultat de l'accumulation au cours de l'étape P.
- **EN :** est un vecteur à 64 bits (1 bit pour chaque MAC).
- **RES (i) :** est l'équivalent du signal formel RESULT. Chaque élément du signal RES est à 24 bits, RES contient 64 éléments.

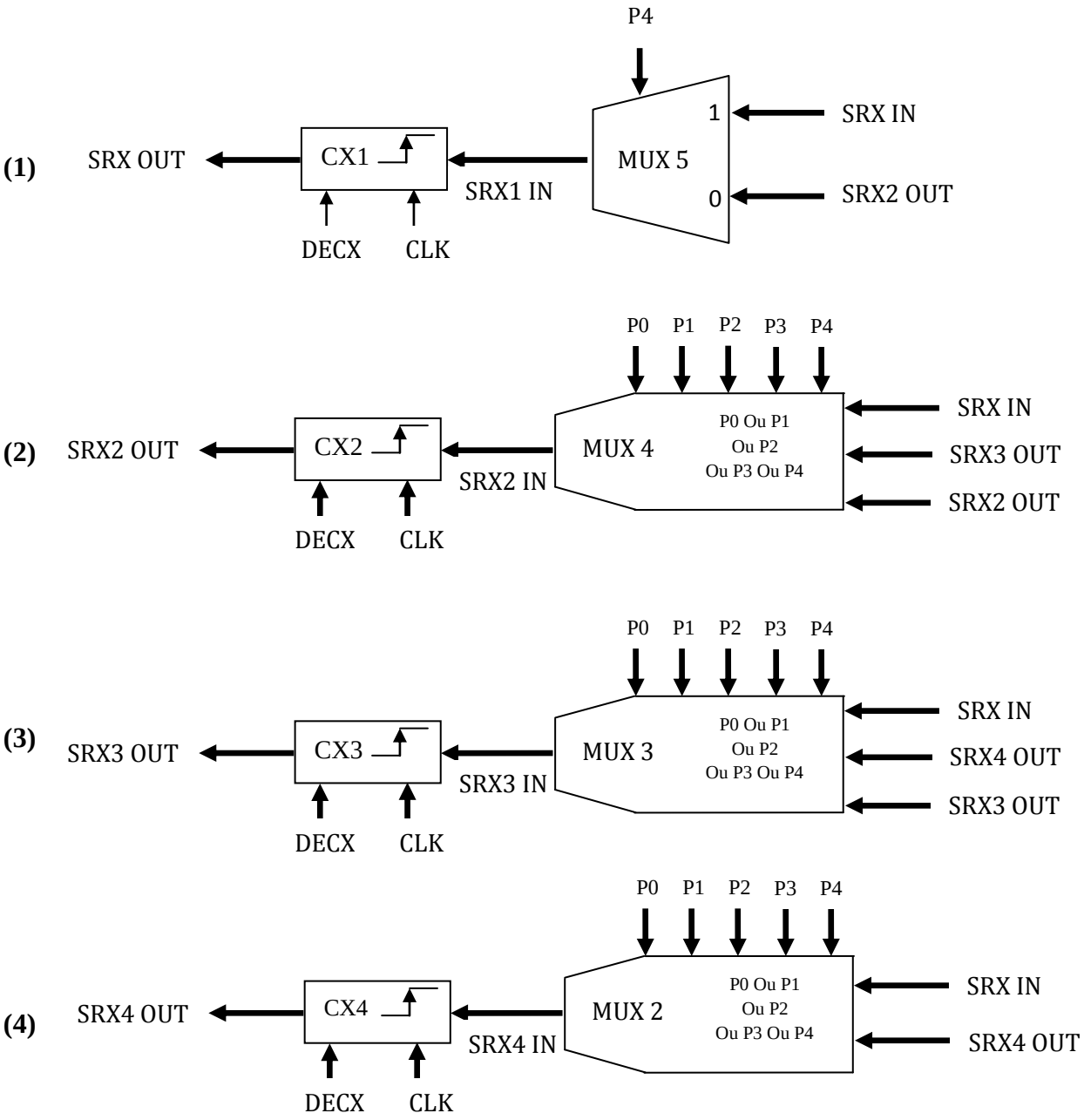
Cette structure MAC est assez rigide, ainsi nous ne pouvons pas par exemple spécifier le front actif d'horloge.

Ressource utilisée : (1 multiplieur 9 bit +24 lut +24 reg)/MAC.

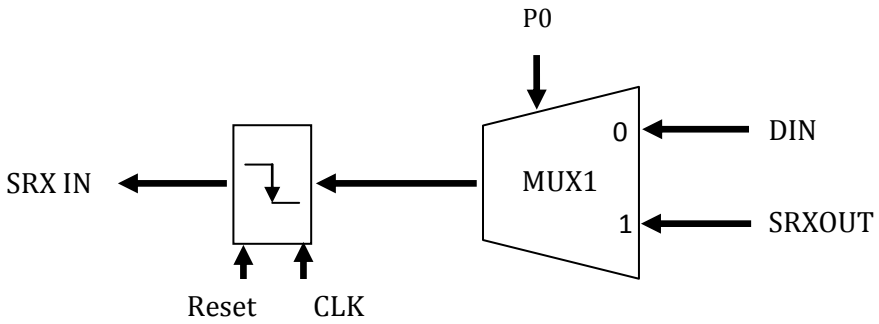
III.5.2)- Le registre X

Comme indiqué précédemment, cette structure comporte 4 registres à décalage 64×8 bits : CX1, CX2, CX3, CX4, ce qui permet de loger les 256 éléments du vecteurs X.

Sont inclus également 4 multiplexeurs combinatoires (MUX2, MUX3, MUX4, MUX5) et un multiplexeur synchronisé par l'horloge (MUX1).



(5)



P_0, P_1, P_2, P_3, P_4 sont les bits du signal P (signal de phase). Le bit 0 (P_0) est associé à la phase de lecture des données. P_1, P_2, P_3 et P_4 sont associés aux quatre phases du traitement.

Cet ensemble de multiplexeurs va nous permettre de réaliser les enchainements adaptés à chaque phase tel que nous l'avons indiqué en **(III.3)**.

Les décalages sont activés par le front montant d'horloge et validés par le signal DECX commun à tous ces registres.

Ces 4 registres sont implémentés à partir de la fonction SHIFT REGISTER (RAM BASED) de l'outil MEGAWIZARD. Comme le nom de la fonction l'indique, ces registres sont réalisés à partir de modules RAM M4K et ne nécessitent aucune description VHDL hormis l'invocation de composant : SRX.

Figure(III.9) : *Registre à décalage X (MEGAWIZARD)*

L'invocation:

```
Component SRX
        PORT (
                ACLR            : IN STD_LOGIC:= '1';
                CLKEN           : IN STD_LOGIC:= '1';
                CLOCK           : IN STD_LOGIC;
                SHIFTIN         : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                SHIFTOUT        : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
                TAPS            : OUT STD_LOGIC_VECTOR (63 DOWNTO 0));
End component;
```

- **SHIFT IN :** entrée données (1 octet)
- **SHIFT OUT :** sortie données (1 octet)
- **CLOCK :** l'horloge
- **CLKEN :** validation d'horloge
- **ACLR :** remise à zéro asynchrone.

Les ressources utilisées pour chaque SRX :

- 2 M4K*
- 6 LUT (Look -Up-Table)
- 7 registres

En ce qui concerne l'instantiation de ces registres à partir de SRX, nous donnons l'exemple du registre CX1 :

CX1: SRX **PORT MAP** (RESET, DECX, CLK, SRX1 IN, SRXOUT, OPEN);

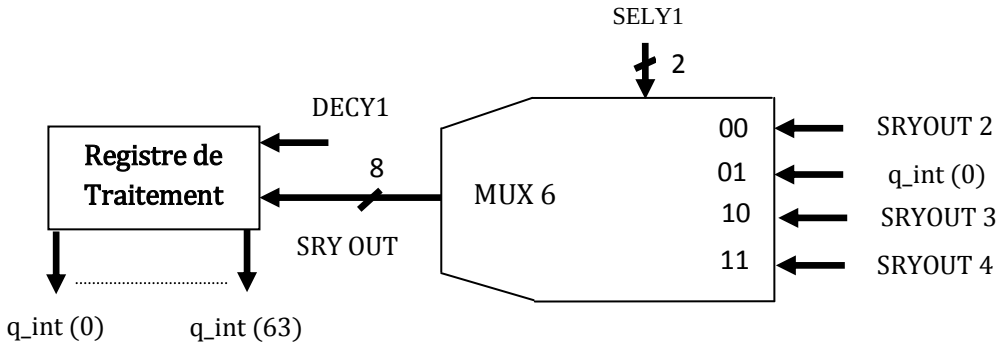
Comme vous le constatez, la sortie TAPS est laissée ouverte. En principe, cette sortie doit nous permettre d'accéder à tout moment au contenu interne du registre.

*M4K bloc de mémoire RAM de taille 128 mots de 36 bits soit 4608 bits.

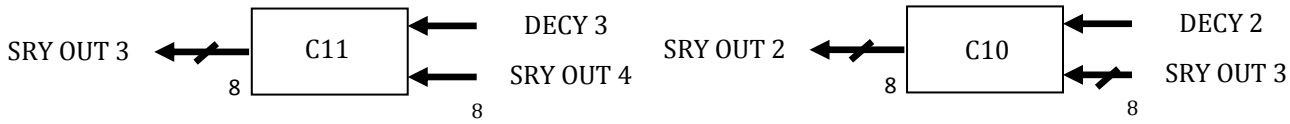
Mais en vérité, la documentation nous indique que ces éléments du registre ne sont accessibles qu'à des intervalles distants au minimum de 3 positions. Par exemple : élément 1, élément 4, élément 7,… ajoutons à celà le fait que le choix du front actif d'horloge n'est pas possible.

III.5.3)- Le registre Y

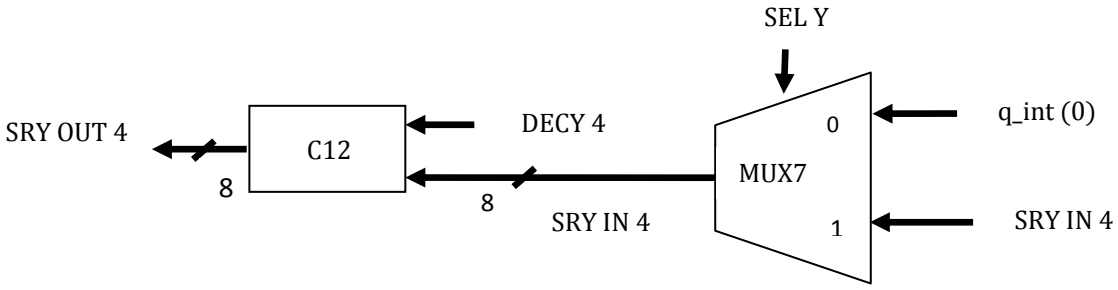
(6)



(7)



(8)



Le registre Y est constitué:

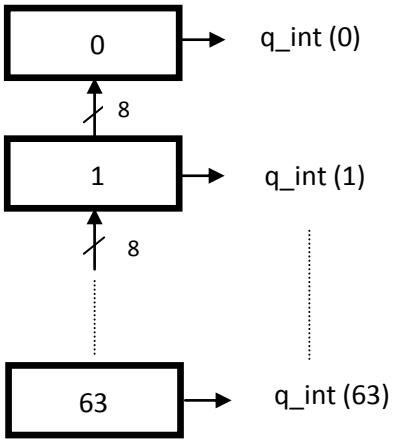
- Le registre de traitement RT.
- Les registres à décalage C10, C11, C12
- 2 multiplexeurs.

a- Le registre de traitement :

C'est un registre à décalage constitué de 64×8 registres de type bascule D.

Les décalages sont validés par DECY1 et activé par l'horloge.

CLK
RESET
DECY1
} Connectés aux 64 registres



Figure(III.10) : *Registre de traitement*

Ce registre va présenter en permanence 64 éléments du vecteur Y (q-int(0),…, q-int(63)) en entrée des MAC.

L'implémentation de cette structure en RAM n'est pas possible puisque comme nous l'avons indiqué précédemment, nous ne pouvons pas dans ce cas accéder simultanément aux 64 éléments.

Le front montant d'horloge active les décalages validés par DECY1.

Pour la description VHDL, nous avons utilisé une boucle FOR…. END LOOP.

b- Les registres à décalage C10, C11, C12 :

Ces 3 registres 64×8 bits, associés au registre de traitement, permettent de loger les 256 éléments du vecteur Y.

Ces 3 registres sont identiques aux registres CX1, CX2, CX3, CX4. Ils sont donc implémentés en RAM, sauf qu'en la circonstance, le composant a été désigné par SRY.

c- Les multiplexeurs :

Cet ensemble de multiplexeurs vont permettre des enchainements de registres adaptés à chaque phase tel que nous l'avons indiqué en **(III.3)**.

Les signaux DECY1, DECY2, DECY3, DECY4, SELY1 et SELY sont envoyés par le séquenceur au travers du bus de commande.

c.1- MUX6 :

Ce multiplexeur va permettre d'organiser le tir vers le registre de traitement au moyen du signal de sélection SELY1, fournir par le séquenceur.

Il permet donc d'injecter dans RT soit la sortie du registre C10, soit celle de C11 soit celle de C12 soit le signal q-int(0). Dans ce dernier cas, le registre de traitement réalise un décalage par rotation.

c.2- MUX7 :

Ce multiplexeur va permettre d'injecter dans le registre Y (C12) soit le signal d'entrée de données (DIN) en phase de lecture (P_0=1) soit le signal q-int(0) par toutes les autres phases, le signal sélection est SELY en provenance du séquenceur.

III.5.4)- Le buffer :

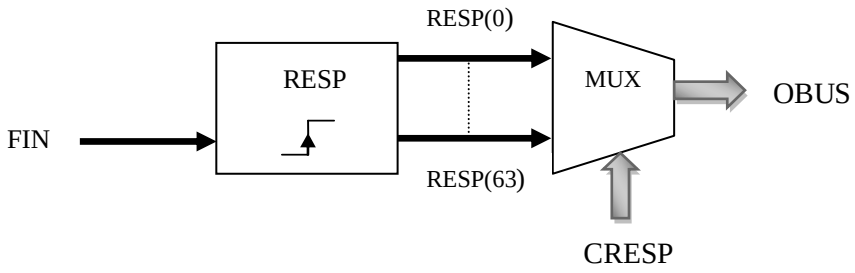
Ce registre est constituée de 64 registres 24 bits de type bascule D. nous y stockons à la fin de chaque phase 1, 2, 3 et 4 (front montant du signal FIN) les résultats de cette phase présents en sortie des MAC (signal RES).

L'identification de ce buffer est faite au moyen du signal RESP.

Dés que la sauvegarde a été effectuée (transfert de RES vers RESP), la phase suivante peut démarrer sans craindre l'écrasement des résultats. D'où un gain de temps appréciable.

Le contenu du buffer est ensuite envoyé vers la RAM de sortie mot par mot à chaque impulsion d'horloge.

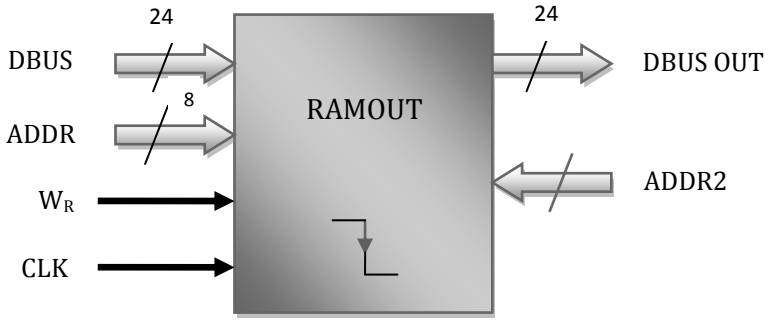
En sortie de ce buffer, nous avons donc inséré un multiplexeur qui ; au moyen du signal de sélection CRESP ; va déposer l'un des mots de RESP dans le bus DBUS connecté à la RAM de sortie.



Figure(III.11) : *schéma du Buffer*

III.5.5)- La mémoire résultats : RAMOUT

Il s'agit d'une mémoire RAM à double port contenant 256 mots de 24 bits et nécessitant 2 M4K.



Figure(III.12) : *Mémoire à double port*

L'écriture en RAMOUT :

Le signal W_R doit être positionné à 1. L'adresse d'écriture est spécifiée par le séquenceur au moyen du signal ADDR. Le contenu de DBUS est alors inscrit en mémoire sur front descendant de l'horloge.

La lecture :

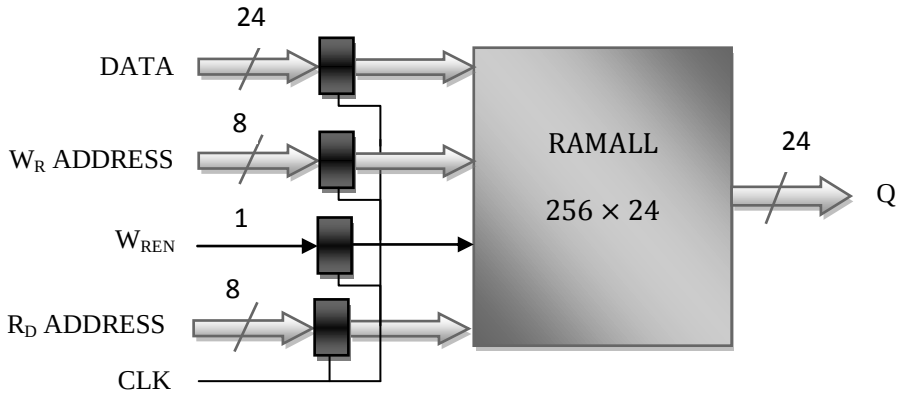
Le signal W_R doit être positionné à 0, l'adresse de lecture est spécifiée par le périphérique au moyen du signal ADDR2.

Le contenu du mot mémoire pointé par ADDR2 est alors déposé dans DBUSOUT (signal de sortie) sur front descendant de l'horloge.

L'implémentation de cette RAM est obtenue au moyen de la fonction RAM : 2-port de l'outil MEGAWIZARD.

L'invocation:

```
Component RAMALL
        PORT (
                CLOCK           : IN STD_LOGIC;
                DATA            : IN STD_LOGIC_VECTOR (23 DOWNTO 0);
                RDADDRESS       : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                WRADDRESS       : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                WREN            : IN STD_LOGIC:= '1';
                Q               : OUT STD_LOGIC_VECTOR (23 DOWNTO 0));
End COMPONENT;
```



Figure(III.13) : *Mémoire à double port (MEGAWIZARD)*

Comme nous le constatons, toutes les entrées sont à registres activés par l'horloge.

L'instantation :

Elle est relativement simple puisqu'il s'agit d'un seul composant :

RAMOUT: RAMALL **PORT MAP** (CLK, DBUS, ADDR2, ADDR, W_R, DBUSOUT);

Le signal W_R est fourni par le séquenceur.

III.6)- Le séquenceur :

Il s'agit assurément du cœur de la machine.

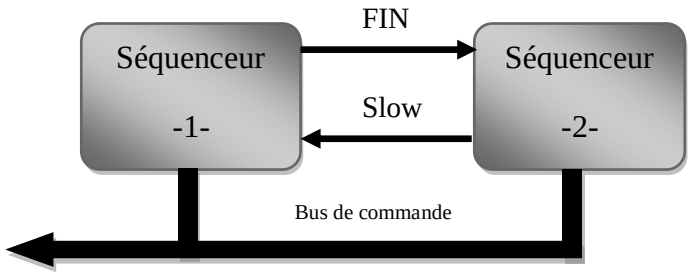
Etant donné que nous devons implémenter deux tâches concurrentes, à savoir :

- Lecture des données X, Y et calculs.
- Sauvegarde des résultats.

Nous devons donc scinder notre séquenceur en deux :

- Séquenceur 1.
- Séquenceur 2.

Des signaux de synchronisation : FIN et SLOW permettront d'assurer la synchronisation entre ces 2 séquenceurs.



Contrôle du DATAPATH

Le signal FIN permet au séquenceur 1 d'indiquer la fin d'exécution d'une phase.

Le séquenceur 2 peut alors entreprendre la sauvegarde des résultats de la phase qui vient de s'achever.

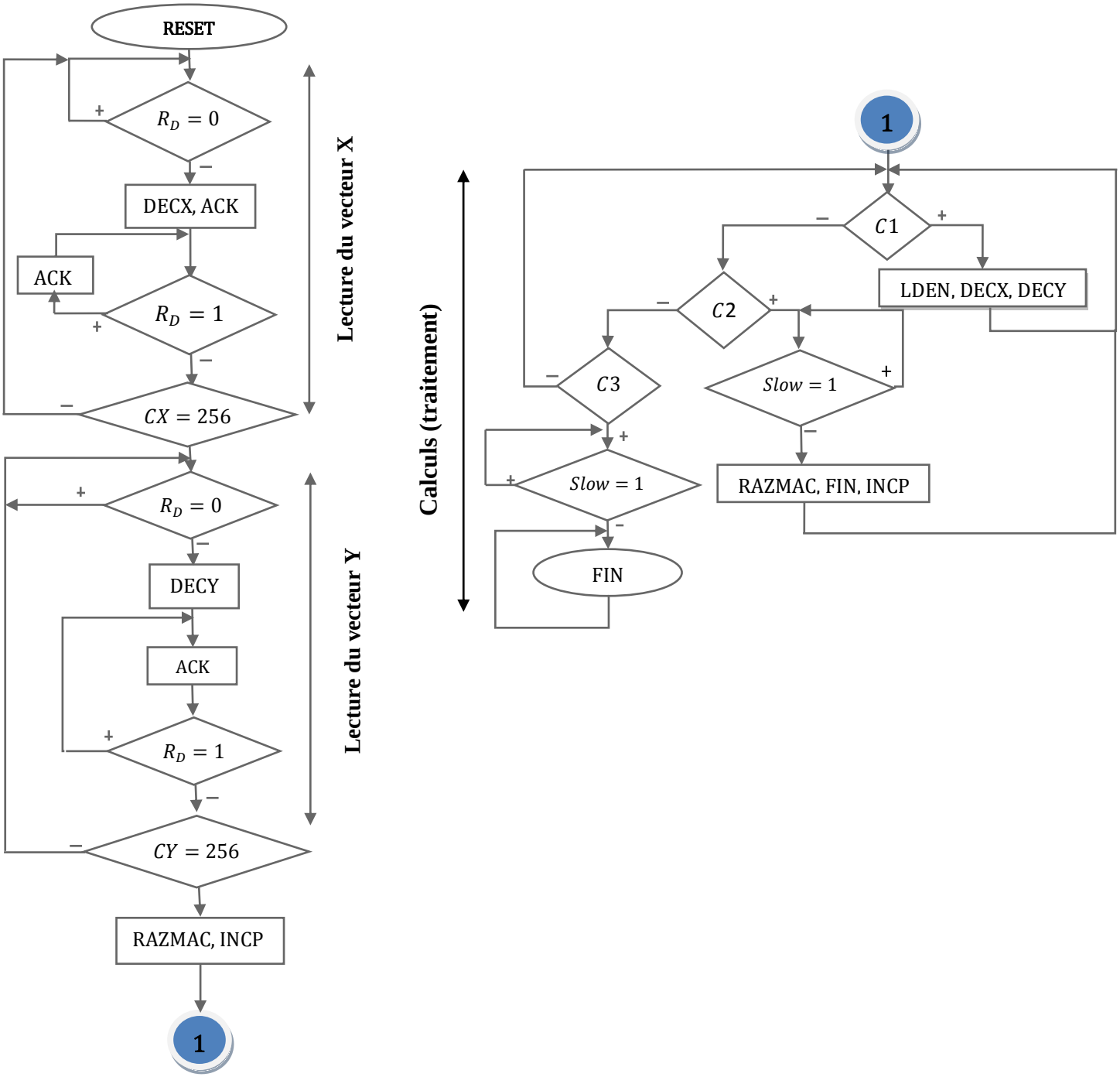
Le signal SLOW permet au séquenceur 2 d'indiquer qu'une sauvegarde est en cours d'exécution. Le séquenceur 1 doit donc suspendre son exécution afin d'éviter un écrasement des résultats de la phase précédente.

Les 2 séquenceurs sont bâtis autour de machines d'états (FSM1, FSM2). Ces deux machines d'états sont réalisées en logique structurée de type PLA.

Autour d'elles, nous avons disposé de la logique anarchique pour implémenter des compteurs, des multiplexeurs,….. Les 2 machines d'états sont activées par le front descendant de l'horloge, leurs états sont codés.

III.6.1)- **Séquenceur 1** :

Nous donnons ci-dessus le flow-chart (ou algorithme) de FSM1

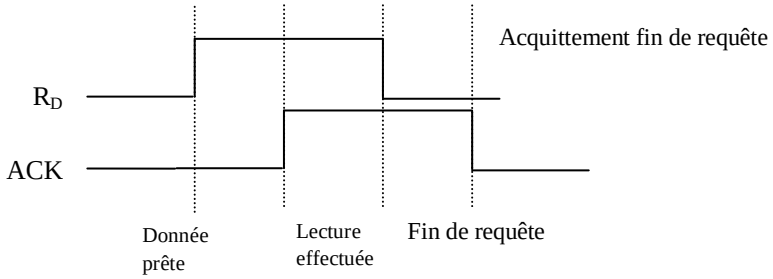


Figure(III.14) : *Algorithme FSM1*

Les signaux de condition C1, C2 et C3 sont donnés par :

$\mathbf{C1} = (\mathrm{Cy} \neq 256 \ \text{ et } \mathrm{P} = 1)$ ou $(\mathrm{Cy} \neq 192 \ \text{ et } \mathrm{P} = 2)$ ou $(\mathrm{Cy} \neq 128 \ \text{ et } \mathrm{P} = 3)$ ou $(\mathrm{Cy} \neq 64 \ \text{ et } \ \mathrm{P} = 4)$	$\mathbf{C2} = (\mathrm{Cy} = 256 \ \text{ et } \mathrm{P} = 1)$ ou $(\mathrm{Cy} = 192 \ \text{ et } \mathrm{P} = 2)$ ou $(\mathrm{Cy} = 128 \ \text{ et } \mathrm{P} = 3)$ $\mathbf{C3} = (\mathrm{Cy} = 64 \ \text{ et } \mathrm{P} = 4)$

La lecture des vecteurs X et Y est activée par le signal d'entrée R_D et acquittée par le signal de sortie ACK.



Figure(III.15) : *protocole de lecture et acquittement des vecteurs X et Y par HANDSHAKING*

Le protocole est certes lent, mais il a le mérite de ne pas nécessiter de composants spécifiques tel qu'une FIFO (synchronisation FIFO).

L'observation des signaux en simulation a montré que le délai minimum entre le front descendant de l'horloge CLK et la mise à 1 de ACK est de 14.8 ηs. Pour la mise à 0 de ACK, ce délai est de 5 ηs. Pour une horloge cadencée à 10 ηs, ceci signifie qu'il faut au moins 4 impulsions d'horloge entre 2 requêtes R_D.

Pour une période d'horloge de 10 ηs, l'introduction des donnés réclame # 20 μs.

Ceci est certes excessif en regard de la durée du traitement qui est de 8.5 μs.

Un travail supplémentaire est nécessaire afin de diminuer la durée de lecture des données X et Y, cette lecture des données X et Y se termine quand les 2 compteurs CX et CY sont positionnés à 256.

Ces compteurs sont alors remis à 0 au moyen de RAZMAC tout en incrémentant le compteur de phase au moyen du signal INCP pour passer à la phase suivante.

Rappel :

La lecture correspond à P=0.

Le traitement P=1, 2, 3 et 4.

Le traitement est constitué ; pour l'essentiel ; d'une succession de décalages simultanés des registres X et Y activés par les signaux DEC et DECY. Chaque décalage donne lieu à une accumulation du résultat de (multiplication, addition) dans les MAC.

Cette accumulation est conditionnée par le signal LDEN ; la phase 1 (P=1 ou P1) se termine quand 256 décalages auront été effectués. Quant aux phases 2, 3 et 4 elles réclament 192, 128 et 64 décalages respectivement.

A la fin de chaque phase, et si le signal SLOW est à l'état bas, alors le signal FIN reçoit une impulsion, les MAC et les compteurs CX et CY sont remis à zéro (RAZMAC) et le compteur de phase est de nouveau incrémenté (INCP) pour passer à la phase suivante. Par contre, si le signal SLOW est à l'état haut, la machine boucle jusqu'à ce que la condition SLOW=0 soit vérifiée.

Les signaux de condition C1, C2 et C3 que nous avons explicité précédemment correspondent aux situations suivantes :

C1 : phase 1, 2 ,3 ou 4 en cours.

C2 : phase 1 ou phase 2 ou phase 3 terminée.

C4 : phase 4 terminée.

Le flow-chart se traduit par le tableau des états donné ci-dessous :

ENTREES	E_P	E_S	SORTIES	R
R_D =0	**0**	**0**		**1**
R_D =1	**0**	**1**		**2**
	1	**2**	DECX, ACK	**3**
R_D =1	**2**	**2**	ACK	**4**
R_D =0 CX ≠ 256	**2**	**0**		**5**
R_D =0 CX = 256	**2**	**3**		**6**
R_D =0	**3**	**3**		**7**
R_D =1	**3**	**5**	DECY	**8**
R_D =1	**5**	**5**	ACK	**9**
R_D =0 CY ≠ 256	**5**	**3**		**10**
R_D =0 CY = 256	**5**	**6**	RAZMAC, INCP	**11**
C1	**6**	**6**	LDEN, DECX, DECY	**12**
C2	**6**	**7**		**13**
C3	**6**	**8**		**14**
SLOW = 0	**7**	**6**	RAZMAC, FIN, INCP	**15**
SLOW = 0	**8**	**8**	FIN	**16**
SLOW = 1	**7**	**7**		**17**
SLOW = 1	**8**	**8**		**18**

Tableau (III.1) : *Tableau des états du deuxième séquenceur FSM1*

III.6.1.1)- Les équations logiques

Nous en déduisons le système d'équations logiques suivantes :

Plan ET du PLA :

$\boldsymbol{R_1}= \overline{R_D}\ .\overline{e_3}.\ \overline{e_2}.\ \overline{e_1}.\ \overline{e_0}$

$\boldsymbol{R_2} = R_D\ .\overline{e_3}.\ \overline{e_2}.\ \overline{e_1}.\ \overline{e_0}$

$\boldsymbol{R_3} = \overline{e_3}.\ \overline{e_2}.\ \overline{e_1}.\ e_0$

$\boldsymbol{R_4}=R_D\ .\overline{e_3}.\ \overline{e_2}.\ e_1.\ \overline{e_0}$

$\boldsymbol{R_5}=\overline{R_D}\ .\ \overline{CX_8}\ .\overline{e_3}.\ \overline{e_2}.\ e_1.\ \overline{e_0}$

$\boldsymbol{R_6} =\overline{R_D}\ .\ CX_8\ .\overline{e_3}.\ \overline{e_2}.\ e_1.\ \overline{e_0}$

$\boldsymbol{R_7} =\overline{R_D}\ .\overline{e_3}.\ \overline{e_2}.\ e_1.\ e_0$

$\boldsymbol{R_8}=R_D\ .\overline{e_3}.\ \overline{e_2}.\ e_1.\ e_0$

$\boldsymbol{R_9}=R_D\ .\overline{e_3}.\ e_2.\ \overline{e_1}.\ e_0$

$\boldsymbol{R_{10}} = \overline{R_D}\ .\ \overline{CY_8}\ .\overline{e_3}.\ e_2.\ \overline{e_1}.\ e_0$

$\boldsymbol{R_{11}}=\overline{R_D}\ .\ CY_8\ .\overline{e_3}.\ e_2.\ \overline{e_1}.\ e_0$

$\boldsymbol{R_{12}} = \left[\left(\overline{CY_8}.P_1\right) + \left(\overline{(CY_7.CY_6)}.P_2\right) + \left(\overline{CY_7}.P_3\right) + \left(\overline{CY_6}.P_4\right)\right].\ (\overline{e_3}.e_2.\ e_1.\ \overline{e_0})$

$\boldsymbol{R_{13}} = [(CY_8.P_1) + (CY_7.CY_6.P_2) + (CY_7.P_3)].\ (\overline{e_3}.e_2.\ e_1.\ \overline{e_0})$

$\boldsymbol{R_{14}} =\ CY_6.P_4.\overline{e_3}.e_2.\ e_1.\ \overline{e_0}$

$\boldsymbol{R_{15}} =\overline{SLOW}\ .\overline{e_3}.\ e_2.\ e_1.\ e_0$

$\boldsymbol{R_{16}} =\overline{SLOW}\ .e_3.\ \overline{e_2}.\ \overline{e_1}.\ \overline{e_0}$

$\boldsymbol{R_{17}} =SLOW\ .\overline{e_3}.\ e_2.\ e_1.\ e_0$

$\boldsymbol{R_{18}} =SLOW\ .e_3.\ \overline{e_2}.\ \overline{e_1}.\ \overline{e_0}$

PLAN OU du PLA :

$\boldsymbol{e_3} = R_{15} + R_{17} + R_{19}$

$\boldsymbol{e_2} = R_9 + R_{10} + R_{12} + R_{13} + R_{14} + R_{16} + R_{18}$

$\boldsymbol{e_1} = R_3 + R_4 + R_6 + R_7 + R_{11} + R_{12} + R_{13} + R_{14} + R_{16} + R_{18}$

$\boldsymbol{e_0} = R_2 + R_6 + R_7 + R_9 + R_{10} + R_{11} + R_{14} + R_{18}$

$\boldsymbol{DECX} = R_3 + R_{13}$

$\boldsymbol{DECY} = R_9 + R_{13}$

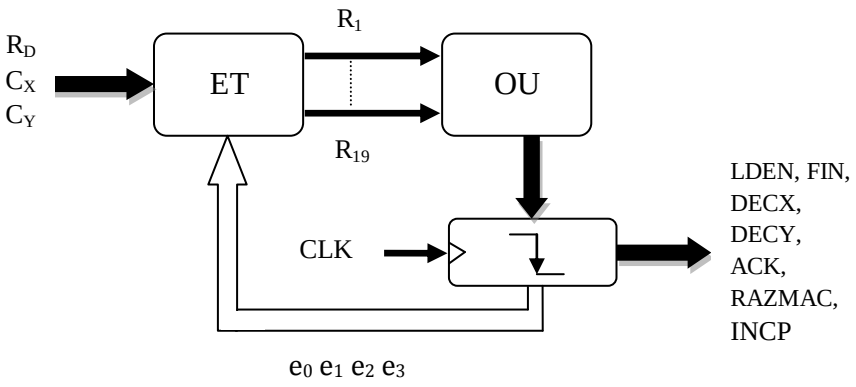
$\boldsymbol{ACK} = R_3 + R_4 + R_{10}$

$\boldsymbol{RAZMAC} = R_{12} + R_{16}$

$\boldsymbol{LDEN} = R_{13}$

$\boldsymbol{FIN} = R_{16} + R_{17}$

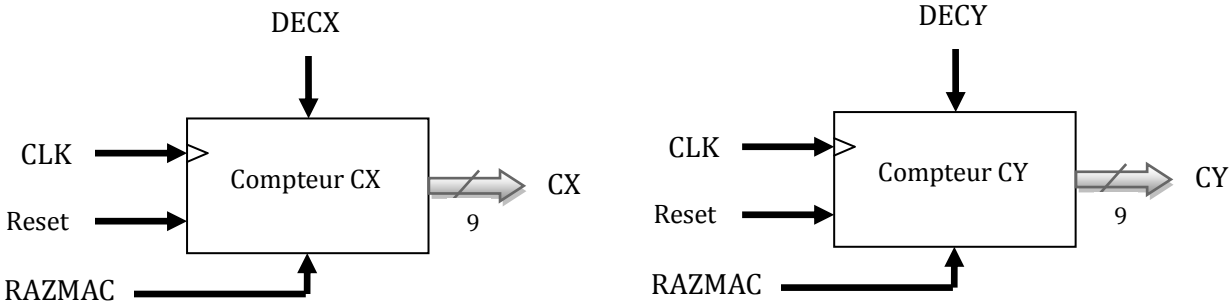
$\boldsymbol{INCP} = R_{16} + R_{12}$



Figure(III.16) : *schéma du plan (ET/OU) du séquenceur FSM1*

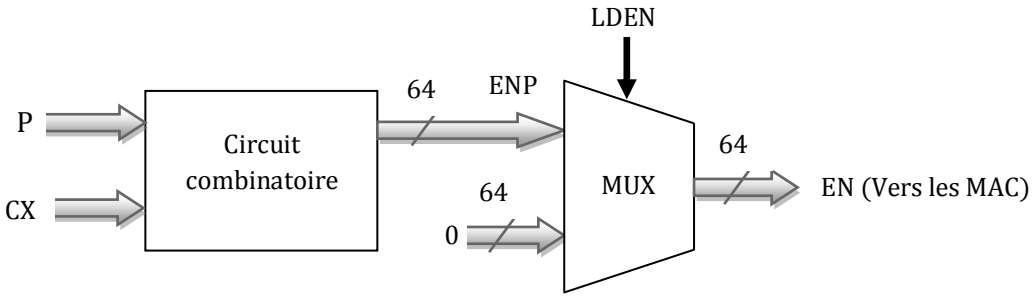
Le séquenceur 1 contient également des éléments de logique anarchique à savoir :

- Compteurs CX et CY :
  - **Remise à zéro :** au moyen des signaux RESET ou RAZMAC, ces 2 signaux agissent de façon asynchrone.
  - **L'incrémentation** (+1) est validée par les signaux DECX et DECY puisqu'il s'agit de compter le nombre de décalages effectués par les registres X et Y. Cette incrémentation est activée par le front montant de l'horloge CLK.
  - **Taille des signaux CX et CY** : 9 bits.



Figure(III.17) : *compteurs CX et CY*

- **Circuit de validation d'accumulation :** il comporte une partie combinatoire suivie d'un multiplexeur :



La partie combinatoire est chargée d'effectuer les tests suivant le schéma de calcul que nous avons donné en **(III.1)** :

À savoir : pour $\mathbf{0 \leq i \leq 63}$

```
Si      (P="1" . CX > 192 . I+CX > 255)
   OU (P="2" . CX > 128 . I+CX+64 > 255)
   OU (P="3" . CX > 64  . I+CX+128 > 255)
   OU (P="4" . CX > 0   . I+CX+192 > 255)
   ALORS        ENP (I) = 0;
SINON           ENP (I) = 1;
```

De ce fait, chaque bit de EN étant associé à l'entrée de validation d'horloge du MAC de même indice, celui-ci sera autorisé donc à effectuer l'accumulation ou la bloquer ;

EN(i) =1 ; Accumulation dans MAC(i).

EN(i) =0 ; blocage.

- **Autres circuits combinatoires :**

Ces circuits détectent l'occurrence des valeurs 0, 64, 128, 192 et 256 en sortie des compteurs CX et CY. Ils positionnent alors à 1 l'un des signaux CX0, CX64, CX128, CX192, CX256, CY0, CY64, CY128, CY192, CY256.

Un second groupe de circuits combinatoires est chargé de générer les signaux DECY1, DECY2, DECY3, DECY4, SELY et SELY1 utilisés par le contrôle du registre Y : (DECY provient FSM1).

DEC1 et **DECY**4 : 1 **si** DECY = 1 , 0 autrement

DECY2: $\begin{cases} 1 & \textbf{si } \textbf{DECY} = 1 \text{ et } \textbf{P} = 0,1,4 \\ 0 & \text{autrement} \end{cases}$

DECY3: $\begin{cases} 1 & \textbf{si } \textbf{DECY} = 1 \text{ et } \textbf{P} = 0,1,2,4 \\ 0 & \text{autrement} \end{cases}$

SELY: $\begin{cases} 1 & \textbf{si } \textbf{P} = 1 \\ 0 & \text{autrement} \end{cases}$

SELY1 (2bits)

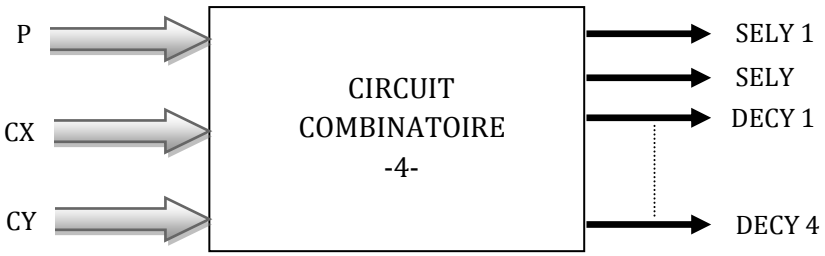
SELY1 =

00 **si** (**P**=0 ou (**P**=1 et **CY** < 192))

10 **si** ((**P**=1 and **CY** ≥ 192) ou (**P**=2 et **CY** <128))

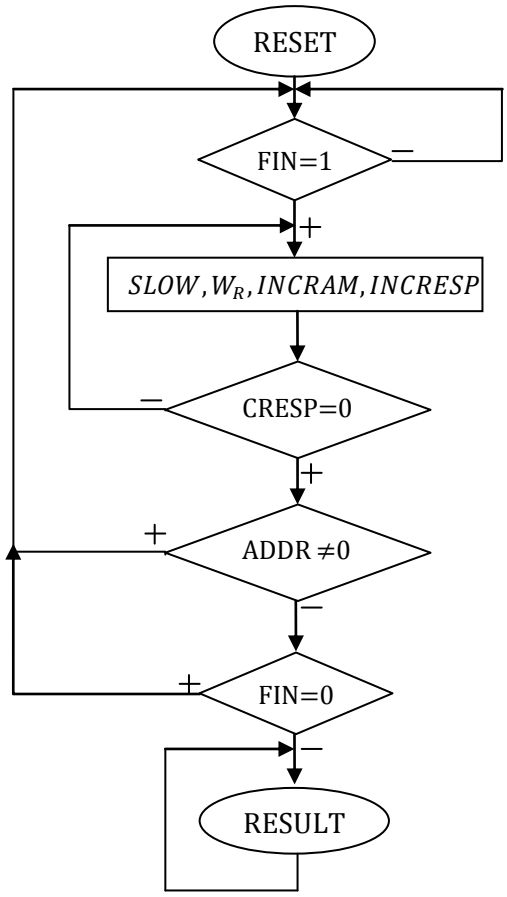
11 **si** ((**P**=2 et **CY** ≥ 128) ou (**P**=3 et **CY** <64))

01 **si** **P**=4 ;



III.6.2)- Séquenceur 2 :

Voici le flow-chart de FSM2 :



Figure(III.18) : *Algorithme FSM2*

Le but de ce flow-chart est :

- Transférer le contenu du buffer dans RAMOUT.
- Indiquer au séquenceur 1 que ce transfert est en cours (signal SLOW)
- Indiquer au périphérique externe la disponibilité de l'ensemble des résultats dans RAMOUT (signal de sortie : RESULT mis à 1).

Une impulsion (durée d'horloge) du signal FIN en provenance du séquenceur 1 indique au séquenceur 2 la fin d'une étape. En même temps, cette impulsion (son front montant) à permis de sauvegarder le contenu des MAC dans le buffer.

Cette impulsion permet au séquenceur 2 de transférer, mot par mot, le contenu du buffer dans la RAM par incrémentation des signaux CRESP et ADDR.

L'opération est répétée jusqu'à ce que le compteur d'adresse se remet à 0.

Le signal RESULT est alors positionné à 1 et la machine boucle sur cette étape.

Ce flow-chart se traduit par le tableau des états suivant :

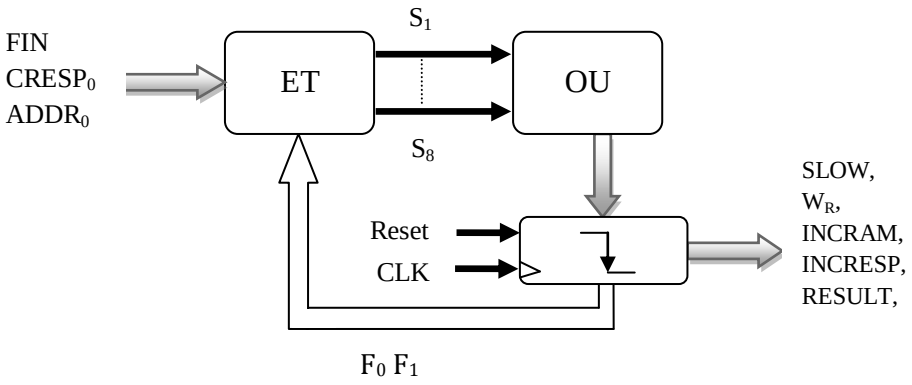
ENTREES	**E_P**	**E_S**	**SORTIES**	**R**
FIN =0	**0**	**0**		**1**
FIN=1	**0**	**1**		**2**
	1	**2**	SLOW,W_R, INCRAM, INCRESP	**3**
CRESP ≠ 0	**2**	**1**	SLOW	**4**
CRESP =0 ADDR ≠ 0	**2**	**0**		**5**
CRESP =0 ADDR = 0	**2**	**3**		**6**
FIN=1	**3**	**3**	RESULT	**7**
FIN =0	**3**	**0**		**8**

Tableau (III.2) : *Tableau des états du deuxième séquenceur FSM2*

III.6.2.1)- Les équations logiques

Nous en déduirons le système d'équations logiques suivantes :

PLAN ET :	**PLAN OU :**
$\boldsymbol{S_1}=\overline{FIN}\ .\overline{F_1}.\overline{F_0}$	$\boldsymbol{F_0} = S_2 + S_4 + S_6 + S_7$
$\boldsymbol{S_2}=FIN.\overline{F_1}.\overline{F_0}$	$\boldsymbol{F_1} = S_3 + S_6 + S_7$
$\boldsymbol{S_3}=\overline{F_1}.F_0$	$\boldsymbol{RESULT} = S_7$
$\boldsymbol{S_4}=\overline{CRESP_0}\ .F_1.\overline{F_0}$	$\boldsymbol{W_R} = S_3$
$\boldsymbol{S_5}=CRESP_0.\overline{ADDR_0}\ .F_1\ .\overline{F_0}$	$\boldsymbol{INCRAM} = S_3$
$\boldsymbol{S_6}=CRESP_0.\ ADDR_0\ .F_1\ .\overline{F_0}$	$\boldsymbol{INCRESP} = S_3$
$\boldsymbol{S_7}=FIN\ .F_1\ .F_0.$	$\boldsymbol{SLOW} = S_3 + S_4$
$\boldsymbol{S_8}=\overline{FIN}\ .F_1\ .F_0.$	



Figure(III.19) : *schéma du plan (ET/OU) du séquenceur FSM2*

Il s'agit, bien entendu ; et comme pour FSM1 ; d'une machine de MOORE.

Le séquenceur 2 contient également des éléments de logique anarchique, à savoir :

- **Compteurs :**

  Nous utilisons 2 compteurs synchrones, validés par les signaux INCRESP et INCRAM en provenance de FSM2.

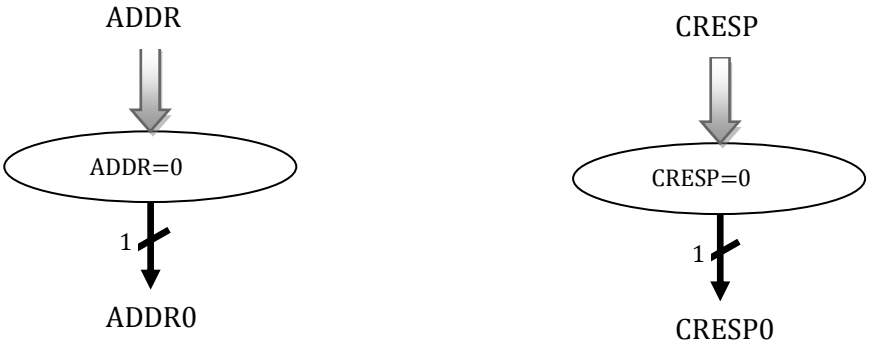
L'incrémentation (+1) est activée par le front montant d'horloge.

ADDR est sur 8 bits (0 → 255)

CRESP est sur 6 bits (0 → 63)

- **Circuit logique combinatoire** :

Ces circuits détectent l'occurrence de la machine 0 à la sortie des compteurs ADDR et CRESP.



III.7)- Conclusion

Dans ce chapitre, nous avons établi l'architecture de notre système. Aucune difficulté particulière n'est à relever hormis l'enchaînement des registres à décalage. A présent, il nous reste à valider cette architecture au moyen d'une simulation VHDL.

CHAPITRE IV

IMPLEMENTATION
ET
RESULTATS EXPERIMENTAUX

IV.1)- L'implantation sur FPGA de l'architecture du corrélateur

L'implantation de notre architecture du corrélateur numérique sur FPGA consiste en la programmation du circuit à l'aide du langage de description de haut niveau VHDL. Nous allons donc décrire le circuit sous forme d'algorithme VHDL. La seconde étape consiste à effectuer une simulation au moyen du logiciel QUARTUS II.

IV.2)- Compilation

La compilation à été validé à l'aide de l'outil Quartus II (version9.1) de la société Altéra [Altéra-web]. Ce logiciel fournit, à la fin de ces phases, divers rapports et schémas. Parmi ces documents, le rapport de compilation **(Figure IV-1)** nous donne les quantités de ressources matérielles utilisées pour l'implémentation physique du système, tandis que la fonction RTL viewer (Register Transfert Level) présente une description hiérarchique par niveaux d'abstraction du système sous forme de schémas.

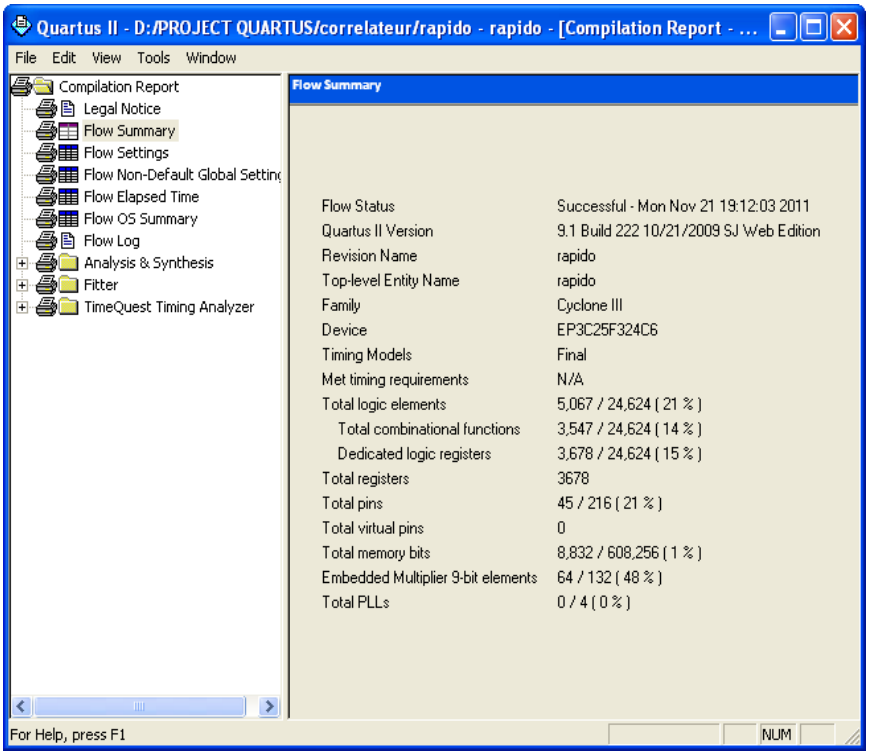


Figure (IV.1) : *Rapport de compilation du logiciel Quartus II*

Le Composant FPGA :

CYCLONE III : EP3C25F324C6 : Ce circuit dispose de :

- ✓ LOGIC ELEMENT : 24624
- ✓ BROCHES I/O : 216
- ✓ MEMOIRE : 608256
- ✓ MULTIPLIEURS 9 BITS : 132
- ✓ PLL : 4

La compilation ne pose pas de problème particulier. Elle dure environ 8 mn sur un PC :

- ✓ 2-C ore PENTTIUM
- ✓ 2GHZ.
- ✓ RAM : 1GO.

Nous exposerons les résultats d'occupation en ressources matérielles sous forme de nombre d'éléments logiques (LEs) correspondant. Le coût global en ressources matérielles de notre système composé d'une architecture simple est détaillé dans le Tableau (IV.1).

Le composant **CYCLONE III EP3C25F324C6**	
Eléments Logiques	5,067/24,624 (21%)
Implémentation bloc Mémoire (en bit)	138,240/608,256 (23%)
Multiplieur -9bit	64/132 (48%)
Totale fonction combinatoire	3,547/24,624 (14%)
Totale PINs	45/216 (21%)

Tableau (IV.1) : Taux d'occupation de l'architecture

IV.3)- Résultats de la simulation

À F =100 MHz

Les essais de simulation ont donné :

- Lecture des données X et Y : # 20.5 μs. (Vecteur X et Y: # 20.5 μs), voir figure **(IV.3).**
- Traitement : 8.37 μs.
- Le signal RESULT est mis à 1 à l'instant 28.88 μs signalant la disponibilité des résultats en RAM.

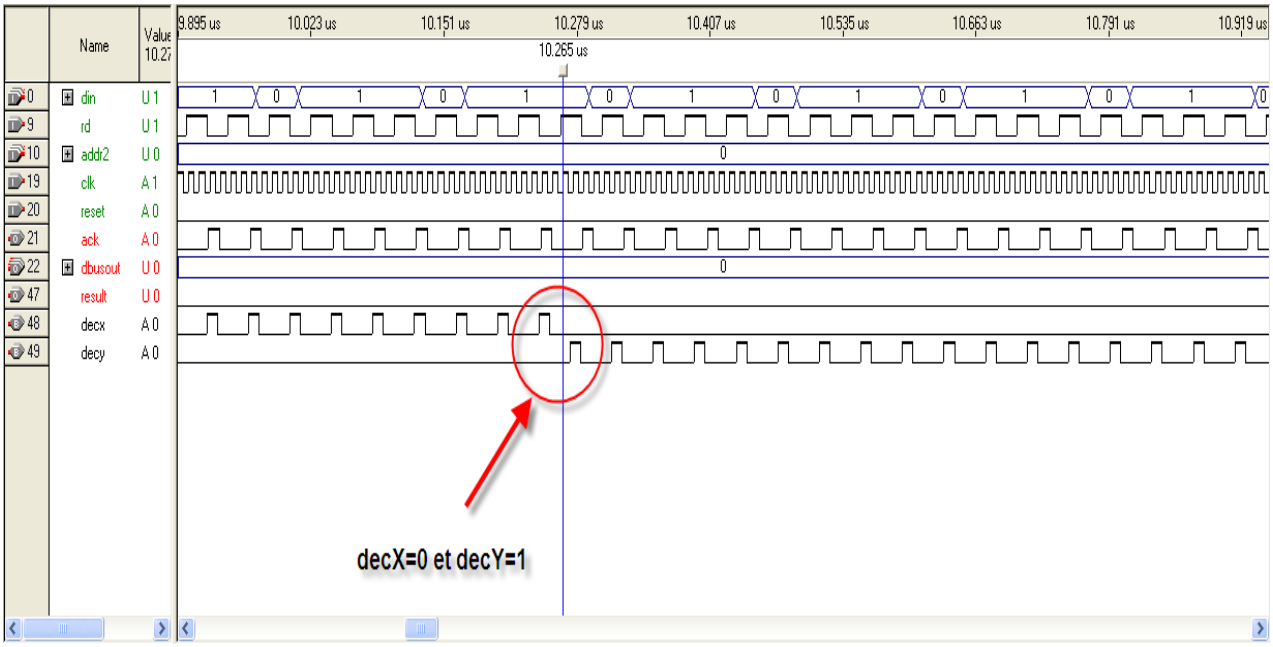


Figure (IV.2) : *Fin du décalage du vecteur X (decX) et début du décalage du vecteur Y (decY)*

La figure suivante présente une vue partielle du chronogramme des résultats ; à l' instant t=10.265 μs; on remarque le moment ou le décalage des données du vecteur X (DECX=0) est terminé et le début du décalage des données du vecteur Y dans le registre (DECY=1).

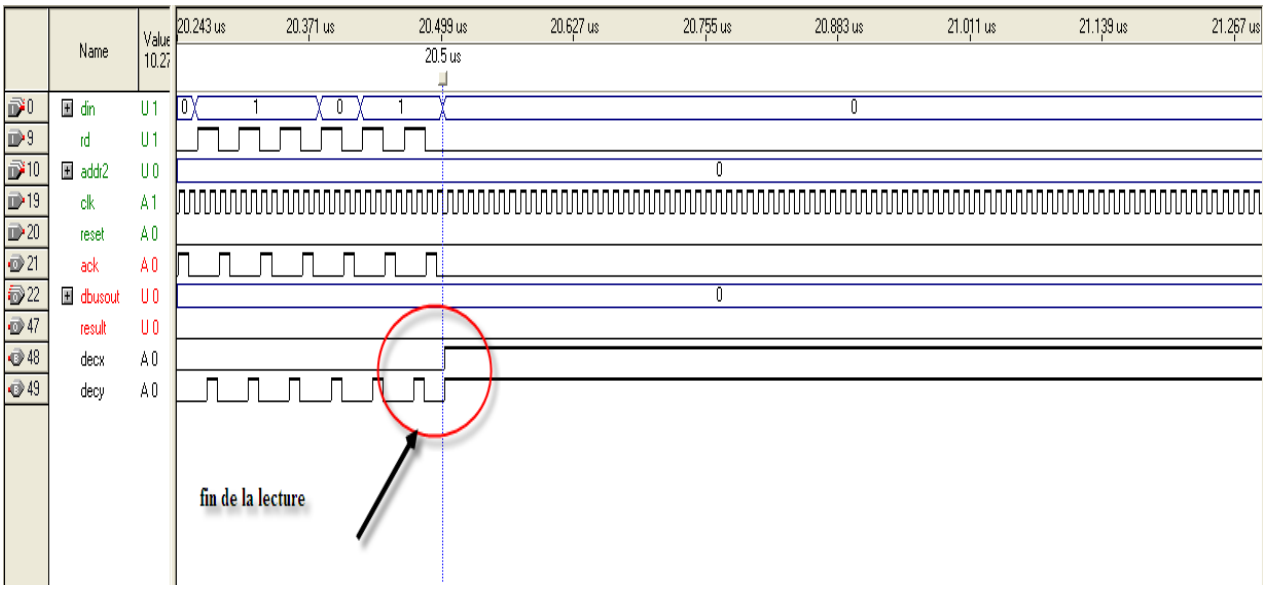


Figure (IV.3) : *fin de lecture des données à* 20.5 μs

On remarque que le chronogramme est subdivisé en deux parties la première en haut représente :

- ✓ Une Entrée sous forme de bus codé en 8 bit en décimal (DIN) son intervalle est $[0 - 20.5]\ \mu s$,
  X : correspond à DIN entre $[0 - 10.25]\ \mu s$;
  Y : correspond à DIN entre $[10.25 - 20.5]\ \mu s$
- ✓ Signal de lecture (Rd),
- ✓ Entrée horloge (Clk),
- ✓ Le signal de remise à zéro « reset » est activé juste au départ de la simulation et son intervalle est $[0.4 - 6.75]\ \eta s$.

La deuxième est la partie détaillée de

- ✓ Bus de sortie encodé dans notre cas sur 24 bits.
- ✓ Signal RESULT
- ✓ Signal de sortie acquittement ACK.

**Les essais de simulation ont porté sur :

1)- Vecteurs X et Y identiques (Autocorrélation)

X et Y sont positionnés à l'état « 1 » ;

Période X et Y : 10.25 μs.

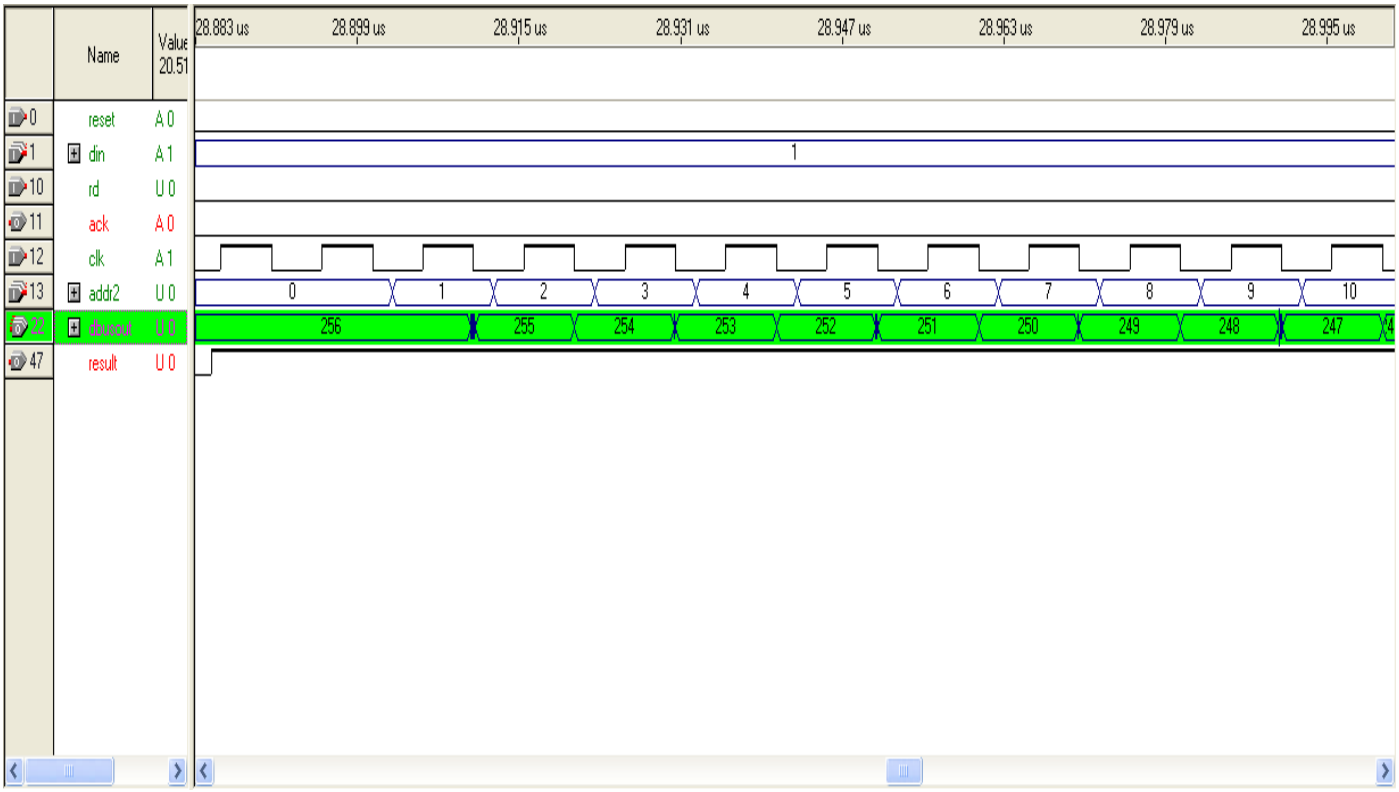
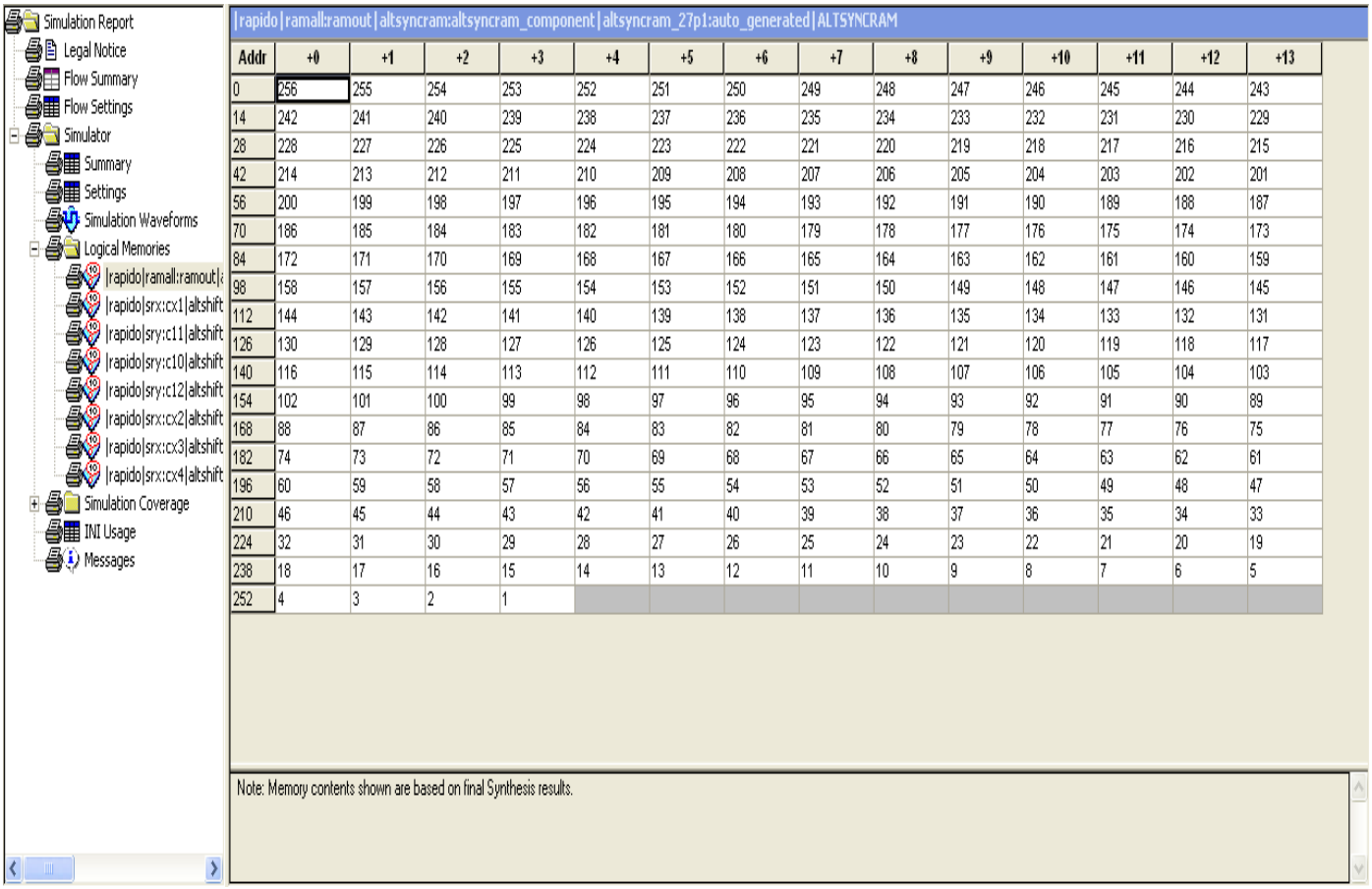


Figure (IV.4) : *Vue partielle du chronogramme des résultats «signaux à l'état haut »*

Interprétation :

D'après la figure de simulation, nous avons donné des valeurs à l'entrée (DIN) et après une durée de 28.88 μs, le signal RESULT est mis à 1 ce qui signale la disponibilité des résultats en RAM, nous avons récupéré les résultats sur le bus de sortie (dbusout), les valeurs des entrées et les sorties sont présentées en décimale dans le contenu de la mémoire.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
0	256	255	254	253	252	251	250	249	248	247	246	245	244	243
14	242	241	240	239	238	237	236	235	234	233	232	231	230	229
28	228	227	226	225	224	223	222	221	220	219	218	217	216	215
42	214	213	212	211	210	209	208	207	206	205	204	203	202	201
56	200	199	198	197	196	195	194	193	192	191	190	189	188	187
70	186	185	184	183	182	181	180	179	178	177	176	175	174	173
84	172	171	170	169	168	167	166	165	164	163	162	161	160	159
98	158	157	156	155	154	153	152	151	150	149	148	147	146	145
112	144	143	142	141	140	139	138	137	136	135	134	133	132	131
126	130	129	128	127	126	125	124	123	122	121	120	119	118	117
140	116	115	114	113	112	111	110	109	108	107	106	105	104	103
154	102	101	100	99	98	97	96	95	94	93	92	91	90	89
168	88	87	86	85	84	83	82	81	80	79	78	77	76	75
182	74	73	72	71	70	69	68	67	66	65	64	63	62	61
196	60	59	58	57	56	55	54	53	52	51	50	49	48	47
210	46	45	44	43	42	41	40	39	38	37	36	35	34	33
224	32	31	30	29	28	27	26	25	24	23	22	21	20	19
238	18	17	16	15	14	13	12	11	10	9	8	7	6	5
252	4	3	2	1										

Figure (IV.5) : *Vue du contenu de la mémoire (RAMOUT) « signaux à l'état haut »*

Cette figure présente les résultats du contenu de la mémoire de sortie RAMOUT fournit par le logiciel à la fin de la simulation. On remarque que les résultats conformes aux résultats donnés sur MATLAB.

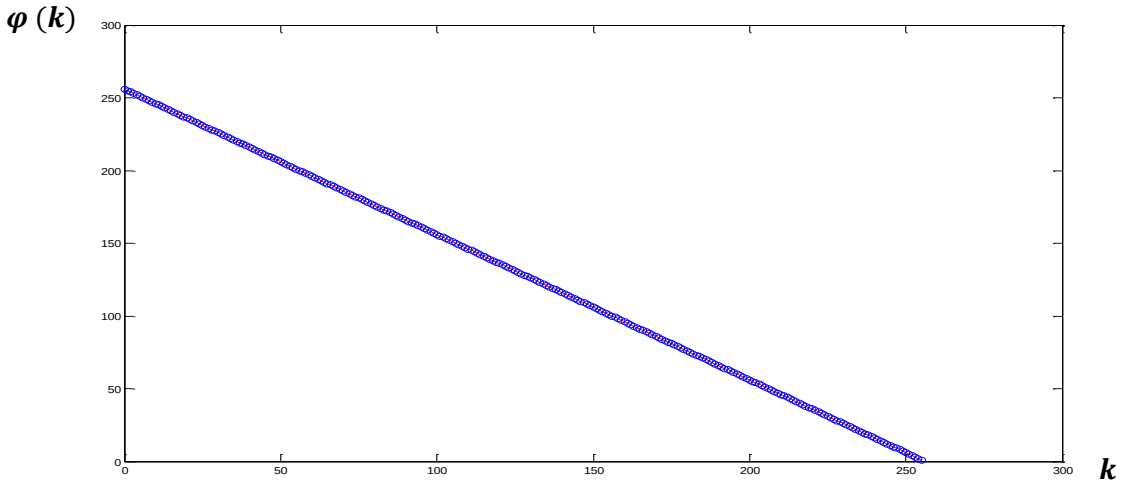
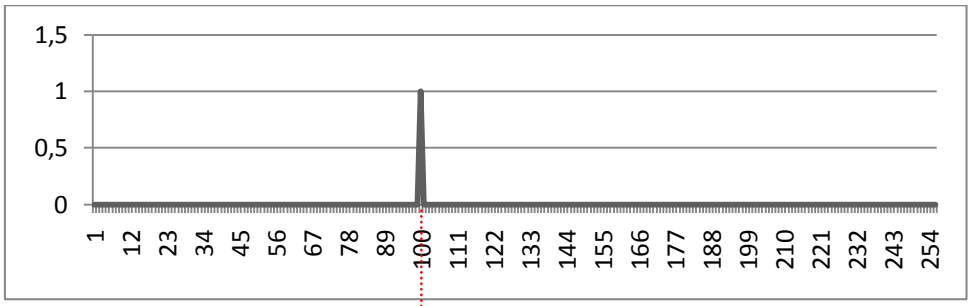


Figure (IV.6) : *Autocorrélation sous Matlab (Bleu : résultats Quartus, rouge : résultats Matlab)*

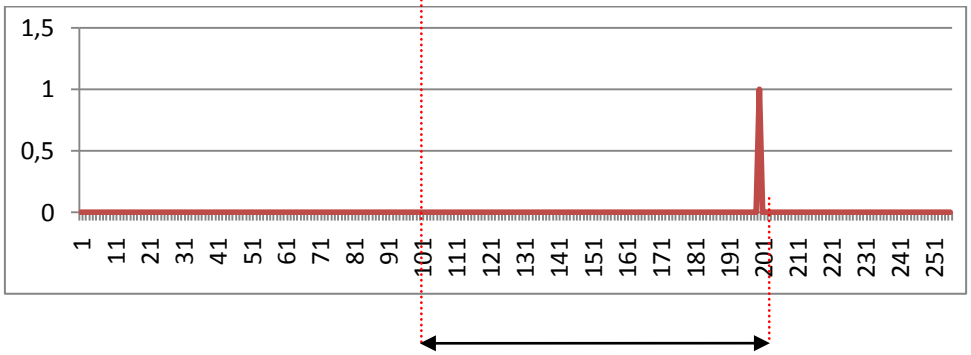
2)- Mesure de retard : signaux forme « impulsion & impulsion retardé »

X : Durée de l'impulsion : 40 ηs à t = 4 μs

Y : Durée de l'impulsion : 40 ηs à t = 18.25 μs

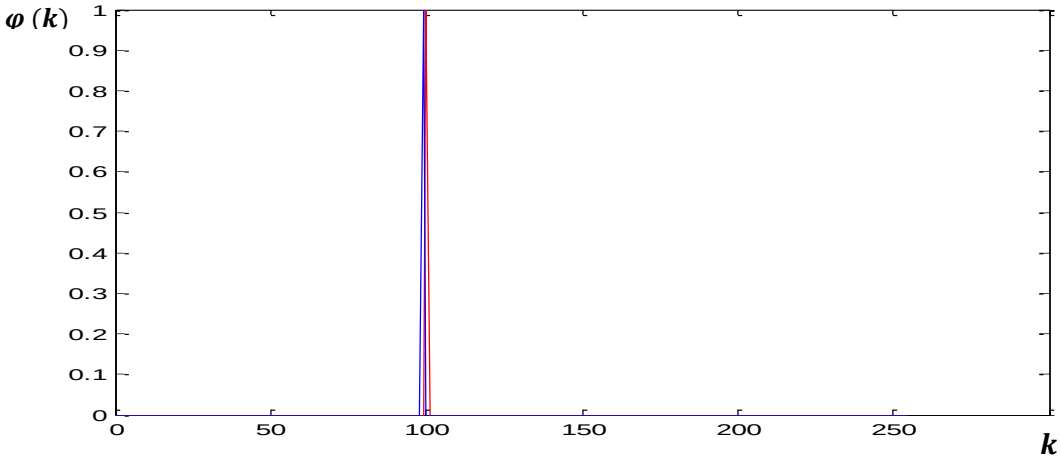


(a)



$\tau = 100\ \eta s$

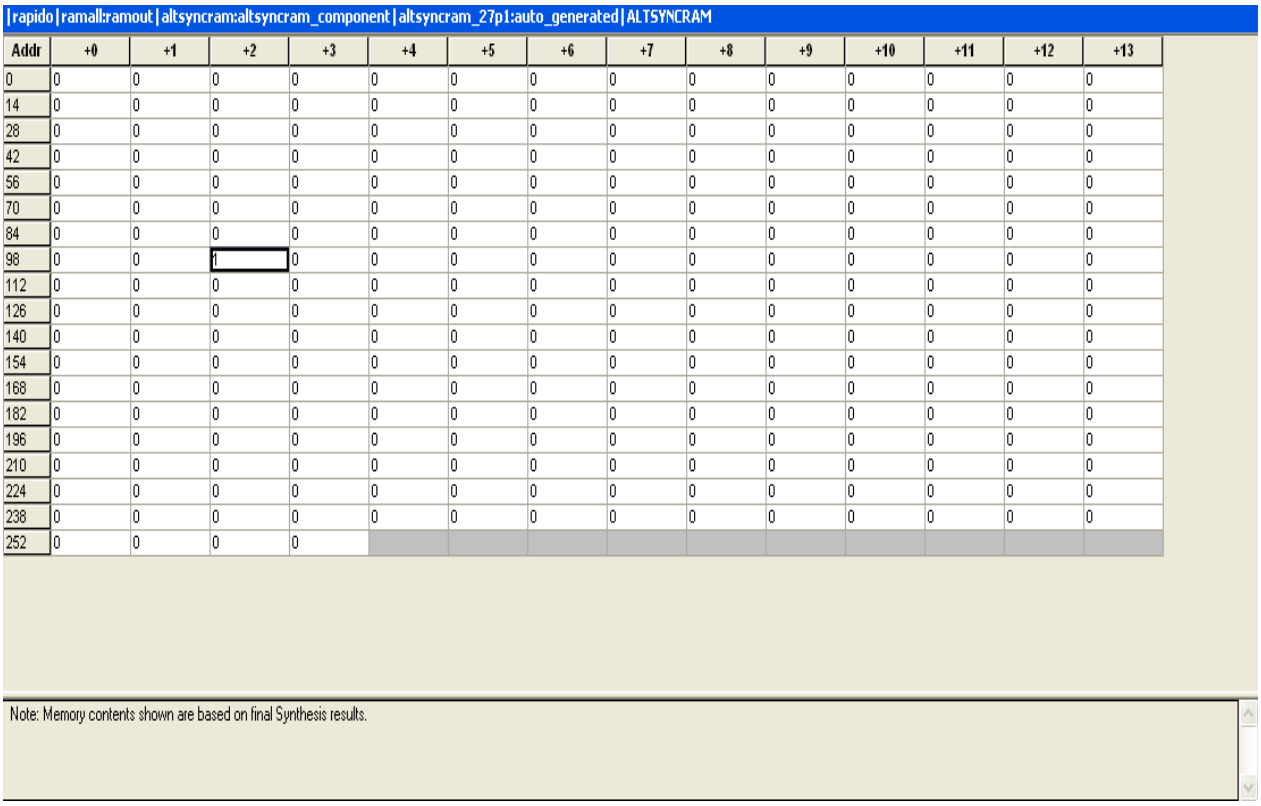
(b)



(c)

Figure (IV.7) : ***(a)*** *signal X,* ***(b)*** *signal Y,* ***(c)*** *intercorrélation sous Matlab*

(Bleu : résultats Quartus, rouge : résultats Matlab)



Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0
84	0	0	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	1	0	0	0	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0	0	0	0	0	0	0
126	0	0	0	0	0	0	0	0	0	0	0	0	0	0
140	0	0	0	0	0	0	0	0	0	0	0	0	0	0
154	0	0	0	0	0	0	0	0	0	0	0	0	0	0
168	0	0	0	0	0	0	0	0	0	0	0	0	0	0
182	0	0	0	0	0	0	0	0	0	0	0	0	0	0
196	0	0	0	0	0	0	0	0	0	0	0	0	0	0
210	0	0	0	0	0	0	0	0	0	0	0	0	0	0
224	0	0	0	0	0	0	0	0	0	0	0	0	0	0
238	0	0	0	0	0	0	0	0	0	0	0	0	0	0
252	0	0	0	0										

Figure (IV.8) : *Vue du contenu de la mémoire (RAMOUT) forme :* « impulsion & impulsion retardé »

Remarque :

On peut vérifier que le retard est donné par la fonction de corrélation et dans notre exemple il est égale à 100 ηs .

3)- Vecteurs X et Y identiques (autocorrélation)

Signal carré de période : 240 ηs ; La valeur MAX : $\varphi\,(0) = 127$.

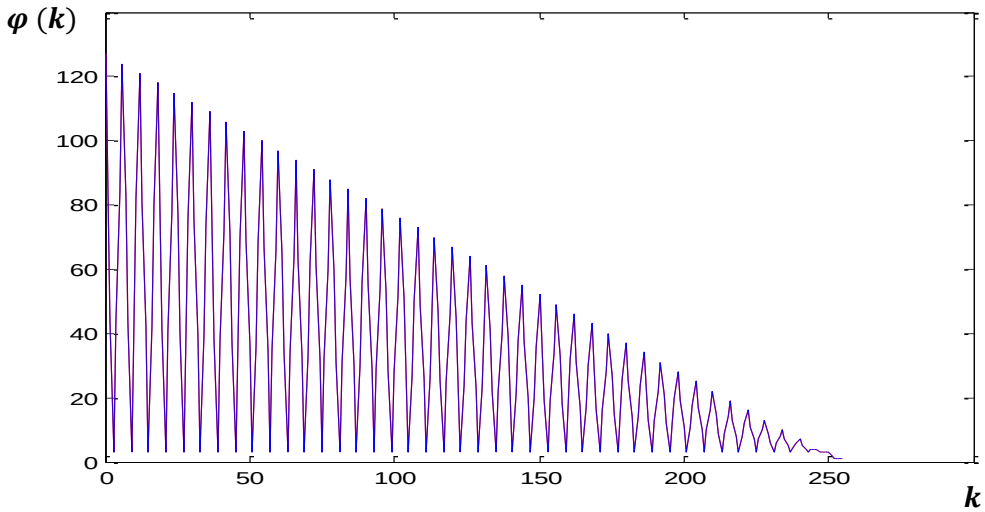
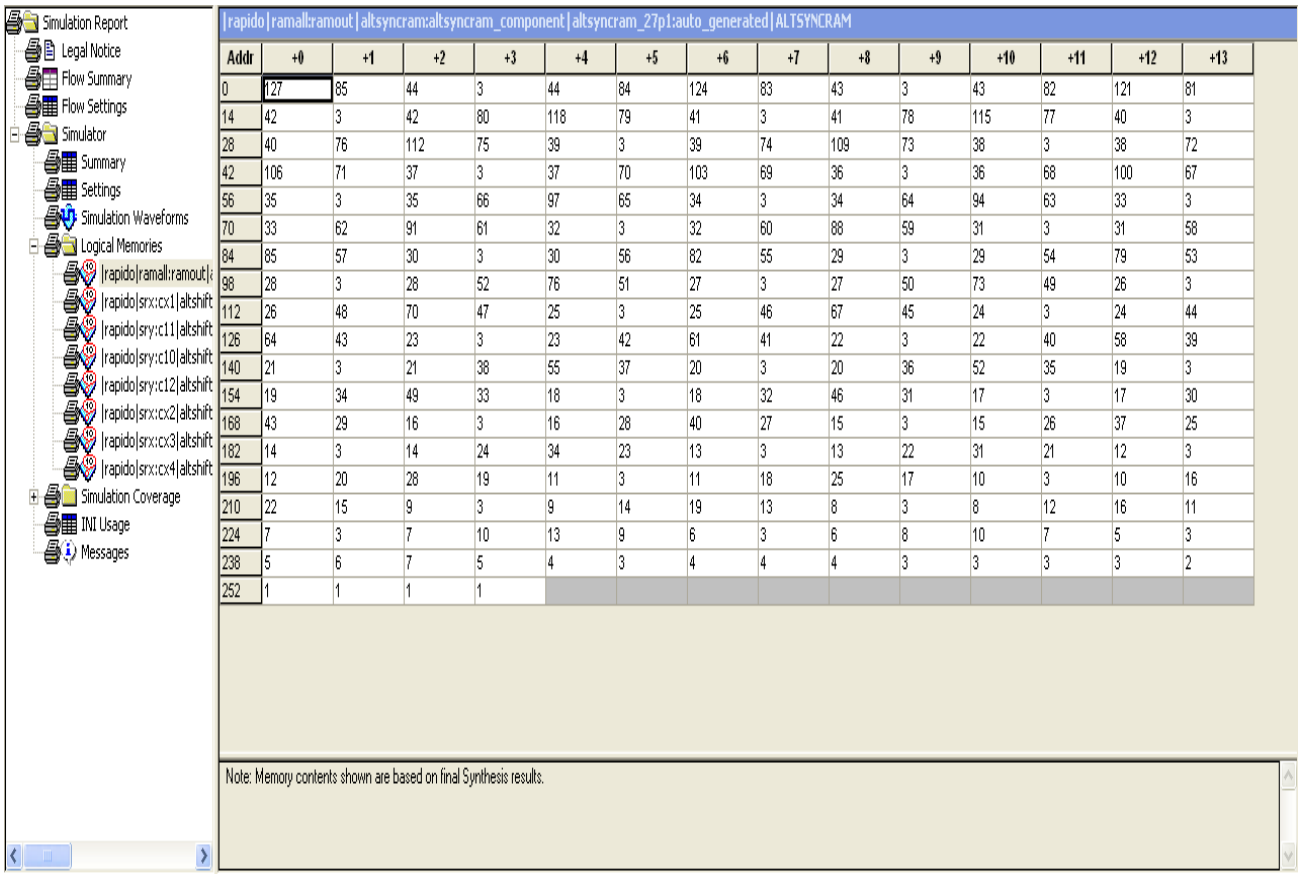


Figure (IV.9) : *Autocorrélation sous Matlab (Bleu : résultats Quartus, rouge : résultats Matlab)*



Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
0	127	85	44	3	44	84	124	83	43	3	43	82	121	81
14	42	3	42	80	118	79	41	3	41	78	115	77	40	3
28	40	76	112	75	39	3	39	74	109	73	38	3	38	72
42	106	71	37	3	37	70	103	69	36	3	36	68	100	67
56	35	3	35	66	97	65	34	3	34	64	94	63	33	3
70	33	62	91	61	32	3	32	60	88	59	31	3	31	58
84	85	57	30	3	30	56	82	55	29	3	29	54	79	53
98	28	3	28	52	76	51	27	3	27	50	73	49	26	3
112	26	48	70	47	25	3	25	46	67	45	24	3	24	44
126	64	43	23	3	23	42	61	41	22	3	22	40	58	39
140	21	3	21	38	55	37	20	3	20	36	52	35	19	3
154	19	34	49	33	18	3	18	32	46	31	17	3	17	30
168	43	29	16	3	16	28	40	27	15	3	15	26	37	25
182	14	3	14	24	34	23	13	3	13	22	31	21	12	3
196	12	20	28	19	11	3	11	18	25	17	10	3	10	16
210	22	15	9	3	9	14	19	13	8	3	8	12	16	11
224	7	3	7	10	13	9	6	3	6	8	10	7	5	3
238	5	6	7	5	4	3	4	4	4	3	3	3	3	2
252	1	1	1	1										

Figure (IV.10) : *Vue du contenu de la mémoire (RAMOUT)*

4)- Vecteurs X et Y identiques « signaux carrés décalés »

« X » et « Y » signaux carrés de Période : 320 ηs

« X » et décalé de « Y » de 4 échantillons (demi-période) c.-à-d $\tau = 40 \times 4 = 160\ \eta s$

La valeur MAX :$\varphi\ (0) = 124$.

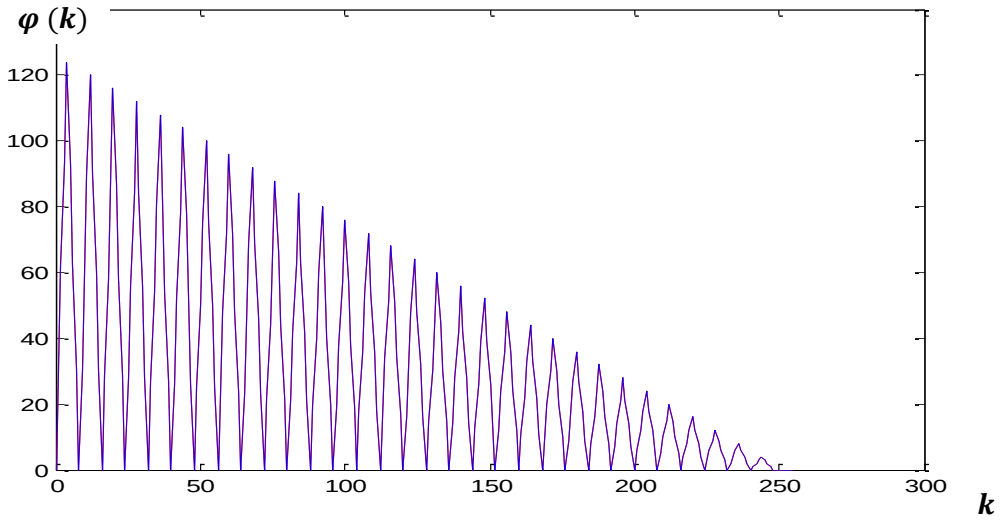


Figure (IV.11) : *Intercorrélation sous Matlab « signaux carrés décalés »*

5)- Vecteur X de période moitié de celle de Y

Signal carré de période X : 160 ηs

Signal carré de Période Y : 320 ηs

La valeur MAX : $\varphi\ (0) = 64$.

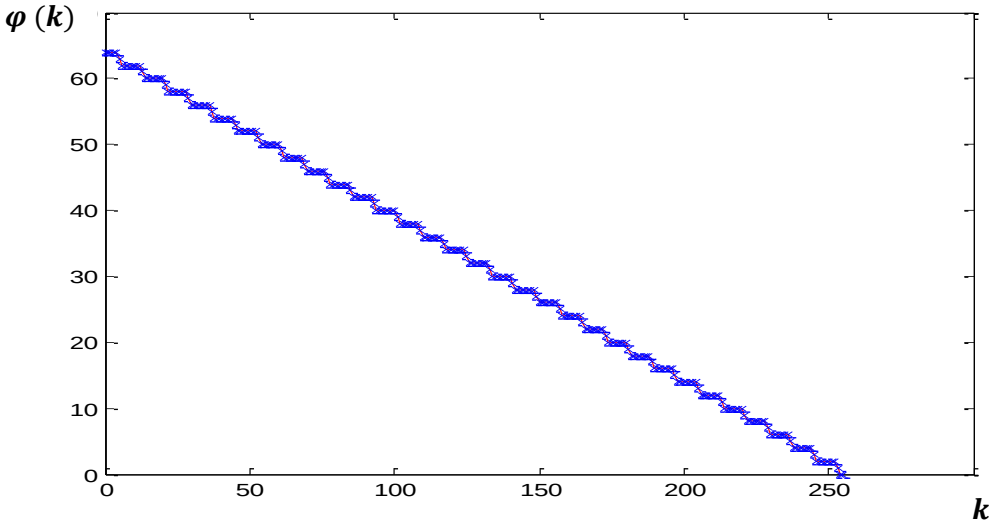
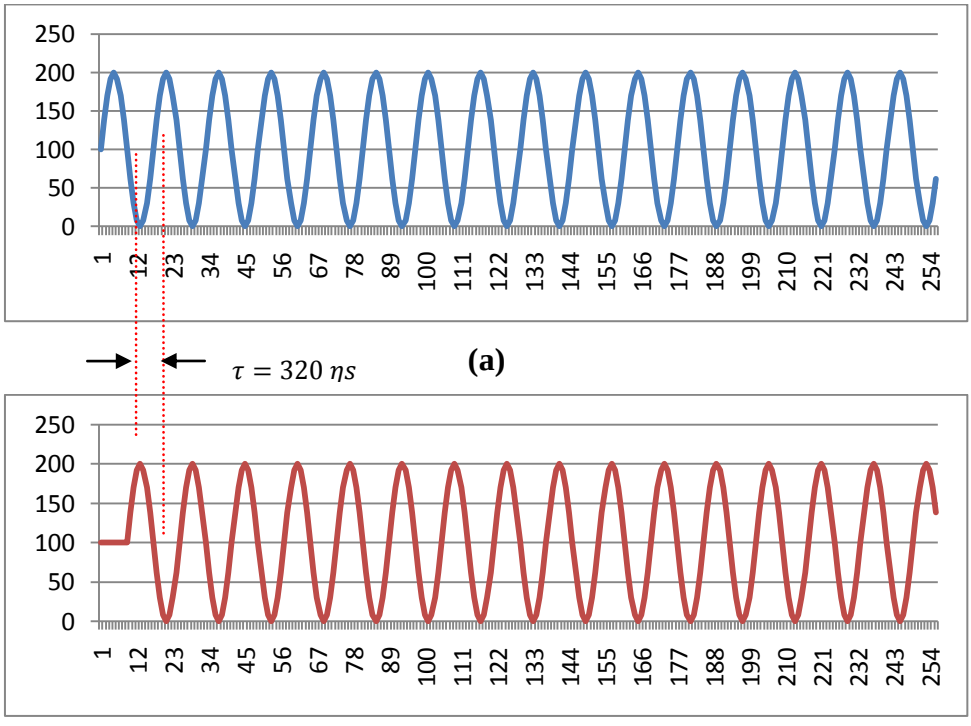


Figure (IV.12) : *Intercorrélation sous Matlab « X de période moitié de Y »*

6)- Signaux forme « Sinusoïde & Sinusoïde décalé »

Période X et Y: 640 ηs ;

« X » et décalé de « Y » de 8 échantillons (c.-à-d $\tau = 40 \times 8 = 320\ \eta s$)



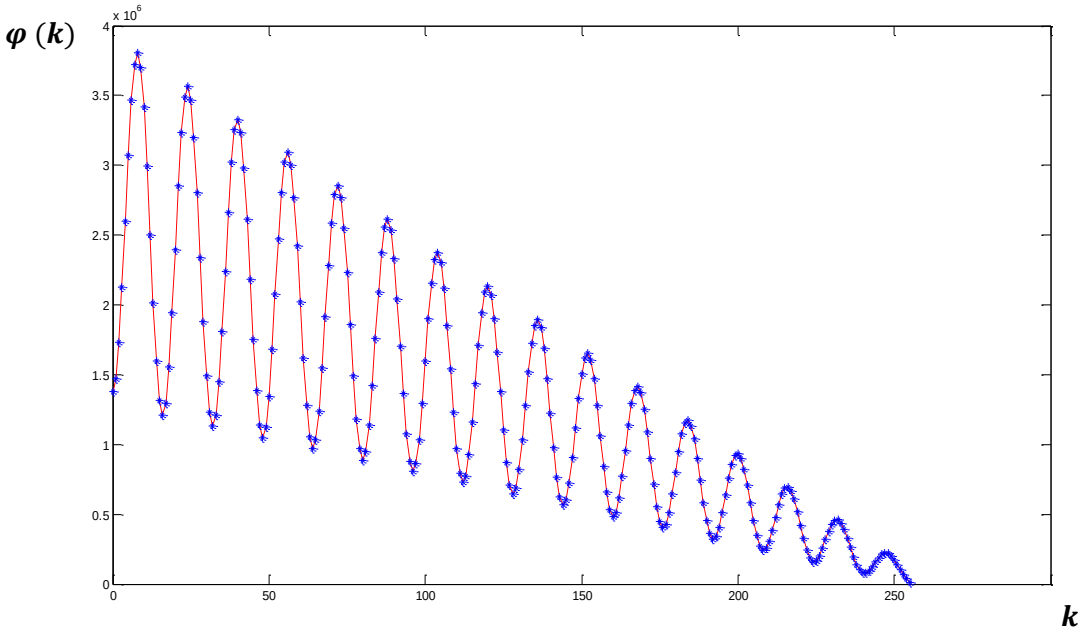


Figure (IV.13) : ***(a)*** *signal X,* ***(b)*** *signal Y,* ***(c)*** *intercorrélation sous Matlab*

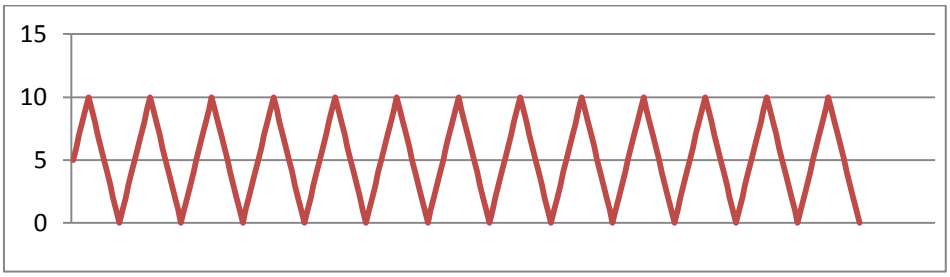
(Bleu : résultats Quartus, rouge : résultats Matlab)

7)- Signaux forme « Triangulaire » (AUTOCORRELATION)

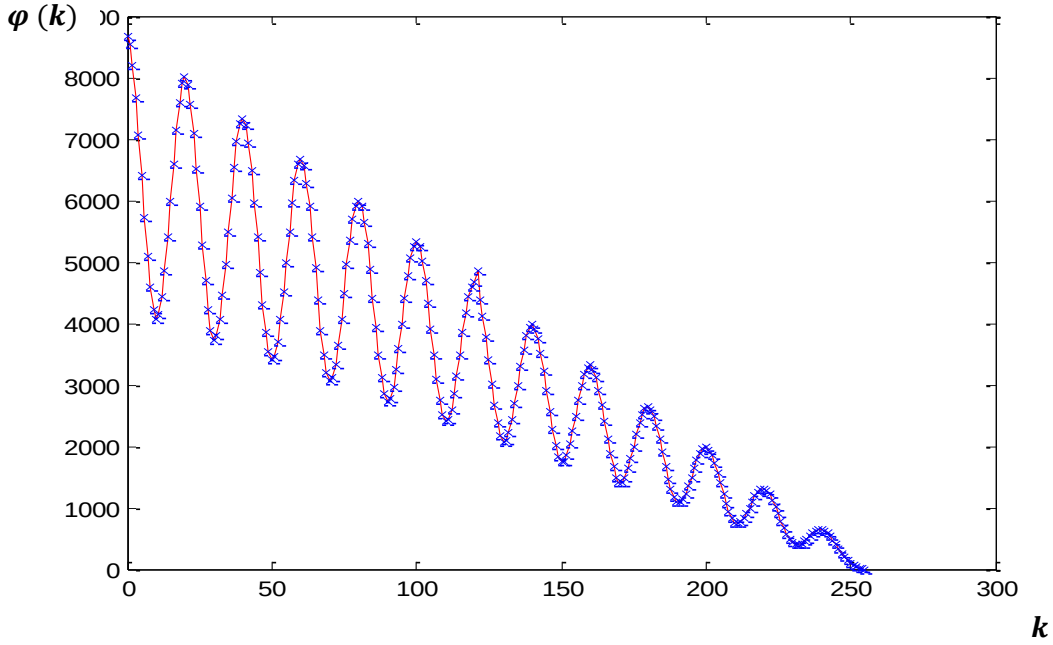
Niveau bas : 0 Période X : 800 ηs

Niveau haut : 10 Période Y :

Valeur MAX : $\varphi\,(0) = 8680$



(a)



(b)

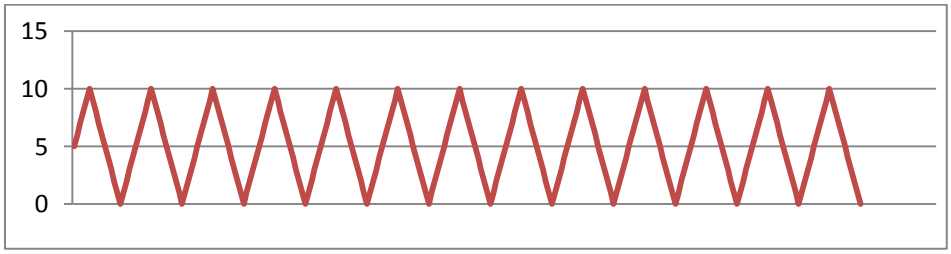
Figure (IV.14) : ***(a)*** *signal X,* ***(b)*** *autocorrélation sous Matlab*

5)- Signaux forme « Triangulaire » et « Triangulaire en escalier »

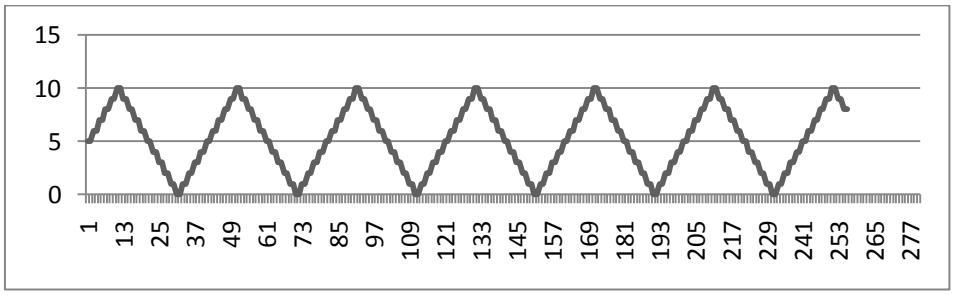
Niveau bas : 0 Période X : 800 ηs

Niveau haut : 10 Période Y : 1600 ηs

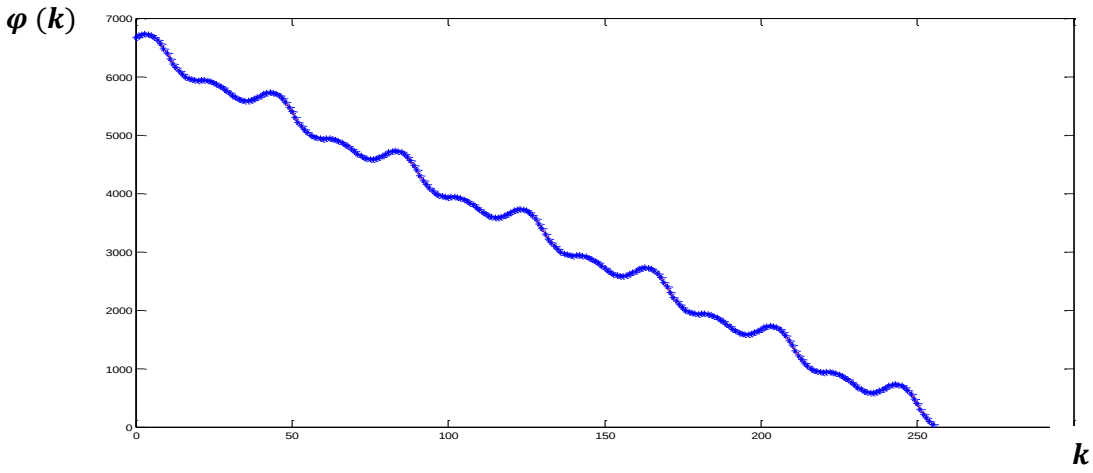
Valeur MAX : $\varphi\,(3) = 6733$



(a)



(b)



(c)

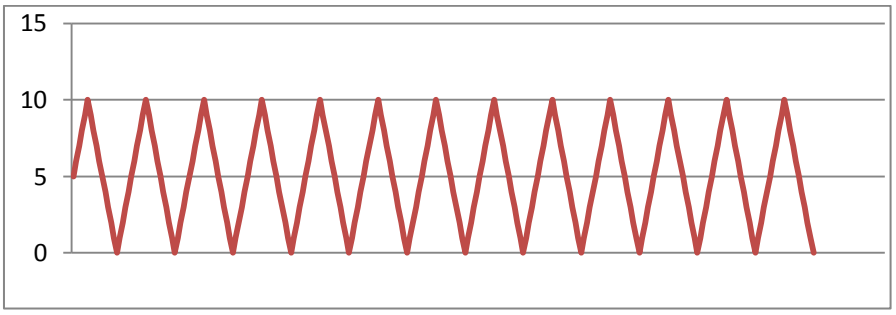
Figure (IV.15) : ***(a)*** *signal X,* ***(b)*** *signal Y,* ***(c)*** *intercorrélation sous Matlab « Triangulaire et triangulaire perturbé»*

8)- Signaux forme « Triangulaire» et forme « Triangulaire avec pic »

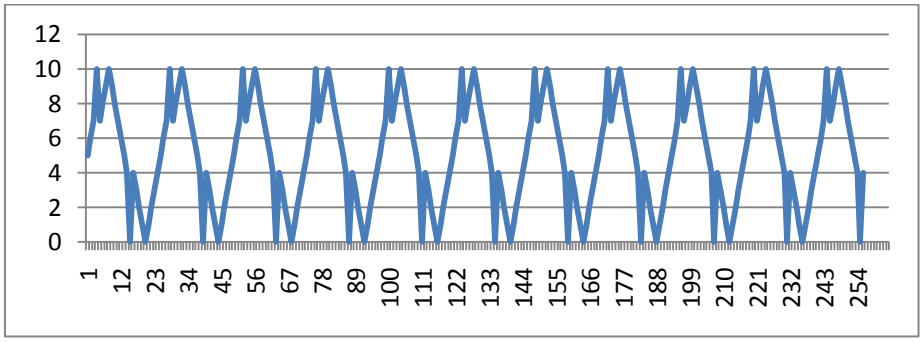
Niveau bas : 0 Période X : 800 ηs

Niveau haut : 10 Période Y : 1600 ηs

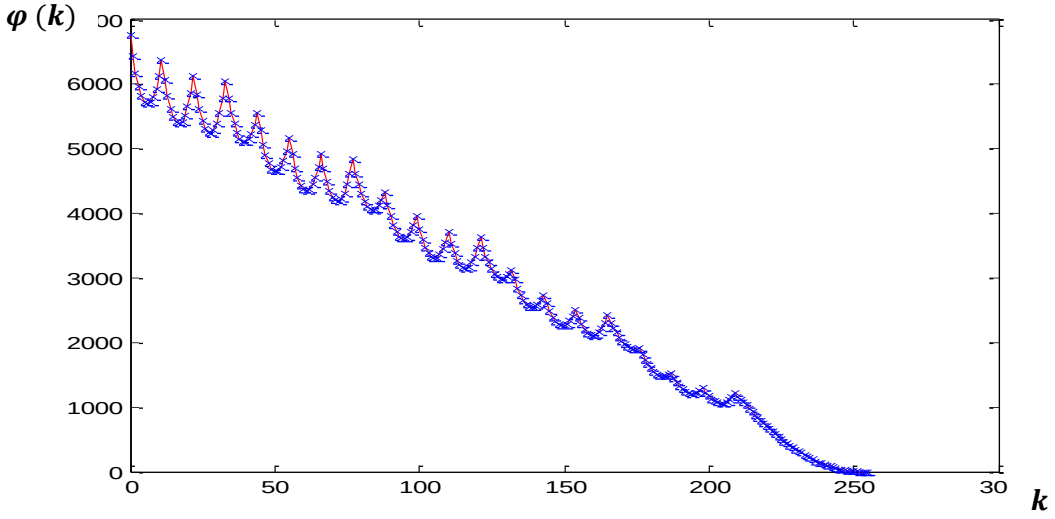
Valeur MAX : $\varphi\,(3) = 6858$



(a)



(b)



(c)

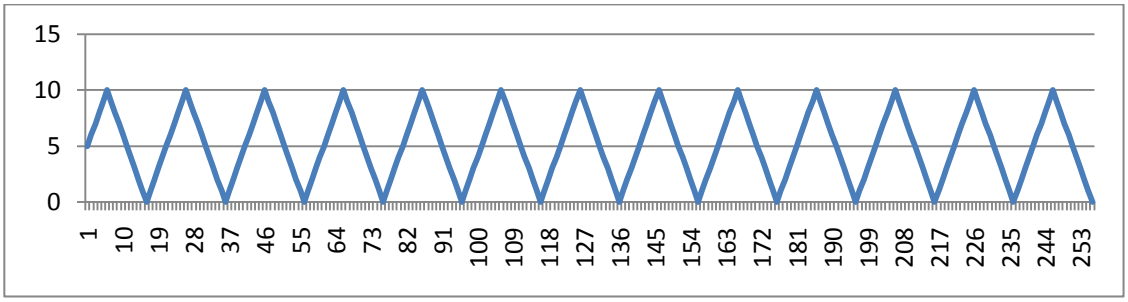
Figure (IV.16) : ***(a)*** *signal X,* ***(b)*** *signal Y,* ***(c)*** *intercorrélation sous Matlab*

9)- Signaux forme « triangulaire » et « triangulaire perturbé »

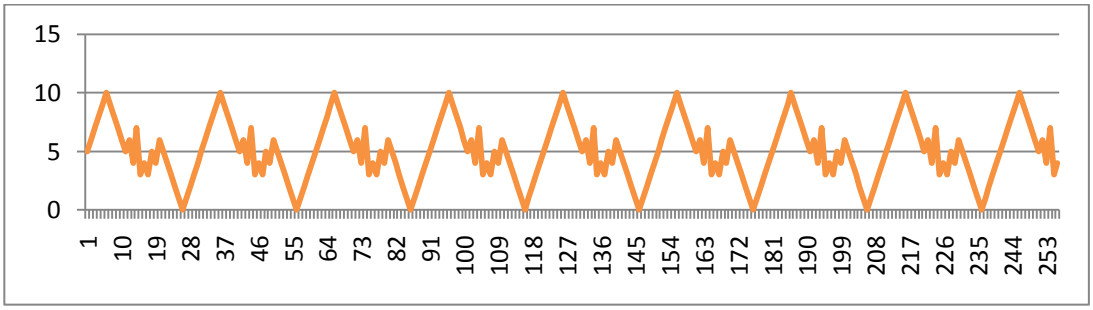
Période X : 800 ηs ;

Période Y : 1200 ηs ;

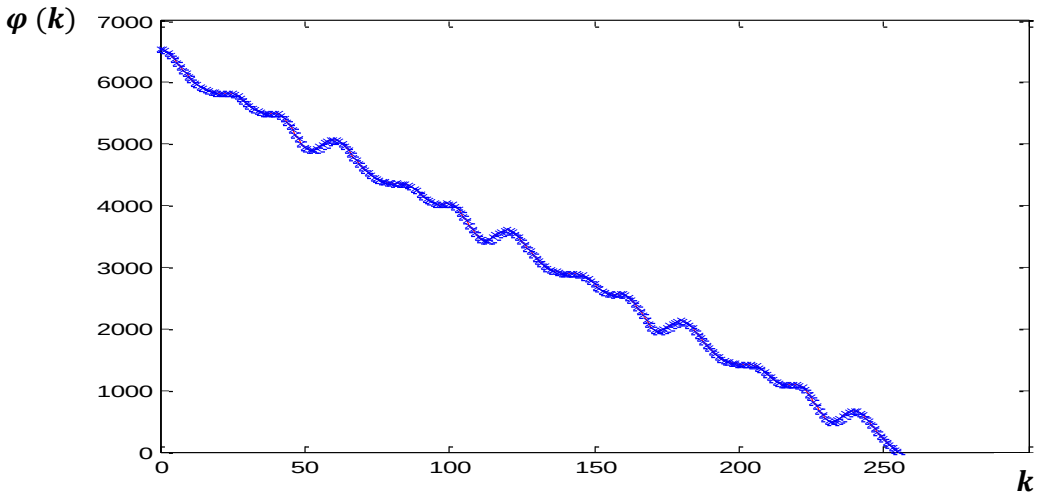
Valeur MAX : $\varphi\,(3) = 6758$



(a)



(b)



(c)

Figure (IV.17) : ***(a)*** *signal X,* ***(b)*** *signal Y,* ***(c)*** *intercorrélation sous Matlab et Quartus*

10)- Comparaison des trois figures (autocorrélation-triangulaire avec pic et triangulaire en escalier)

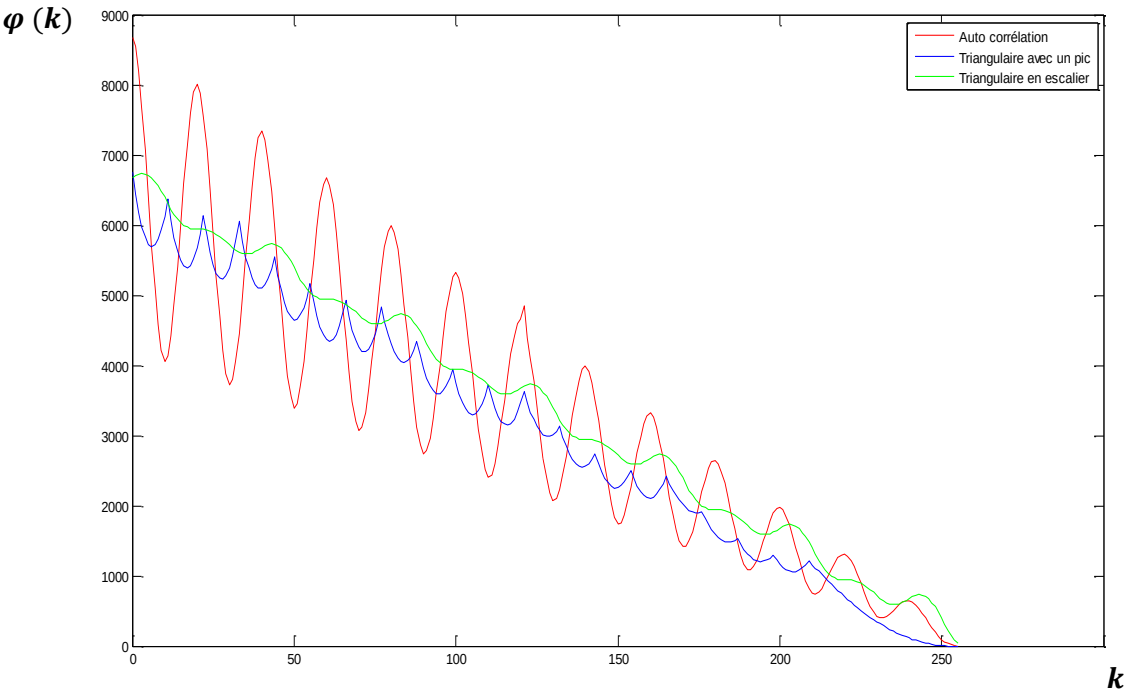


Figure (IV.18) : *Comparaison sous Matlab (rouge : Résultats autocorrélation, bleu : Résultats triangulaire avec pic, vert : Résultats triangulaire en escalier)*

Interprétation :

Cette figure illustre les trois courbes précédentes, on remarque que dans le cas de l'autocorrélation on à un maximum dans ($\varphi\ (0) = 8680$) c'est le cas ou les deux signaux ne sont pas décalés (une bonne similitude), dans le cas contraire on remarque que la valeur max est décalé et ceci présente un retard entre les signaux.

IV.4)- Comparaison avec les résultats obtenus sous MATLAB

La simulation nous a permis de valider notre architecture du corrélateur numérique. Le bloc de sortie représente les résultats de corrélation entre les signaux stockés dans la mémoire RAMALL et dans un fichier texte. L'exploitation de ce fichier va nous permettre d'effectuer une comparaison entre les résultats obtenus sous MATLAB à partir du programme effectué et les résultats issus de l'implémentation VHDL de notre architecture. Le traitement réalisé sur MATLAB dure environ 0.8 ms. Notre circuit est donc environ 100 fois plus rapide.

IV.5)- Comparaison et performance :

La comparaison des performances entre corrélateur numérique qui ont été déjà implantés et l'architecture proposée est résumée dans le tableau suivant :

Auteurs	Corrélation	Type d'architecture	Fréquence (**MHz**)	Temps de Traitement	Type de **FPGA**
[39]	Signaux (32×32)	Par décalage	87	800 ηs	Xilinx Virtex-4
[40]	Signaux (128x128)	TF (T Fourier)	100	60 μs	Xilinx Virtex-2
[41]	signaux (256x256)	Par décalage	89	20 μs	Xilinx Virtex-4
Proposée	Signaux (256 x 256)	Par décalage	100	8.37μs	CYCLONE III

Tableau (IV.2) : *Comparaison entre l'architecture proposé et autre architecture*.

Nous constatons d'après ce tableau que les performances en temps de traitement de notre réalisation sont supérieures aux résultats obtenus en [40] et [41]. Par rapport au résultat obtenu dans référence [39] leur temps de traitement est inférieur par rapport au notre d'un ordre de grandeur mais d'un autre côté la taille des signaux traités est également inférieure d'un ordre de grandeur.

IV.6)- Conclusion

Nous avons présenté dans ce chapitre les résultats de la simulation sous QUARTUS de notre corrélateur en utilisant le circuit FPGA CYCLONE III. Les résultats obtenus sont satisfaisants du point de vue de la fréquence de fonctionnement et des ressources utilisées.

CONCLUSION GENERALE

CONCLUSION ET PERSPECTIVES

Le travail présenté dans ce mémoire porte sur la conception et l'implantation sur circuit FPGA d'une architecture de calcul de la fonction de corrélation.

Cette étude entre dans le cadre d'une implantation mixte matérielle/logicielle de la corrélation à base des registres à décalage et les MAC. Ce que l'on peut dire, c'est que la réalisation d'un circuit tel que celui-ci, demande la capacité de gérer plusieurs fonctions à la fois. Si théoriquement on présente le flot de développement d'un circuit comme étant une succession d'étapes linéaires, il en est tout autre dans la réalité.

L'intérêt de la fonction de corrélation dans le domaine de traitement de signal n'est plus à démontrer, et son intégration sur circuit numérique (FPGA) est la cible d'un grand nombre de travaux, la corrélation par translation s'avère être une technique efficace et relativement aisée à mettre en œuvre par l'emploi des registres à décalages et des MAC.

Notre étude nous a conduit à considérer les points suivants :

Nous avons d'abord fait un tour d'horizon rapide sur la fonction de corrélation des signaux déterministes par ses aspects théoriques, puis nous avons abordé les circuits logiques programmables de types FPGA (CYCLONE III). L'étape suivante a consisté à effectuer une implantation logicielle de notre système afin de s'assurer de son bon fonctionnement. Enfin nous avons présenté les résultats obtenus par notre architecture.

Malgré les nombreuses difficultés rencontrées, la réalisation de ce corrélateur a été menée à bien. Du moins, une version fonctionnant parfaitement a été obtenue. Mais il est clair que le résultat présent n'est que la première base d'une réalisation plus évoluée. Cependant, ces interactions entre les différents niveaux de développement qui rendent le travail intéressant.

La représentation des signaux injectés est du type valeur entière codé sur 8 bits. De même des résultats de fonction de corrélation sont représentés sous forme d'entier sur 24 bits. Les résultats obtenus sont relativement satisfaisants. La fréquence de fonctionnement est assez élevée (100 MHz), le temps nécessaire aux calculs à été minimisé.

Malgré l'importance quantitative des travaux scientifiques, et malgré les immenses progrès réalisés dans les architectures numériques, de nombreuses problématiques restent à explorer. A cet effet, nous envisagerons comme perspectives, quelques axes de recherche pour améliorer les performances de conception pour la corrélation numérique. Ces axes sont récapitulés comme suit :

- Comme perspectives de poursuite du travail nous pouvons améliorer le temps de traitement, on augmente le degré de parallélisme ainsi le CYCLONE III nous offre 132 multiplieurs alors que nous avons utilisé que 64.
- Une deuxième perspective sera de mener les calculs en notation virgule fixe Ce qui nous permettrons de relier notre réalisation à un dispositif à base de PIC
- Une dernière perspective sera la suivante : utiliser la transformée en ondelette des signaux X et Y et calculé la corrélation entre les coefficients de ces transformés. En effet le nombre de coefficient après transformée en ondelette est bien plus faible que le nombre d'échantillons du signal d'entrés (la transformée en ondelette correspond à une compression du signal).

Les prochains travaux porteront sur l'optimisation de l'architecture en elle-même, mais aussi à son évolution par la recherche de meilleures performances et de nouvelles applications de traitement de signal et d'images en temps réel et pour des composants toujours plus rapides.

Bibliographie

[1] *L.LELONG* « Architecture SoC-FPGA pour la mesure temps réel par traitement d'image. Conception d'un système embarque imageur CMOS et circuit logique programmable » Présentée devant l'université Jean Monnet de Saint-Etienne pour obtenir le grade de docteur spécialité : électronique, décembre 2005

[2] *L.RAVERA* « Spectromètre Auto-corrélateur numérique Spatialisable pour l'instrument FIRST-HIFI » Présentée au centre d'étude spatiale des rayonnements, UPR8002 du CNRS pour obtenir le grade de docteur spécialité : conception des circuits microélectroniques, octobre 1999

[3] *J.MAX* « Les Corrélateurs Automatique en Temps Réel » article : Ingénieur au Centre d'Etude Nucléaire de Grenoble.

[4] *http://fr.wikipedia.org :* Historique de la corrélation en traitement de signal.

[5] *Christian Jutten* «Théorie du signal» Université Joseph Fourier - Polytech' Grenoble novembre 2009

[6] *G.Binet Mdc 61* « Energie et puissance des signaux, fonction de corrélation» cours UFR des sciences, université de Caen, 2009/2010.

[7] *F. Cottet*, « Aide-mémoire de traitement du signal » Science Supérieur (DUNOD) PARIS 2000, nouvelle présentation, 2005.

[8] *Serge Dos Santos* « cours de traitement de signal » signaux déterministes et signaux aléatoires) Université François-Rabelais de Tours, (maître de conférences) 2008/2009

[9] *C. Doignon* « Traitement Numérique Des Signaux Déterministes » Université Louis Pasteur de Strasbourg, Maitre de Conférences HdR, (2008-2009) - FIP 2A.

[10] *J.AUVRAY* «Les méthodes classiques du traitement de signal » Université, PARIS SUD-1, février 2000

[11] *V. Saeed* « Advanced Digital Signal Processing and Noise Reduction » livre 4ème edition, 2008 John Wiley & Sons, Ltd. ISBN: 978-0-470-75406-1.

[12] *B. Vijaya kumar, M.Abhijit, Richard Juday* « *Correlation Pattern Recognition* » article, UNIVERSITE CAMBRIDGE PRESS, 2005.

[13] *L.K. Cormack, S. B. Stevenson, C .M. Schor.* « Interocular correlation, luminance constrast and cyclopean processing». Vision Research, 31 (12) 2195-2207, 1991.

[14] *C.Sajabi* FPGA « Implimentation of a Correlator and a PN Code Generator » Spread Spectrum Special Research Report, EE 737-SPREAD SPECTRUM SYSTEMS, May 23, 2004

[15] *Marc Kristol,* « Réalisation VLSI d'un Corrélateur Linéaire Numérique » Projet de fin d'étude, hiver 2003/2004.

[16] *B. Hoppe, H. Meuth, M. Engels and R. Peters* « Design of digital correlation systems for low-intensity precision photon spectrocopic measurements » IEE Proc.-Circuits Devices Syst. Vol. 148 No. 5, October 2000.

[17] *M. Engels, B. Hoppe, H. Meuth and R. Peters* « A Single Chip 200 MHz Digital Correlation System for Laser Spectroscopy with 512 Correlation Channels » IEEE International Symposium on Circuits and Systemes, 1999.

[18] *S. DERRIEN* « Étude quantitative des techniques de partitionnement de réseaux de processeurs pour l'implantation sur circuits FPGA » thèse pour obtenir grade de docteur, décembre 2002, IRISA, 'UNIVERSITÉ DE RENNES 1

[19] *A.VACHOUX* « Modélisation de Systèmes Intégrés Numériques : Introduction à VHDL » Notes de cours Hiver 2002-2003 EPFL-STI-LSM, 2002

[20] *ZAHIR AIT OUALI* : « Application des FPGA à la commande d'un moteur asynchrone » Mémoire Pour l'obtention du diplôme de Magister en Automatique, Option : Automatique des systèmes continus et productique,

[21] *J.WEBER et M.MEAUDRE* « Circuits numériques et synthèse logique » un outil VHDL", Edition Masson collection technologie, 2006.

[22] *ETIENNE MESSERLI* « Manuel VHDL synthèse et simulation », Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (heig-vd), Version partielle septembre 2007.

[23] *JEREMIE DETREY* « Arithmétiques réelles sur FPGA virgule fixe, virgule flottante et système logarithmique ». Thèse doctorat Ecole Normale Supérieure de Lyon, 15 janvier 2007.

[24] *SEBASTIEN SNAIDERO* « Modélisation multidisciplinaire VHDL-AMS de systèmes complexes vers le prototypage virtuel ».Thèse présentée afin d'obtenir le grade de docteur de l'université Louis Pasteur Strasbourg I, décembre 2004.

[25] *DAVID GUIHAL* « Modélisation en langage VHDL-AMS des systèmes Pluridisciplinaires. Thèse présentée au laboratoire d'analyse et d'architecture des systèmes du CNRS en vue de l'obtention du titre de docteur de l'université Toulouse, Mai 2007.

[26] *Synopsys,* « SaberHDL : language-Independant Mixed –Signal Multi-Technology Simulator », Synposys Inc, Etats-Unis d'Amérique 2003.

[27] *D.SMITH* « HDL Chip Design: A pratical Guide for Designing, Synthesis & Simulating Asics &FPGAs using VHDL or Verilog » Doone Pubns,2006

[28] *D. DECKERS* « Composants programmables » Cours de 3è Bachelier en Electronique appliquée, Institut Supérieur Industriel de Mons Haute Ecole de la Communauté française en Hainaut Unité Electronique 2008-2009

[29] *P.MELET.* « Conception et réalisation d'un circuit numérique spécifique à étalement de spectre pour un système multicapteurs en milieu clos », Université Paul Sabatier, Toulouse. Dec. 2001.

[30] *C.GUILLEMINOT, L.ANDRIEUX, JJ.MERCIER* « A contribution to a virtual prototyping for a spread spectrum telecommunication system using VHDL-AMS » IEEE BMAS'05, SanJose,Californie,USA, 22-23 Sept. 2005.

[31] *V. Betz and J.Rose* « FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density ». In International Symposium on Field Programmable Gate Arrays (FPGA), 1999.

[32] *J. Arnold M. Gokhale, J. Stone and M. Kalinowski*. «Stream Oriented FPGA Computing in the Streams-C High Level Language ». In Kenneth L. Pocek and Jeffrey Arnold, editors,

Symposium on FPGAs for Custom Computing Machines. IEEE Computer Society Press, 2000.

[33] *J. Rose, A. El Gamal & A. Sangiovanni-Vincetelli*. «Architecture of Field Programmable Gate Arrays ». Proceedings of the IEEE, vol. 81, July 1993.

[34] *V. Betz* « Architectures and Methodologies for Dynamic Recon_gurable Logic (ADMREL) » - Information Societies Technology (IST) Program. Survey of existing _ne-grain reconfigurable hardware platforms, November 2002. v2.0. 70

[35] *http://www.altera.com* : site officiel ALTERA : CYCLONE III Device handbook, VOLUME-1- copyright 2010 Altéra corporation.

[36] *Gérald ARNOULD* « Etude et Conception d'Architectures Haut-Débit pour la Modulation et la Démodulation Numériques » Présentée devant l'université Université Paul Verlaine - Metz pour obtenir le grade de docteur spécialité : électronique, décembre 2006.

[37] *Wagner, F.* « Modelant le logiciel avec des machines d'état fini : Une approche pratique » publications d'Auerbach, 2006, ISBN 0-8493-8086-3.

[38] *Cassandras, C., Lafortune, S.* « introduction aux systèmes discrets d'événement ». Kluwer, ISBN 0-7923-8609-4, 1999.

[39] *Kulkarni, S.; Mazumder, F. Haddad, G.I* « A high-speed 32-bit parallel correlator for spread spectrum communication » Dept. of Electr. Eng & Comput. Sci, Michigan Univ, Ann Arbor, MI; Publication Year: 2005

[40] *Romero-Aguirre, Parra-Michel, R, Longoria-Gandara* « A Hardware-Efficient Frequency Domain Correlator Architecture for Acquisition Stage in GPS » Dept. of Electr. Eng, CINVESTAV-IPN, Mexico City, Mexico; Janvier 2011

[41] *Perez, M.C.; Urena, J.; Hernandez, A. De Marziani, C. Ochoa, A. Marnane, W.P* « FPGA Implementation of an Efficient Correlator for Complementary Sets of Sequences » Field Programmable Logic and Applications, 2006. FPL '06. International Conference on Digital Object Identifier: 10.1109/FPL.2006.311293, Publication Year: 2006

ANNEXE

Cette annexe présente une petite présentation du langage de description matérielle VHDL qui a été développé dans le but de faire la description, la vérification et la synthèse des circuits.

A.1)- HISTORIQUE DU LANGAGE DE DESCRIPTION MATERIELLE VHDL

Il existe actuellement différents langages qui permettent la description de circuits numériques. Alors, il faut déterminer le langage de modélisation le mieux adapté à nos besoins. Le VHDL et le VERILOG sont les deux langages qui s'imposent comme standards mondiaux et restent les plus utilisés des langages dans le domaine de la conception des circuits électriques (Entreprises Européennes utilisent majoritairement VHDL, Entreprises Américaines utilisent majoritairement VERILOG). Le langage de description VHDL (Very High Speed Integrated Circuit Hardware Description Language) est un langage de description matérielle (comportement et/ou architecture) pour les systèmes numériques très populaire dans le domaine d'industrie de conception. Il est le fruit d'un projet de recherche mené par le groupement IBM/Texas Instruments/Intermetrics et conséquence d'un besoin croissant d'outils de conception de haut niveau pour décrire les systèmes numériques et les circuits intégrés qui sont de plus en plus complexes. Ce langage est né dans les années 80 comme successeur du langage ADA (1979) au département de la défense américain DOD qui a lancé un appel d'offre pour créer un langage de description matérielle numérique standard. Ce langage est ouvert au domaine public en 1985 puis il est adopté par IEEE (Institute of Electrical and Electronic Engineers) comme standard et deviendra une norme en 1987 sous la dénomination VHDL {IEEE 1076-1987} puis complété et enrichi en 1993 {IEEE 1076- 1993 et IEEE 1164-1993} sous la dénomination VHDL'93 avec les extensions {IEEE 1076.3-1997, IEEE 1076.4-1995 }puis viennent d'importantes extensions qui touchent les signaux analogiques et mixtes en 1999{IEEE 1076.1-1999} sous le nom VHDL-AMS (1998 pour Verilog-AMS) où AMS est une abréviation en anglais pour(Analog and Mixed-Signal – AMS).

Le langage matériel VHDL-AMS est un standard de description et de modélisation des systèmes à temps continus et à temps discret exploitable pour des fins de simulation. Le VHDL-AMS est un langage particulièrement bien adapté à la simulation de problèmes multi - domaines (multidisciplinaire) et en 2001 la norme {IEEE 1076-2001} et finalement {IEEE 1076.1-2006}.

Le langage VHDL a été conçu pour être non seulement un langage de description matériel, mais aussi un langage de conception et de synthèse pour les systèmes numériques. Il est devenu incontournable dans le domaine de la conception des circuits numériques avec une grande capacité de modéliser les circuits digitaux à différents niveaux d'abstraction. Ce langage est basé sur une simulation évènementielle, et non temporelle des systèmes.

A.2)- POSITION DU VHDL AU SEIN DES LANGAGES INFORMATIQUES

Les langages informatiques sont une abstraction qui facilite l'élaboration des algorithmes et un moyen de coder et d'introduire l'information sur un calculateur. Le calculateur s'occupe-lui même de convertir ces langages en code machine. La diversité des langages se justifie par la diversité des domaines cibles. C'est ce que nous résumons comme suit :

- ✓ *Les langages de programmation orientés objet tels C++, VisualBasic ou Java*
- ✓ *Les langages de modélisation numérique tels VHDL {IEEE 1076-2000},Verilog {IEEE 1364-2001}, SystemC , SpecC et SystemVerilog.*
- ✓ *Les langages mathématiques formels explicites Matlab Simulink*
- ✓ *Les langages de modélisation implicites dédiés à l'électronique SPICE*
- ✓ *Les langages de modélisation mixte multi-domaines VHDL-AMS avec Verilog-AMS.*

<table>
<tr><td></td><td colspan="4">Outils issus du milieu de la CAO</td><td>Outils issus du monde logiciel</td></tr>
<tr><td>Type de langages
Niveau d'abstraction</td><td>Analogique</td><td colspan="2">Numérique</td><td>Mixte</td><td></td></tr>
<tr><td>Niveau Spécifications</td><td colspan="4">VHDL -AMS</td><td>C/C++</td></tr>
<tr><td>Niveau Comportemental</td><td rowspan="3">Spice</td><td rowspan="3">VHDL</td><td rowspan="3">Verilog</td><td rowspan="3">Saber</td><td rowspan="3">Matlab /Simulink</td><td></td></tr>
<tr><td>Niveau architecture</td><td></td></tr>
<tr><td>Niveau Composant</td><td></td></tr>
</table>

Tableau (A.01) : *Positionnement du langage VHDL au sein des autres langages scientifiques.*

Une description ou un modèle en VHDL d'un circuit est une abstraction ou une représentation du comportement de ce dernier ou un fichier VHDL contient une seule entité et son architecture (une ou plusieurs) avec la déclaration des paquetages.

A.3)- ETAPES DE CONCEPTION A BASE DU VHDL

Cette méthodologie contient les principales règles à utiliser pour les différents traitements numériques complexes et la vérification de conformité avec le cahier des charges.

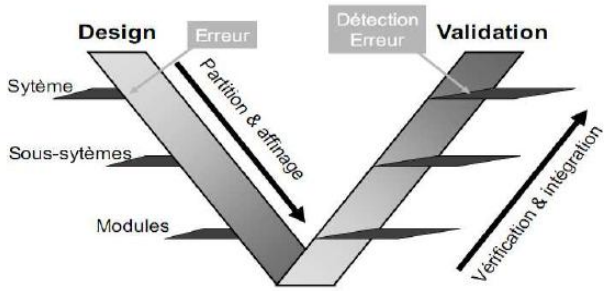


Figure (A.01) : *Le cycle de conception et de vérification traditionnel en V*

Les étapes fondamentales de conception matérielle à base du VHDL ainsi que les simulations tout au long du processus de conception sont illustrées dans le schéma suivant.

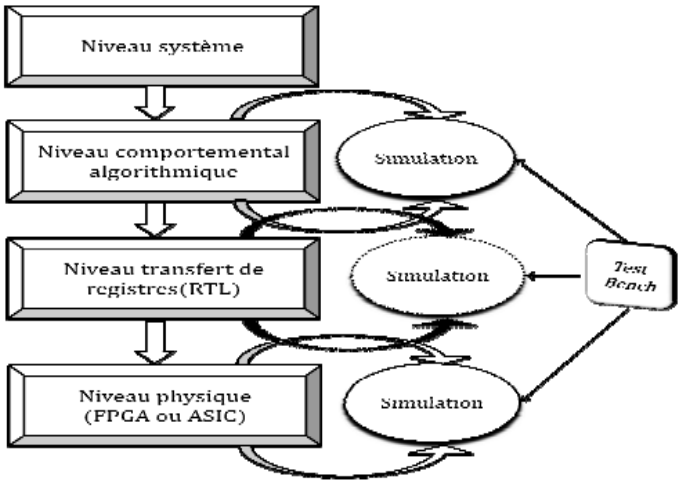


Figure (A.02) : *Les étapes fondamentales de conception matérielle*

La synthèse est l'étape qui transforme la description HDL en portes logiques et qui respecte les contraintes imposées par l'utilisateur (de superficie, de temps). Malheureusement, la synthèse des circuits ne concerne qu'un ensemble limité de descriptions VHDL qui n'est pas bien déterminée. Certaines structures non synthétisables sont prévisibles comme l'instruction « *after* » mais souvent il est difficile de distinguer le synthétisable du non synthétisable. Même la manière dont les instructions sont utilisées et d'un synthétiseur à l'autre peut rendre la description non synthétisable.

Conception et Implémentation d'un Corrélateur Numérique sur FPGA

Réalisé par : M^r Guettat Abdelghani

Encadrée par : M^r N.Boughanmi & M^r H.Bendouma

RESUME : L'objectif de notre projet de Magistère consiste à implanter un algorithme d'un corrélateur numérique des signaux qui nous permettent d'effectuer une corrélation complète. Il existe de nombreuses techniques pour effectuer la corrélation des signaux. Les critères de sélection de l'une ou l'autre des méthodes qui existent dépend de l'application envisagée et de la cible d'implantation de l'algorithme. La technique de corrélation que nous avons choisie pour une implantation FPGA est la technique de translation d'un signal par rapport à l'autre. Notre circuit cible est le FPGA CYCLONE III du constructeur ALTERA. La méthodologie de conception est la suivante : procéder à une implantation logicielle de cette architecture afin de pouvoir la valider, enfin aborder l'implantation matérielle proprement dite par une description comportementale de l'architecture à l'aide du langage VHDL, une simulation à l'aide du simulateur QUARTUS II.

Mots clé : FPGA, corrélateur, Conception, Description, Synthèse, VHDL, simulation, CYCLONE III.

ABSTRACT: The objective of our project of magister degree is to implement an algorithm a digital signal correlator that allows us to perform a complete correlation. There are many techniques to perform the correlation of signals. The criteria for selection of one or other of the methods that exist depend on the intended application and the target of implementing algorithm. The correlation technique we have chosen an FPGA implementation is the technique of translation of a report to another. Our circuit is the target FPGA CYCLONE III of the ALTERA manufacturer. The methodology of conception is: to make a software implementation of this architecture in order to validate it, finally addressing the actual hardware implementation by a behavioral description of the architecture using the VHDL language, a simulation to using the Quartus II simulator.

Keywords: FPGA, correlator, Design, Description, Synthesis, VHDL, simulation, CYCLONE III, QUARTUS II simulator