REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE
ET DE LA RECHERCHE SCIENTIFIQUE
Université Mohamed Boudiaf - M'sila

Faculté de Technologie
Département d'Electronique

THESE

Présentée en vue de l'obtention du diplôme de
Doctorat en Sciences en Electronique
Présentée par :

BELILITA FAIROUZ

THEME

Contribution à l'implémentation d'algorithmes de traitement numérique du signal sur des architectures parallèles très performantes. Application à la TFD et aux filtres numériques RII.

Soutenue publiquement le : 20 / 12 / 2017

Devant le jury :

Président :	**Mr HOCINI Abdesselam**	**Professeur**	**Université Mohamed Boudiaf - M'Sila**
Rapporteur :	**Mr CHIKOUCHE Djamel**	**Professeur**	**Université Mohamed Boudiaf - M'Sila**
Co-Rapporteur :	**Mr AMARDJIA Nourredine**	**Professeur**	**Université Ferhat Abbas - SETIF 1**
Examinateurs :	**Mr BOUROUBA Nacerdine**	**Professeur**	**Université Ferhat Abbas - SETIF 1**
	Mr LADJAL Mohamed	**MCA**	**Université Mohamed Boudiaf - M'Sila**
	Mr AYAD Mouloud	**MCA**	**Université Mohand Oulhadj - Bouira**

Dédicaces

A la mémoire de ma très chère défunte mère et à mon très cher père pour tout le bien qu'ils m'ont inculqué.

A mon mari et à mes enfants pour tout le bonheur qu'ils m'apportent dans ma modeste vie.

A toute ma grande famille.

Remerciements

Tout d'abord, je remercie Dieu, Le Tout Puissant, de m'avoir procuré la volonté nécessaire pour arriver à terme dans l'élaboration de cette thèse de doctorat en sciences.

Je remercie mes promoteurs, Monsieur CHIKOUCHE Djamel et Monsieur AMARDJIA Nourredine, Professeurs respectivement à l'Université Mohamed Boudiaf de M'sila et à l'Université Ferhat Abbas de Sétif 1, pour tous leurs conseils scientifiques et leurs continuels encouragements.

Je tiens également à exprimer toute ma gratitude aux membres du jury pour l'honneur qu'ils me font en acceptant d'examiner mon travail. Je cite Monsieur, HOCINI Abdesselam, Professeur à l'Université Mohamed Boudiaf de M'sila, Monsieur BOUROUBA Nacerdine, Professeur à l'Université Ferhat Abbas de Sétif 1, Monsieur LADJAL Mohamed, Maître de Conférence Classe A à l'Université Mohamed Boudiaf de M'sila et Monsieur AYAD Mouloud, Maître de Conférence Classe A à l'Université Mohand Oulhadj de Bouira.

Enfin, je remercie aussi tous mes collègues enseignants et j'exprime ma gratitude au corps administratif du département d'Electronique de la faculté de Technologie de l'université Mohamed Boudiaf de M'sila.

Table des figures
et
Liste des tableaux

Table des figures

Liste des tableaux

Table des matières

Table des matières

Page

Introduction générale

Introduction générale

La plupart des applications appartenant au très vaste et important domaine du traitement numérique du signal (TNS) travaillent sur un volume important de données et par conséquent accomplissent un nombre considérable de calculs [1 – 3]. Les différents domaines d'application tels que l'acoustique, la sismographie, le radar, le sonar, la reconnaissance de la parole, les systèmes numériques modernes de communications comme la radio et la télédiffusion numérique, le traitement de l'image comme le filtrage 2-D et la compression d'image, et la tomographie en médecine en sont des exemples concrets [4 – 12]. Cependant, beaucoup de ces applications dévoilent deux exigences fondamentales : l'exécution des calculs en temps réel et la localisation de l'unité de traitement au niveau même de l'application. Ces deux exigences sont pratiquement satisfaites par l'utilisation de processeurs dédiés, sujet d'intérêt de plusieurs travaux de recherches [13 – 26]. Aujourd'hui, la réalisation de tels processeurs est rendue possible grâce à l'avancée de la technologie microélectronique d'intégration à très grande échelle (VLSI) et à l'utilisation des techniques de conception par ordinateur (CAO), deux facteurs révélateurs de la révolution que nous vivons actuellement dans le domaine de l'implémentation sur silicium de la plupart des algorithmes du traitement numérique du signal, dont ceux de la Transformée de Fourier Discrète (TFD), de son algorithme de calcul rapide (TFR pour Transformée de Fourier Rapide ou FFT pour Fast Fourier Transform) et des filtres numériques à réponse impulsionnelle infinie appelés communément filtres récursifs IIR, sujets d'intérêt de cette thèse.

Dans ce sens, l'idée directrice de ce travail est de développer des architectures de processeurs dédiés, aussi performantes que possible en termes de temps de réponse, pour l'implémentation de la TFD, de la FFT et des filtres numériques récursifs IIR. L'utilisation d'architectures parallèles dans la construction de ces processeurs dédiés, et dont le but est une distribution efficace et appropriée des calculs sur un ensemble ordonné de processeurs élémentaires (PE), permettra d'atteindre des débits en données élevés [27 – 31]. Cette utilisation, accommodée avec une exécution en pipeline, est rendue possible grâce à la forme récursive dont jouissent les algorithmes de la TFD, de la FFT et ceux calculant les filtres numériques IIR.

Les réseaux systoliques présentés initialement par H.T. Kung et C.E. Leiserson [32], et adaptés ensuite pour l'implémentation d'un nombre important d'applications [33 – 38], auront une place importante parmi les architectures que nous développerons. En effet, la régularité de

la structure de leurs processeurs et la localité des interconnexions entre ces derniers les rendent très favorables à une implantation VLSI et leur donnent une remarquable puissance de calcul. Cette puissance de calcul est accentuée grâce à l'utilisation d'un parallélisme intensif qui est véhiculé par un pipelining aussi bien des données que des instructions. Une étude de performance des architectures proposées sera considérée pour permettre leur validation.

Partant de cet ensemble de fondements, le travail contenu dans cette thèse sera réparti sur quatre chapitres :

- Le premier chapitre sera destiné au développement de trois algorithmes rapides pour le calcul de la TFD : un algorithme FFT en base arbitraire *B* suivant l'entrelacement fréquentiel et deux autres algorithmes suivant l'entrelacement temporel, l'un en base 2 et l'autre en base 4. Le premier algorithme, sera une généralisation du célèbre algorithme de Cooley-Tukey [39] en base 2 à une base quelconque *B*. Les deux autres (en bases 2 et 4), nous permettront par leur complexité réduite par rapport aux autres bases, de déduire des architectures parallèles implémentant la FFT. Ces algorithmes auront une forme récursive qui leur permettra d'être calculés sur place par un réseau de PEs en structure parallèle.
- Le deuxième chapitre sera le siège de développements de différentes architectures parallèles très performantes de la FFT. Les résultats obtenus dans le premier chapitre formeront la base de départ de ces différents développements. La performance des architectures développées sera mesurée selon deux facteurs : le temps de réponse et le nombre d'opérateurs utilisés qui est directement lié à la surface de silicium utilisée lors de l'implantation.
- Le troisième chapitre sera consacré à l'implémentation de la TFD sur des réseaux systoliques très performants. Tout d'abord nous présenterons les réseaux systoliques avec leurs propriétés et leurs caractéristiques, puis nous donnerons des exemples d'application pour montrer toute leur puissance de calcul. Ces exemples, qui ne sont pas donnés fortuitement, nous permettront d'ailleurs de présenter dans ce même chapitre des implémentations de la TFD et dans le quatrième chapitre des implémentations des filtres numériques IIR.
- Le quatrième et dernier chapitre sera exclusivement réservé aux architectures systoliques dédiées aux filtres numériques IIR. Les facteurs régularité et localité, synonymes de puissance de calcul et aptitude à une implantation VLSI seront recherchés.

Une conclusion générale, avec les remarques nécessaires et des perspectives valables, parachèvera notre travail.

Chapitre I

Transformée de Fourier Rapide en bases *B*, 2 et 4

1. Introduction

Joseph FOURIER, mathématicien français, affirma dans un mémoire daté de 1807 qu'il était possible, dans certaines conditions, de décomposer une fonction périodique sous la forme d'une somme infinie de signaux sinusoïdaux. Cette somme infinie, appelée communément série de Fourier, représente donc une fonction périodique définie sur l'axe complet des réels. Cependant, si la fonction n'est pas périodique, donc non représentable par une série de Fourier, elle peut alors être représentée par une intégrale faisant appel aux signaux sinusoïdaux et est appelée dans ce cas Transformée de Fourier. Donc, la transformée de Fourier est considérée comme l'extension, pour les fonctions non périodiques, du développement en série de Fourier des fonctions périodiques.

Dans la pratique, on est confronté à l'évaluation de la transformée de Fourier sur des calculateurs numériques, qui sont limités en espace mémoire et en temps d'exécution. Par conséquent, l'introduction de la Transformée de Fourier Discrète, notée TFD (en anglais DFT pour Discrete Fourier Transform) [3, 40], qui travaille sur un nombre fini de valeurs ou d'échantillons, est tout à fait indiquée.

La transformée de Fourier peut être définie pour une fonction ayant une variable (c'est le cas le plus courant) ou plusieurs. Le nombre de variables indépendantes d'une fonction est égal à la dimension de la transformée de Fourier qui peut lui être associée. La transformée de Fourier la plus connue et la plus utilisée est la transformée unidimensionnelle qui associe à une fonction temporelle une fonction fréquentielle. Son utilisation dans l'analyse spectrale, le filtrage, les télétransmissions et télécommunications numériques, le radar et le sonar ne sont souvent limitées que par la puissance de calcul à mettre en œuvre [1 – 3, 40]. Les transformées multidimensionnelles les plus courantes sont appliquées à un plan ou à l'espace géométrique, en conjonction éventuelle avec le temps pour faire intervenir la dynamique temporelle du système étudié [41]. On peut citer par exemple le traitement des images en général [42 – 44], la tomographie et le scanner en médecine en particulier [45, 46].

Le seul problème à l'emploi des transformées de Fourier ayant un très grand nombre d'échantillons à traiter, est le nombre élevé des calculs à effectuer. Heureusement, l'avancée rapide technologique qui a permis l'arrivée d'ordinateurs très puissants et le développement d'algorithmes de calcul rapide ont pu remédier à ce problème en repoussant toujours plus loin les limites liées au nombre d'échantillons à traiter.

Dans ce chapitre, les définitions relatives à la transformée de Fourier et à la transformée de Fourier discrète sont données. Le Calcul de la TFD par trois algorithmes rapides, l'un par entrelacement fréquentiel en base arbitraire *B* et les deux autres par entrelacement temporel respectivement en bases 2 et 4, sous forme de séries arithmétiques, seront développés et présentés de manière à faire résulter des architectures parallèles pouvant être implantés sur silicium.

2. Définitions

2.1. La transformée de Fourier unidimensionnelle

La transformée de Fourier peut être, suivant son domaine d'application, unidimensionnelle, bidimensionnelle, ou dans le cas général multidimensionnelle. Les transformées de Fourier les plus courantes sont unidimensionnelle et bidimensionnelle. Dans le cas unidimensionnel, la transformée de Fourier directe et son inverse sont données respectivement par les formules (I.1) et (I.2) (t et f représentent respectivement les composantes temps et fréquence) :

$$X(f) = \int_{-\infty}^{+\infty} x(t).e^{-2j\pi ft}.dt \tag{I.1}$$

et

$$x(t) = \int_{-\infty}^{+\infty} X(f).e^{2j\pi ft}.dt \tag{I.2}$$

Alors que dans le cas bidimensionnel, la transformée de Fourier directe et son inverse sont donnés respectivement par les formules (I.3) et (I.4) :

$$X(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x(y,z).e^{-2\pi j(yu+zv)}\,dy.dz \tag{I.3}$$

et la transformée de Fourier inverse par :

$$x(y,z) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} X(u,v).e^{2\pi j(yu+zv)}\,du.dv \tag{I.4}$$

où *y* et *z* représentent généralement les coordonnées géométriques d'un point *x* appartenant à un plan, et *u* et *v* les coordonnées fréquentielles de sa transformée de Fourier *X*.

2.2. La transformée de Fourier discrète (TFD)

Puisque l'évaluation de la transformée de Fourier se réalise sur des calculateurs numériques, les variables à traiter doivent être discrètes et normalisées. Dans le cas unidimensionnel par exemple, l'échantillonnage du signal *x(t)* à une fréquence adéquate permet de substituer à celui-ci un signal à temps discret, c'est-à-dire une suite de valeurs

finies. En limitant le nombre d'échantillons à *N*, la transformée de Fourier discrète unidimensionnelle, notée tout simplement TFD, peut donc s'écrire sous la forme d'une sommation discrète comme suit :

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n).W_N^{kn} \quad ; \textit{pour } k : 0,1,\cdots,N-1 \tag{I.5}$$

où :

- *n* et *k* sont les variables discrètes et normalisées respectivement des domaines original transformé ;
- *N* est le nombre des valeurs discrètes et successives des variables *n* et *k* ;
- *x(n)* est la fonction originale et *X(k)* la fonction transformée de *x(n)* ;
- $W_N^{kn} = e^{-2j\pi kn/N}$ (en posant $W_N = e^{-2j\pi/N}$) sont les coefficients de rotation ou de transformation de la TFD, appelés « twiddle factors » en anglais. Ils possèdent les propriétés importantes suivantes :

  1) Périodicité : $W_N^{(\alpha N+n)k} = W_N^{(\alpha N+k)n} = W_N^{nk}$ pour α : entier. $(W_N^{\alpha.N} = 1)$

  2) Conjugaison : $W_N^{(\alpha N-n)k} = W_N^{(\alpha N-k)n} = (W_N^{nk})^*$

  3) Orthogonalité : $\sum_{k=0}^{N-1} W_N^{(n'-n)k} = \begin{cases} N & si \quad n'-n = \alpha N \\ 0 & si \quad n'-n \neq \alpha N \end{cases}$

  4) $W_{R.N}^{R.n} = W_N^n$ pour tout réel $R \neq 0$

  5) $W_N^{n+\frac{N}{4}} = -jW_N^n \qquad W_N^{n+\frac{N}{2}} = -W_N^n \qquad W_N^{n+\frac{3N}{4}} = jW_N^n$

Sous forme matricielle, la TFD s'écrit :

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ \vdots \\ X(N-1) \end{pmatrix} = \frac{1}{N} \times \begin{pmatrix} 1 & 1 & 1 & \cdots & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & \cdots & W_N^{(N-1)(N-1)} \end{pmatrix} \times \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ \vdots \\ x(N-1) \end{pmatrix} \tag{I.6}$$

ou sous une forme plus allégée :

$$X = \frac{1}{N} \times T_N \times x \tag{I.7}$$

où *x* et *X* sont respectivement les vecteurs original et transformé, et T_N la matrice carrée (*N*x*N*) de transformation qui a la particularité d'être une matrice symétrique.

D'après les formules (I.5) et (I.6)), on peut dire qu'à l'exception du facteur multiplicatif 1/*N* et du calcul des puissances de W_N, chaque composante *k* nécessite *N*

multiplications et (N-1) additions complexes, soit pour toutes les composantes, N^2 multiplications et $N(N\text{-}1)$ additions complexes. On dit que la TFD est de complexité $O(N^2)$, ce qui est immense quand le nombre d'échantillons N est grand.

La transformée de Fourier discrète unidimensionnelle inverse, notée TFDI, est donnée par :

$$x(n) = \sum_{k=0}^{N-1} X(k).W_N^{-nk} \tag{I.8}$$

Sous forme matricielle la TFDI a la même forme que la TFD, à la différence près qu'il faut retirer le scalaire $1/N$ et remplacer W_N^{kn} par W_N^{-kn}.

Dans le cas bidimensionnel, où le signal $x(n_1, n_2)$ est représenté par une matrice et en prenant le nombre d'échantillons égal à N suivant les deux dimensions considérées, la transformée de Fourier discrète bidimensionnelle, est donnée par :

$$X(k_1, k_2) = \frac{1}{N^2} \sum_{n_2=0}^{N-1} \sum_{n_1=0}^{N-1} x(n_1, n_2).W_N^{\sum_{j=1}^{2} k_j n_j} \quad ; avec \;\; k_1 \; et \; k_2 : 1,0,\cdots,N-1 \tag{I.9}$$

et la transformée de Fourier discrète bidimensionnelle inverse par :

$$x(n_1, n_2) = \sum_{k_2=0}^{N-1} \sum_{k_1=0}^{N-1} X(k_1, k_2).W_N^{\sum_{j=1}^{2} -k_j n_j} \tag{I.10}$$

3. Développement de la transformée de Fourier rapide (TFR)

3.1. Généralités

Dans la plupart des cas le calcul direct de la TFD fait intervenir un grand nombre d'opérations complexes. Ce nombre est d'autant plus grand que le nombre d'échantillons N est élevé. Comme la TFD est de complexité $O(N^2)$, ce qui est très élevé, il serait donc indispensable de trouver des méthodes qui permettent de réduire cette complexité.

La TFD sous ses différentes formes d'écriture présente des particularités très intéressantes. En regardant par exemple la matrice carrée de transformation T_N (formule (I.6)) on remarque que les lignes et les colonnes de mêmes indices ont les mêmes éléments et ces éléments sont des puissances du nombre de base W_N qui a aussi ses propres spécificités (propriétés de périodicité, conjugaison, etc…). Des simplifications importantes peuvent alors être envisagées, aboutissant à des algorithmes rapides [39, 47 – 61] désignés sous un nom commun : **Transformée de Fourier Rapide**, notée **TFR** (**FFT :** Fast Fourier Transform, en anglais). La TFR est donc un algorithme efficace pour le calcul de la TFD ou la TFDI.

L'idée générale d'une TFR, bien que déjà connue auparavant, a été vulgarisée par un algorithme présenté dans une publication de J. W. Cooley et J. W. Tukey en 1965 [39]. Cet algorithme de TFR, connu généralement sous le nom d'Algorithme de Cooley-Tukey (ACT),

est de loin et jusqu'à présent, avec ses variantes, le plus utilisé dans les calculs de la TFD. Il s'appuie, dans sa version de base, sur le principe du 'diviser pour régner'. Il décompose, d'une manière récursive une TFD de taille $N=N_1.N_2$ en deux TFDs plus réduites de tailles N_1 et N_2, menant ainsi à une réduction importante du nombre d'opérations complexes à exécuter.

Différents algorithmes basés sur des principes plus ou moins semblables ont ensuite été proposés, créant ainsi tout un ensemble de TFRs [52, 59 – 61]. Les formes les plus connues, issues de l'ACT, sont celles où la taille N est une puissance de 2 ou de 4, car les calculs impliqués sont relativement simples. Toutefois, cet algorithme peut être généralisé pour une base B arbitraire. La transformée globale est alors décomposée récursivement en des transformées minimales sur B valeurs. Cela conduit à (N/B) calculs de base répétés sur $log_B(N)$ étapes, chacun d'entre eux étant calculé par une cellule ou processeur élémentaire si une implantation sur silicium est à considérer. Il faut rappeler que chaque calcul élémentaire d'une étape de la TFR dépend des seules données fournies par l'étape précédente, ce qui entraîne qu'une cellule de base est indépendante de ses semblables de même rang, et qu'une TFR peut donc être parallélisée et calculée sur place lors de son implantation. Cela se verra sur les différentes structures parallèles proposées dans le chapitre suivant et qui vont être déduites des résultats des algorithmes développés dans ce qui suit de ce chapitre.

L'établissement des algorithmes de TFR sont le plus souvent abordés selon l'une des deux manières suivantes : l'utilisation de la représentation matricielle de la définition de la TFD (équation (I.6)), ou bien l'emploi de celle-ci sous forme de séries arithmétiques. La première démarche citée traite l'ensemble des données et est adaptée pour une réalisation logicielle. La seconde approche ne décrit qu'une composante singulière, mais permet une approche adéquate pour une implantation matérielle. Ces deux approches peuvent être traitées en utilisant l'une des deux bien connues variantes : la TFR avec entrelacement temporel [39], ou la TFR avec entrelacement fréquentiel [52]. L'approche sous forme de séries arithmétiques sera étudiée dans ce qui suit car, comme mentionnée juste avant, elle est plus proche dans son développement à la structure d'une implantation matérielle, permettant ainsi un calcul sur place de la TFD.

3.2. L'algorithme conventionnel de Cooley–Tukey avec entrelacement fréquentiel (E.F.) pour le calcul de la TFR en base arbitraire *B*

L'idée de départ, dans le cas de l'utilisation de séries arithmétiques, consiste à réécrire différemment les indices des composantes $x(n)$ et $X(k)$ dans l'équation de la TFD donnée par (I.5) [49]. L'équation (I.5) est réécrite ci-dessous sans le facteur multiplicatif $1/N$ qui est

commun pour toutes les composantes $X(k)$ et donc omis dans le développement qui suit parce qu'il n'y intervient pas :

$$X(k) = \sum_{n=0}^{N-1} x(n).W_N^{kn} \qquad ; pour\ k : 0,1,\cdots,N-1 \tag{I.11}$$

où $W_N = \exp(-2j\pi/N)$ et $j = \sqrt{-1}$.

Si N est supposé être une puissance entière d'un nombre positif B, c'est-à-dire $N = B^d$ donc $d = log_B(N)$, alors nous pouvons utiliser la table des indices de l'entrelacement fréquentiel (E.F) en base B qui consiste à modifier l'indice d'entrée n dans (I.11) par

$$n + q\frac{N}{B}, \quad 0 \le n \le \frac{N}{B} - 1, \quad 0 \le q \le B-1 \tag{I.12}$$

La TFD donnée en (I.11) peut alors être exprimée par

$$\begin{aligned} X(k) = \sum_{n=0}^{(N/B)-1} \Bigg[& x(n) + x\left(n + \frac{N}{B}\right)W_B^k + x\left(n + 2\frac{N}{B}\right)W_B^{2k} + \quad \cdots \\ & \cdots + x\left(n + (B-2)\frac{N}{B}\right)W_B^{(B-2)k} + x\left(n + (B-1)\frac{N}{B}\right)W_B^{(B-1)k} \Bigg] W_N^{kn} \\ & k : 0,1,\cdots,N-1 \end{aligned} \tag{I.13}$$

Comme pour l'indice d'entrée n, nous pouvons utiliser la table des indices pour l'entrelacement fréquentiel en base B pour changer l'indice de sortie k dans (I.13) par :

$$Bk + p, \qquad 0 \le k \le \frac{N}{B} - 1, \qquad 0 \le p \le B-1 \tag{I.14}$$

En utilisant (I.14) et le fait que $W_N^{Rnk} = W_{N/R}^{nk}$, l'équation (I.13) peut être réarrangée comme suit :

$$\begin{aligned} X(Bk+p) = \sum_{n=0}^{(N/B)-1} \Bigg\{ \Bigg[& x(n) + x\left(n + \frac{N}{B}\right)W_B^p + x\left(n + 2\frac{N}{B}\right)W_B^{2p} + \quad \cdots \\ & \cdots \quad + x\left(n + (B-2)\frac{N}{B}\right)W_B^{(B-2)p} + x\left(n + (B-1)\frac{N}{B}\right)W_B^{(B-1)p} \Bigg] W_N^{pn} \Bigg\} W_{N/B}^{kn} \\ & 0 \le k \le \frac{N}{B} - 1, \qquad 0 \le p \le B-1 \end{aligned} \tag{I.15}$$

Une forme compacte de (I.15) peut être obtenue comme suit :

$$X(Bk+p) = \sum_{n=0}^{(N/B)-1} \left\{ g_p(n) W_N^{pn} \right\} W_{N/B}^{kn} \qquad 0 \le k \le \frac{N}{B} - 1, \quad 0 \le p \le B-1 \tag{I.16}$$

où

$$g_p(n) = \sum_{q=0}^{B-1} x(n + q\frac{N}{B}) W_B^{pq}, \qquad 0 \le p \le B-1 \tag{I.17}$$

La forme matricielle de (I.17) est donnée par :

$$\begin{bmatrix} g_0(n) \\ g_1(n) \\ g_2(n) \\ \vdots \\ g_{B\text{-}1}(n) \end{bmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_B^1 & W_B^2 & \cdots & W_B^{B\text{-}1} \\ 1 & W_B^2 & W_B^4 & \cdots & W_B^{2(B\text{-}1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_B^{B\text{-}1} & W_B^{2(B\text{-}1)} & \cdots & W_B^{(B\text{-}1)(B\text{-}1)} \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/B) \\ x(n+2N/B) \\ \vdots \\ x(n+(B\text{-}1)N/B) \end{bmatrix} \tag{I.18}$$

L'équation (I.18) peut être écrite sous une forme réduite comme suit :

$$\mathbf{G}_n = \mathbf{W}_B \mathbf{X}_n \tag{I.19}$$

où, comme vu dans (I.18), $\mathbf{W}_B$ est l'opérateur matriciel de la TFD de longueur *B*. Les composantes des vecteurs d'entrée et de sortie $\mathbf{X}_n$ and $\mathbf{G}_n$ de la relation (I.19) sont liées aux séquences d'entrée et de sortie de la relation (I.17) par $X_n(q) = x(n + q\frac{N}{B})$ et $G_n(p) = g_p(n)$, respectivement. Maintenant, en posant:

$$y_p(n) = g_p(n) W_N^{pn}, \quad 0 \le p \le B-1 \tag{I.20}$$

et en utilisant (I.18), on obtient:

$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ \vdots \\ y_{B\text{-}1}(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & W_N^n & 0 & \cdots & 0 \\ 0 & 0 & W_N^{2n} & 0 & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & W_N^{(B\text{-}1)n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_B^1 & W_B^2 & \cdots & W_B^{B\text{-}1} \\ 1 & W_B^2 & W_B^4 & \cdots & W_B^{2(B\text{-}1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_B^{B\text{-}1} & W_B^{2(B\text{-}1)} & \cdots & W_B^{(B\text{-}1)(B\text{-}1)} \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/B) \\ x(n+2N/B) \\ \vdots \\ x(n+(B\text{-}1)N/B) \end{bmatrix} \tag{I.21}$$

L'équation (I.21) représente la forme matricielle du papillon de l'algorithme FFT classique en base *B*, basé sur l'approche de Cooley-Tukey. Il peut également être exprimé sous une forme compacte telle que :

$$\mathbf{Y}_n = \mathbf{T}_n^{\text{conv}} \mathbf{W}_B \mathbf{X}_n \tag{I.22}$$

où les composantes du vecteur de sortie $\mathbf{Y}_n$ sont liées à la séquence de sortie de (I.20) par

$$Y_n(p) = y_p(n)$$

$\mathbf{T}_n^{\text{conv}}$ est la matrice diagonale des facteurs de rotation *W* représentée en (I.21) et qui peut être exprimée comme suit :

$$\mathbf{T}_n^{\text{conv}} = diag(1, W_N^n, W_N^{2n}, \cdots, W_N^{(B-2)n}, W_N^{(B-1)n}) \tag{I.23}$$

Enfin, (I.15) peut être réécrite sous la forme :

$$X(Bk+p) = \sum_{n=0}^{(N/B)-1} y_p(n) W_{N/B}^{kn} \qquad 0 \le k \le \frac{N}{B} - 1, \quad 0 \le p \le B-1 \tag{I.24}$$

qui est l'expression familière d'une TFD de longueur *N*/*B*.

De ce fait, nous notons qu'à partir de (I.15), (I.16) ou (I.24), la TFD donnée en (I.11) a été décomposée dans la première étape en *B* TFDs de longueur (*N*/*B*), une pour chaque valeur

de *p*. La séquence d'entrée de chacune de ces TFDs est donnée en (I.17). L'ensemble de ces *B* TFDs de longueur (*N*/*B*) constituent la TFD initiale à *N* points. Le processus de décomposition est répété dans la deuxième étape en divisant chacune de ces TFDs de longueur (*N*/*B*) en *B* TFDs de longueur (N/B^2). Ces TFDs de longueur (N/B^2) sont ensuite divisées à l'étape suivante en *B* TFDs de longueur (N/B^3), et ainsi de suite jusqu'à ce que la décimation finale produise des TFDs de longueur *B*. Le calcul total de la TFD nécessite $d = log_B(N)$ étapes.

Pour illustrer la démonstration précédente, nous montrons dans les exemples suivants les calculs impliqués dans le premier étage d'une FFT de longueur *N* en base *B* ($N = B^d$) pour différentes valeurs de *B*.

Exemple 1 : L'algorithme FFT proposé en base 3 pour une TFD à *N* points, où $N=3^d$

$$X(3k) = \sum_{n=0}^{(N/3)-1} y_0(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

$$X(3k+1) = \sum_{n=0}^{(N/3)-1} y_1(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

$$X(3k+2) = \sum_{n=0}^{(N/3)-1} y_2(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

où
$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & W_N^n & 0 \\ 0 & 0 & W_N^{2n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & W_3^1 & W_3^2 \\ 1 & W_3^2 & W_3^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/3) \\ x(n+2N/3) \end{bmatrix}$$

Exemple 2 : L'algorithme FFT proposé en base 4 pour une TFD à *N* points, où $N=4^d$.

$$X(4k) = \sum_{n=0}^{(N/4)-1} y_0(n) W_{N/4}^{kn} \quad 0 \le k \le \frac{N}{4} - 1$$

$$X(4k+1) = \sum_{n=0}^{(N/4)-1} y_1(n) W_{N/4}^{kn} \quad 0 \le k \le \frac{N}{4} - 1$$

$$X(4k+2) = \sum_{n=0}^{(N/4)-1} y_2(n) W_{N/4}^{kn} \quad 0 \le k \le \frac{N}{4} - 1$$

$$X(4k+3) = \sum_{n=0}^{(N/4)-1} y_3(n) W_{N/4}^{kn} \quad 0 \le k \le \frac{N}{4} - 1$$

où
$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ y_3(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 \\ 0 & 0 & 0 & W_N^{3n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/4) \\ x(n+2N/4) \\ x(n+3N/4) \end{bmatrix}$$

puisque $W_4^1 = -j \quad W_4^2 = -1 \quad W_4^3 = j \quad W_4^4 = 1$

Exemple 3 : L'algorithme FFT proposé en base 5 pour une TFD à N points, où $N=5^d$

$$X(5k) = \sum_{n=0}^{(N/5)-1} y_0(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+1) = \sum_{n=0}^{(N/5)-1} y_1(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+2) = \sum_{n=0}^{(N/5)-1} y_2(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+3) = \sum_{n=0}^{(N/5)-1} y_3(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+4) = \sum_{n=0}^{(N/5)-1} y_4(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

où
$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ y_3(n) \\ y_4(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 & 0 \\ 0 & 0 & 0 & W_N^{3n} & 0 \\ 0 & 0 & 0 & 0 & W_N^{4n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W_5^1 & W_5^2 & W_5^3 & W_5^4 \\ 1 & W_5^2 & W_5^4 & W_5^1 & W_5^3 \\ 1 & W_5^3 & W_5^1 & W_5^4 & W_5^2 \\ 1 & W_5^4 & W_5^3 & W_5^2 & W_5^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/5) \\ x(n+2N/5) \\ x(n+3N/5) \\ x(n+4N/5) \end{bmatrix}$$

Exemple 4 : L'algorithme FFT proposé en base 6 pour une TFD à N points, où $N=6^d$.

$$X(6k) = \sum_{n=0}^{(N/6)-1} y_0(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+1) = \sum_{n=0}^{(N/6)-1} y_1(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+2) = \sum_{n=0}^{(N/6)-1} y_2(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+3) = \sum_{n=0}^{(N/6)-1} y_3(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+4) = \sum_{n=0}^{(N/6)-1} y_4(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+5) = \sum_{n=0}^{(N/6)-1} y_5(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

où
$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ y_3(n) \\ y_4(n) \\ y_5(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_N^{3n} & 0 & 0 \\ 0 & 0 & 0 & 0 & W_N^{-2n} & 0 \\ 0 & 0 & 0 & 0 & 0 & W_N^{-n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_6^1 & W_6^2 & W_6^3 & W_6^4 & W_6^5 \\ 1 & W_6^2 & W_6^4 & 1 & W_6^2 & W_6^4 \\ 1 & W_6^3 & 1 & W_6^3 & 1 & W_6^3 \\ 1 & W_6^4 & W_6^2 & 1 & W_6^4 & W_6^2 \\ 1 & W_6^5 & W_6^4 & W_6^3 & W_6^2 & W_6^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/6) \\ x(n+2N/6) \\ x(n+3N/6) \\ x(n+4N/6) \\ x(n+5N/6) \end{bmatrix}$$

3.3. L'algorithme conventionnel de Cooley–Tukey avec E.F. amélioré pour le calcul de la TFR en base arbitraire *B*

L'algorithme donné ici et qu'on a développé dans [49] est basé sur l'approche de Cooley-Tukey présentée dans la section ci-dessus, mais avec une différence dans le processus d'indexation de presque la moitié des sous-TFDs. Cela permettra de réduire, dans la même proportion, le nombre de calculs des facteurs de rotation *W* ou bien leur lecture à partir d'une table de recherche (en anglais 'lookup table'). Deux cas différents peuvent survenir en fonction de la parité de la base *B*. Ces deux cas doivent être étudiés séparément en raison de leurs particularités.

3.3.1. Cas pour une base *B* impaire

Les premières $\frac{B+1}{2}$ sous-TFDs, c'est à dire $0 \le p \le \frac{B-1}{2}$, sont calculées selon (I.16) ou (I.24), et réécrites ici avec de nouvelles limites de *p*.

$$X(Bk+p) = \sum_{n=0}^{(N/B)-1} \left\{ g_p(n) W_N^{pn} \right\} W_{N/B}^{kn} = \sum_{n=0}^{(N/B)-1} y_p(n) W_{N/B}^{kn}$$

$$0 \le k \le \frac{N}{B} - 1, \quad p = 0,1,\cdots,\frac{B-1}{2} \tag{I.25}$$

Pour les sous-TFDs restantes, c'est-à-dire $\frac{B+1}{2} \le p \le B-1$, une nouvelle indexation, qui tire profit de la périodicité des facteurs de rotation *W*, tout en générant les mêmes sous-TFDs, est introduite comme suit :

$$X((N+Bk-p) \bmod N) = \sum_{n=0}^{(N/B)-1} \left\{ g_{-p}(n) W_N^{-pn} \right\} W_{N/B}^{kn} = \sum_{n=0}^{(N/B)-1} y_{-p}(n) W_{N/B}^{kn}$$

$$0 \le k \le \frac{N}{B} - 1 \qquad p = \frac{B-1}{2}, \frac{B-1}{2} - 1, \frac{B-1}{2} - 2, \cdots, 1 \tag{I.26}$$

Il est à noter que la notation de l'index *p* est la même afin d'éviter d'en introduire une nouvelle. Des relations (I.25) et (I.26), nous remarquons qu'une nouvelle matrice de facteurs de rotation *W*, notée $\mathbf{T}_n^{\text{new-odd}}$ et spécifiée par :

$$\mathbf{T}_n^{\text{new-odd}} = diag\left(1, W_N^n, W_N^{2n}, \cdots, W_N^{\frac{R-1}{2}n}, W_N^{-\frac{R-1}{2}n}, \cdots, W_N^{-2n}, W_N^{-n}\right) \tag{I.27}$$

a remplacé la matrice conventionnelle donnée en (I.23), et par conséquent, le papillon de la FFT conventionnelle en base *B* donné en (I.22) devient

$$\mathbf{Y}_n = \mathbf{T}_n^{\text{new-odd}} \mathbf{W}_B \mathbf{X}_n \tag{I.28}$$

Maintenant, en utilisant le fait que $W_N^{pn} = \cos\left(\frac{2\pi}{N} pn\right) - j\sin\left(\frac{2\pi}{N} pn\right)$ et $W_N^{-pn} = \cos\left(\frac{2\pi}{N} pn\right) + j\sin\left(\frac{2\pi}{N} pn\right)$ dans les éléments de $\mathbf{T}_n^{\text{new-odd}}$ donnés par (I.27), seulement $(B-1)$ facteurs de rotation réels (cos() et sin()) doivent être calculés ou chargés à partir de la 'look-up table' pendant le traitement du papillon spécifié par (I.28) de l'algorithme proposé au lieu des $(2(B-1))$ nécessaires dans le papillon conventionnel donné par (I.22). En conséquence, un gain en calculs d'exactement 50% est réalisé en utilisant l'algorithme proposé. Il faut remarquer que lorsque la 'look-up table' est utilisée, un gain similaire est obtenu dans la génération des adresses.

Il convient de noter aussi que la nouvelle indexation proposée $(N + Bk - p) \bmod N$ **,** n'ajoute aucune complexité de calcul par rapport à la complexité conventionnelle $(Bk + p)$, car

$$(N + Bk - p) \bmod N = \begin{cases} N - p, & \text{pour } k = 0 \\ Bk - p, & \text{ailleurs} \end{cases} \quad 1 \le p \le \frac{B-1}{2} \tag{I.29}$$

Pour illustrer la démonstration précédente, nous montrons dans les deux exemples suivants les calculs impliqués dans le premier étage d'une FFT de longueur *N* en base *B* impaire ($N = B^d$).

Exemple 5 : L'algorithme FFT proposé en base 3 pour une TFD à *N* points, où $N=3^d$

$$X(3k) = \sum_{n=0}^{(N/3)-1} y_0(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

$$X(3k+1) = \sum_{n=0}^{(N/3)-1} y_1(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

$$X((N+3k-1) \bmod N) = \sum_{n=0}^{(N/3)-1} y_{-1}(n) W_{N/3}^{kn} \quad 0 \le k \le \frac{N}{3} - 1$$

où $$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_{-1}(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & W_N^n & 0 \\ 0 & 0 & W_N^{-n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & W_3^1 & W_3^2 \\ 1 & W_3^2 & W_3^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/3) \\ x(n+2N/3) \end{bmatrix}$$

Exemple 6 : L'algorithme FFT proposé en base 5 pour une TFD à *N* points, où $N=5^d$

$$X(5k) = \sum_{n=0}^{(N/5)-1} y_0(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+1) = \sum_{n=0}^{(N/5)-1} y_1(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X(5k+2) = \sum_{n=0}^{(N/5)-1} y_2(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X((N+5k-2) \bmod N) = \sum_{n=0}^{(N/5)-1} y_{-2}(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

$$X((N+5k-1) \bmod N) = \sum_{n=0}^{(N/5)-1} y_{-1}(n) W_{N/5}^{kn} \quad 0 \le k \le \frac{N}{5} - 1$$

où
$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ y_{-2}(n) \\ y_{-1}(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 & 0 \\ 0 & 0 & 0 & W_N^{-2n} & 0 \\ 0 & 0 & 0 & 0 & W_N^{-n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W_5^1 & W_5^2 & W_5^3 & W_5^4 \\ 1 & W_5^2 & W_5^4 & W_5^1 & W_5^3 \\ 1 & W_5^3 & W_5^1 & W_5^4 & W_5^2 \\ 1 & W_5^4 & W_5^3 & W_5^2 & W_5^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/5) \\ x(n+2N/5) \\ x(n+3N/5) \\ x(n+4N/5) \end{bmatrix}$$

Afin de voir le nombre exact des réductions réalisées en termes de calcul des composantes réelles des facteurs *W* ou de leurs lectures à partir de la 'look-up table' par l'algorithme proposé par rapport à l'ACT, nous montrons dans le tableau I.1 différentes TFDs de taille *N* traitées par les deux algorithmes avec différentes bases impaires (*B* = 3, *B* = 5 et *B* = 7). Les bases 3 et 5 sont fréquemment utilisées dans la conception de noyaux FFT ou de processeurs FFT [24 – 26]. Pour de grands *N*, les réductions sont importantes. La taille de la mémoire de la 'look-up table' est réduite de moitié dans les processeurs dédiés FFT.

	ACT en base *B*			Algorithme proposé en base *B*		
N \ *B*	3	5	7	3	5	7
9	8			4		
25		32			16	
27	56			28		
49			72			36
81	272			136		
125		352			176	
243	1136			568		
343			1080			540
625		2752			1376	
2401			11664			5832
3125		18752			9376	
16807			110448			55224

Tableau I.1: Comparaison en nombres de calculs des composantes réelles des facteurs *W* ou de leurs lectures à partir de la 'look-up table' entre l'ACT et l'algorithme proposé (cas des bases impaires).

3.3.2. Cas pour une base *B* paire

Pour une base paire, la procédure est la même que pour le cas d'une base impaire illustrée dans la sous-section précédente, à l'exception des limites de *p*. Les premiers $(\frac{B}{2}+1)$ sous-TFDs correspondant à $0 \le p \le \frac{B}{2}$ sont calculées selon les relations (I.16) ou (I.24), et réécrites ici avec de nouvelles limites de *p*.

$$X(Bk+p) = \sum_{n=0}^{(N/B)-1} \left\{ g_p(n) W_N^{pn} \right\} W_{N/B}^{kn} = \sum_{n=0}^{(N/B)-1} y_p(n) W_{N/B}^{kn}$$

$$0 \le k \le \frac{N}{B}-1, \quad p = 0,1,\cdots,\frac{B}{2} \qquad \text{(I.30)}$$

Les sous-TFDs restantes (c.-à-d. $\frac{B}{2}+1 \le p \le B-1$) sont calculées avec une nouvelle indexation comme suit :

$$X((N+Bk-p)\bmod N)=\sum_{n=0}^{(N/B)-1}\left\{g_{-p}(n)W_N^{-pn}\right\}W_{N/B}^{kn}=\sum_{n=0}^{(N/B)-1}y_{-p}(n)W_{N/B}^{kn}$$

$$0\le k\le\frac{N}{B}-1\qquad p=\frac{B}{2}-1,\frac{B}{2}-2,\cdots,1 \tag{I.31}$$

La nouvelle matrice des facteurs *W* qui est ($\mathbf{T}_n^{\text{new-even}}$) est alors donnée par

$$\mathbf{T}_n^{\text{new-even}}=diag(1,W_N^n,W_N^{2n},\cdots,W_N^{\left(\frac{B}{2}-1\right)n},W_N^{\left(\frac{B}{2}\right)n},W_N^{-\left(\frac{B}{2}-1\right)n},\cdots,W_N^{-2n},W_N^{-n}) \tag{I.32}$$

et l'expression du papillon de l'algorithme proposé pour une base *B* paire est :

$$\mathbf{Y}_n=\mathbf{T}_n^{\text{new-even}}\mathbf{W}_B\mathbf{X}_n \tag{I.33}$$

Avec $W_N^{pn}=\cos\left(\frac{2\pi}{N}pn\right)-j\sin\left(\frac{2\pi}{N}pn\right)$ et $W_N^{-pn}=\cos\left(\frac{2\pi}{N}pn\right)+j\sin\left(\frac{2\pi}{N}pn\right)$, et en regardant la structure de $\mathbf{T}_n^{\text{new-even}}$ on remarque que seulement *B* composantes réelles des facteurs *W* doivent être évaluées ou lues à partir de la 'look-up table' au cours du traitement du papillon spécifié par (I.33) de l'algorithme proposé au lieu des $\left(2\left(B-1\right)\right)$ nécessaires dans le papillon de l'ATC. Cela implique un gain en calculs d'exactement $\left(100\left(1-\frac{R}{2\left(R-1\right)}\right)\right)\%$.

Ce gain tend vers les 50% pour les FFTs à bases plus élevées. Illustrons maintenant, dans les exemples qui suivent, les calculs impliqués dans la première étape de l'algorithme FFT proposé pour les bases 4 et 6 pour une TFD à *N* points.

Exemple 7 : L'algorithme FFT proposé en base 4 pour une TFD à *N* points, où $N=4^d$.

$$X(4k)=\sum_{n=0}^{(N/4)-1}y_0(n)W_{N/4}^{kn}\quad 0\le k\le\frac{N}{4}-1$$

$$X(4k+1)=\sum_{n=0}^{(N/4)-1}y_1(n)W_{N/4}^{kn}\quad 0\le k\le\frac{N}{4}-1$$

$$X(4k+2)=\sum_{n=0}^{(N/4)-1}y_2(n)W_{N/4}^{kn}\quad 0\le k\le\frac{N}{4}-1$$

$$X((N+4k-1)\bmod N)=\sum_{n=0}^{(N/4)-1}y_{-1}(n)W_{N/4}^{kn}\quad 0\le k\le\frac{N}{4}-1$$

où
$$\begin{bmatrix}y_0(n)\\y_1(n)\\y_2(n)\\y_{-1}(n)\end{bmatrix}=\begin{pmatrix}1&0&0&0\\0&W_N^n&0&0\\0&0&W_N^{2n}&0\\0&0&0&W_N^{-n}\end{pmatrix}\begin{pmatrix}1&1&1&1\\1&-j&-1&j\\1&-1&1&-1\\1&j&-1&-j\end{pmatrix}\begin{bmatrix}x(n)\\x(n+N/4)\\x(n+2N/4)\\x(n+3N/4)\end{bmatrix}$$

Exemple 8 : L'algorithme FFT proposé en base 6 pour une TFD à *N* points, où $N=6^d$.

$$X(6k)=\sum_{n=0}^{(N/6)-1}y_0(n)W_{N/6}^{kn}\quad 0\le k\le\frac{N}{6}-1$$

$$X(6k+1)=\sum_{n=0}^{(N/6)-1}y_1(n)W_{N/6}^{kn}\quad 0\le k\le\frac{N}{6}-1$$

$$X(6k+2) = \sum_{n=0}^{(N/6)-1} y_2(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X(6k+3) = \sum_{n=0}^{(N/6)-1} y_3(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X((N+6k-2) \bmod N) = \sum_{n=0}^{(N/6)-1} y_{-2}(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

$$X((N+6k-1) \bmod N) = \sum_{n=0}^{(N/6)-1} y_{-1}(n) W_{N/6}^{kn} \quad 0 \le k \le \frac{N}{6} - 1$$

où

$$\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ y_3(n) \\ y_{-2}(n) \\ y_{-1}(n) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_N^{3n} & 0 & 0 \\ 0 & 0 & 0 & 0 & W_N^{-2n} & 0 \\ 0 & 0 & 0 & 0 & 0 & W_N^{-n} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_6^1 & W_6^2 & W_6^3 & W_6^4 & W_6^5 \\ 1 & W_6^2 & W_6^4 & 1 & W_6^2 & W_6^4 \\ 1 & W_6^3 & 1 & W_6^3 & 1 & W_6^3 \\ 1 & W_6^4 & W_6^2 & 1 & W_6^4 & W_6^2 \\ 1 & W_6^5 & W_6^4 & W_6^3 & W_6^2 & W_6^1 \end{pmatrix} \begin{bmatrix} x(n) \\ x(n+N/6) \\ x(n+2N/6) \\ x(n+3N/6) \\ x(n+4N/6) \\ x(n+5N/6) \end{bmatrix}$$

Comme pour les bases impaires, aucune complexité de calcul supplémentaire ne sera engendrée dans la génération des nouveaux indices puisque :

$$(N+Bk-p) \bmod N = \begin{cases} N-p, & \text{pour } k=0 \\ Bk-p, & \text{ailleurs} \end{cases} \quad 1 \le k \le \frac{B}{2} - 1 \tag{I.34}$$

Le tableau I.2 présente le nombre de calculs des composantes réelles des facteurs *W* ou de leurs lectures à partir de la 'look-up table' requis par l'ATC et l'algorithme proposé et cela pour les bases fréquemment utilisées: *B* = 4, 6, 8 et 16. Il ressort de ce tableau que l'avantage d'utiliser l'algorithme proposé sur le CTA est exceptionnel. Dans les processeurs dédiés FFT, la taille de la mémoire de la 'look-up table' est considérablement réduite. Elle est restreinte de presque la moitié pour *B* = 16.

	ACT en base *B*				Algorithme proposé en base *B*			
N \ *B*	4	6	8	16	4	6	8	16
16	18				12			
36		50				30		
64	162		98		108		56	
216		650				390		
256	1026			450	684			240
512			1666				952	
1024	5634				3756			
1296		6050				3630		
4096	28674		20482	14850	19116		11704	7920
7776		49250				29550		
16384	139266				92844			
32768			221186				126392	
46656		373250				223950		
65536	655362			360450	436908			192240

Tableau I.2: Comparaison en nombres de calculs des composantes réelles des facteurs *W* ou de leurs lectures à partir de la 'look-up table' entre l'ACT et l'algorithme proposé (cas des bases paires).

3.4. L'algorithme conventionnel de Cooley–Tukey avec entrelacement temporel (E.T.) pour le calcul de la TFR en bases 2 et 4

Bien que la TFR en base arbitraire *B* développée ci-haut soit applicable à n'importe quelle base, par exemple les bases 2, 3, 4, 5, 7, 8, 16 etc., les bases les plus communément utilisées lors des calculs des FFTs sont les bases 2 et 4 parce qu'elles mènent à des algorithmes peu complexes et efficaces. Les bases 3, 5, 7, 8, 16 restent réservées à des cas spécifiques car leur complexité de calcul est largement supérieure. Dans ce qui suit, nous allons développer l'algorithme de Cooley/Tukey avec entrelacement temporel (E.T.) en bases 2 et 4 parce ces dernières permettent non seulement une complexité des calculs réduite mais aussi des implémentations de la TFR sur des architectures parallèles simples et performantes.

3.4.1. L'algorithme de Cooley–Tukey avec entrelacement temporel en base 2

Comme pour l'algorithme conventionnel de Cooley–Tukey avec entrelacement fréquentiel développé dans la section 3.2, l'idée consiste ici aussi à réécrire différemment les indices des composantes *x(n)* et *X(k)* dans l'équation de la TFD donnée par (I.11), mais d'une façon différente [62]. Le facteur multiplicatif $1/N$ est aussi omis ici parce qu'il n'intervient pas dans le développement de la TFR.

En démarrant du fait que N est une puissance de 2 ($N=2^d$, i.e. $d=log_2(N)$), les deux variables k et n, dont les représentations binaires sont respectivement $(k_{d-1}k_{d-2}\cdots k_1k_0)_2$ et $(n_{d-1}n_{d-2}\cdots n_1n_0)_2$, k_i et n_i : 0 ou 1, peuvent être écrites sous forme des sommes suivantes :

$$k = k_{d-1}.2^{d-1} + k_{d-2}.2^{d-2} + \cdots + k_1.2^1 + k_0.2^0$$
$$n = n_{d-1}.2^{d-1} + n_{d-2}.2^{d-2} + \cdots + n_1.2^1 + n_0.2^0$$

Le produit kn peut donc être écrit comme suit :

$$kn = (k_{d-1}.2^{d-1} + k_{d-2}.2^{d-2} + \cdots + k_1.2^1 + k_0.2^0)(n_{d-1}.2^{d-1} + n_{d-2}.2^{d-2} + \cdots + n_1.2^1 + n_0.2^0)$$

et l'équation (I.11) de la TFD s'écrit sous la forme :

$$X(k_{d-1},\cdots k_1,k_0) = \sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{d-1}=0}^{1} x(n_{d-1},\cdots,n_1,n_0).W_N^{(k_{d-1}.2^{d-1}+\cdots+k_1.2+k_0)(n_{d-1}.2^{d-1}+\cdots+n_1.2+n_0)} \qquad \text{(I.35)}$$

La factorisation des termes suivant les n_i permet d'aboutir à l'entrelacement temporel (E.T.). Elle débute par mettre en facteur le sous indice n_{d-1} qui permet l'écriture du produit kn sous la forme :

$$kn = (k_{d-1}.2^{d-1} + k_{d-2}.2^{d-2} + \cdots + k_1.2 + k_0)\left[n_{d-1}.2^{d-1} + (n_{d-2}.2^{d-2} + \cdots + n_1.2 + n_0)\right]$$

et qui devient, après réorganisation suivant le sous indice n_{d-1} :

$$kn = (k_{d-1}.2^{d-2} + k_{d-2}.2^{d-3} + \cdots + k_1).n_{d-1}.2^d + k_0.n_{d-1}.2^{d-1} + k(n_{d-2}.2^{d-2} + \cdots + n_1.2 + n_0)$$

En remplaçant kn par son expression ci-dessus, tout en sachant que $W_N^{\alpha N} = 1$ pour tout nombre relatif α, l'équation de la TFD, donnée par (I. 35), s'écrit alors sous la forme :

$$X(k_{d-1},\cdots,k_1,k_0) = \sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{d-2}=0}^{1} W_N^{k.(n_{d-2}.2^{d-2}+\cdots+n_1.2+n_0)} \sum_{n_{d-1}=0}^{1} x(n_{d-1},\cdots,n_1,n_0).W_N^{k_0.n_{d-1}.2^{d-1}} \qquad \text{(I.36)}$$

En posant : $X_0(n_{d-1},n_{d-2},\cdots,n_1,n_0) \equiv x(n_{d-1},n_{d-2},\cdots,n_1,n_0)$ et $T_0 = k_0.2^{d-1}$

on définit un vecteur intermédiaire, indicé par $(k_0,n_{d-2},\cdots,n_1,n_0)$, de longueur *N*, qui s'écrit :

$$X_1(k_0,n_{d-2},\cdots,n_1,n_0) = \sum_{n_{d-1}=0}^{1} X_0(n_{d-1},\cdots,n_1,n_0).W_N^{n_{d-1}.T_0}$$

L'équation (I.36) s'écrit alors :

$$X(k_{d-1},\cdots,k_1,k_0) = \sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{d-2}=0}^{1} X_1(k_0,n_{d-2}\cdots,n_1,n_0).W_N^{k.(n_{d-2}.2^{d-2}+\cdots+n_1.2+n_0)} \qquad \text{(I. 37)}$$

La même procédure de factorisation du produit noté cette fois-ci kn', où $n' = n_{d-2}.2^{d-2} + \cdots + n_1.2 + n_0$, est refaite telle que :

$$\begin{aligned} kn' &= k(n_{d-2}.2^{d-2} + \cdots + n_1.2 + n_0) \\ &= (k_{d-1}.2^{d-1} + k_{d-2}.2^{d-2} + \cdots + k_1.2 + k_0)(n_{d-2}.2^{d-2} + n_{d-3}.2^{d-3} + \cdots + n_1.2 + n_0) \\ &= (k_{d-1}.2^{d-1} + k_{d-2}.2^{d-2} + \cdots + k_1.2 + k_0)\left[n_{d-2}.2^{d-2} + (n_{d-3}.2^{d-3} + \cdots + n_1.2 + n_0)\right] \end{aligned}$$

De même que ce qui a été fait précédemment, le développement des calculs par réorganisation suivant le sous indice n_{d-2}, donne pour le produit kn' la relation :

$$kn' = (k_{d-1}.2^{d-3} + k_{d-2}.2^{d-4} + \cdots + k_2).n_{d-2}.2^d + (2k_1 + k_0).n_{d-2}.2^{d-2} + k(n_{d-3}.2^{d-3} + \cdots + n_1.2 + n_0)$$

En remplaçant kn' par son expression ci-dessus (avec $W_N^{\alpha N} = 1$), l'équation (I. 37) devient :

$$X(k_{d-1},\cdots,k_1,k_0) = \sum_{n_0=0}^{1}\cdots\sum_{n_{d-3}=0}^{1} W_N^{k(n_{d-3}.2^{d-3}+\cdots+n_1.2+n_0)} \sum_{n_{d-2}=0}^{1} X_1(k_0,n_{d-2},\cdots,n_1,n_0).W_N^{(2.k_1+k_0).n_{d-2}.2^{d-2}} \qquad \text{(I. 38)}$$

En posant $T_1 = (2k_1 + k_0).2^{d-2}$, et en faisant comme en haut, on peut définir un deuxième vecteur intermédiaire, indicé par $(k_0,k_1,n_{d-3}\cdots,n_1,n_0)$, de longueur *N*, et défini par :

$$X_2(k_0,k_1,n_{d-3},\cdots,n_1,n_0) = \sum_{n_{d-2}=0}^{1} X_1(k_0,n_{d-2},\cdots,n_1,n_0).W_N^{n_{d-2}.T_1}$$

L'équation (I. 38) s'écrit alors:

$$X(k_{d-1},\cdots,k_1,k_0) = \sum_{n_0=0}^{1}\sum_{n_1=0}^{1}\cdots\sum_{n_{d-3}=0}^{1} X_2(k_0,k_1,n_{d-3},\cdots,n_1,n_0).W_N^{k(n_{d-3}.2^{d-3}+\cdots+n_1.2+n_0)} \qquad \text{(I. 39)}$$

La même procédure est rééditée par sommation maintenant sur le sous indice n_{d-3}. Elle permet de définir un 3ème vecteur intermédiaire X_3, de longueur *N*, indicé par $(k_0,k_1,k_2,n_{d-4},\cdots,n_1,n_0)$:

$$X_3(k_0,k_1,k_2,n_{d-4},\cdots,n_1,n_0) = \sum_{n_{d-3}=0}^{1} X_2(k_0,k_1,n_{d-3},\cdots,n_1,n_0).W_N^{n_{d-3}.T_2}$$

tel que : $T_2 = (2^2.k_2 + 2k_1 + k_0).2^{d-3}$

Et ainsi de suite, on refait cette séquence d'opérations jusqu'à obtenir le *i*-ième vecteur intermédiaire X_i , ($i < d$), de longueur N et donné par :

$$X_i(k_0,k_1,\ldots,k_{i-2},k_{i-1},n_{d-(i+1)},\cdots,n_1,n_0) = \sum_{n_{d-i}=0}^{1} X_{i-1}(k_0,k_1,\cdots,k_{i-2},n_{d-i},n_{d-(i+1)},\cdots,n_1,n_0).W_N^{n_{d-i}T_{i-1}} \quad \text{(I. 40)}$$

tel que : $$T_{i-1} = (2^{i-1}.k_{i-1} + \cdots + 2^2.k_2 + 2k_1 + k_0).2^{d-i} = \left[\sum_{q=0}^{i-1} 2^q.k_q\right].2^{d-i} \quad \text{(I. 41)}$$

La dernière itération permet d'obtenir le *d*-ième et dernier vecteur intermédiaire X_d qui est donné par la relation suivante:

$$X_d(k_0,k_1,\ldots,k_{d-2},k_{d-1}) = \sum_{n_0=0}^{2} X_{d-1}(k_0,k_1,\ldots,k_{d-2},n_0).W_N^{n_0T_{d-1}} \quad \text{(I. 42)}$$

avec : $$T_{d-1} = 2^{d-1}.k_{d-1} + \cdots + 2^2.k_2 + 2k_1 + k_0 = \sum_{q=0}^{d-1} 2^q.k_q$$

Ce *d*-ième vecteur X_d n'est autre que le vecteur final X, avec une réorganisation des sous-indices différente, c.-à-d. $(k_0,k_1,\ldots,k_{d-2},k_{d-1})$. Cette réorganisation correspond à un ordre différent de stockage des différents éléments du vecteur X_d par rapport à celui du vecteur final X. Elle satisfait à la transposition suivante qui est en fait un miroir binaire :

$$\begin{array}{ccc} & \textit{Miroir} & \\ (k_0,k_1,\ldots,k_{d-2},k_{d-1}) & \updownarrow & (k_{d-1},k_{d-2},\cdots k_1,k_0) \end{array}$$

On remarque que dans (I.40) les indices de deux vecteurs intermédiaires successifs ne diffèrent que d'un seul sous indice. Par exemple pour l'étape *i*, c'est le sous indice de rang *d-i*. Pour plus de simplifications [62], on introduit un nouvel indice *m* et l'équation (I.40) devient :

$$X_i(m_i) = \sum_{n_{d-i}=0}^{1} X_{i-1}(m_{i-1}).W_N^{n_{d-i}T_{i-1}(m_i)} \quad \text{(I. 43)}$$

avec $(m_{i-1}) = (k_0,k_1,\cdots,k_{i-2},n_{d-i},n_{d-(i+1)},\cdots,n_1,n_0)$

et $(m_i) = (k_0,k_1,\ldots,k_{i-2},k_{i-1},n_{d-(i+1)},\cdots,n_1,n_0)$ (I. 44)

et où $m_{i-1} - m_i = (n_{d-i} - k_{i-1}).2^{d-i}$

donc $m_{i-1} = m_i + (n_{d-i} - k_{i-1}).2^{d-i}$ (I. 45)

Comme le montre la relation (I.41), T_{i-1} est unique pour chaque vecteur intermédiaire X_i, et est donc adressé avec le même indice que X_i (relation (I.43)).

En représentation binaire, m_i est de la forme :

$$(m_i) = ((m_i)_{d-1},(m_i)_{d-2},\cdots,(m_i)_{d-i},(m_i)_{d-(i+1)},\cdots,(m_i)_1,(m_i)_0)$$

En faisant l'analogie de cette représentation avec celle donnée par (I.44), cela donne pour $T_{i-1}(m_i)$ (équation (I.41)) et m_{i-1} (équation (I.45)) :

$$T_{i-1}(m_i) = \left[\sum_{q=0}^{i-1} 2^q (m_i)_{d-1-q} \right] .2^{d-i} \qquad \text{(I. 46)}$$

et $$m_{i-1} = m_i + (n_{d-i} - (m_i)_{d-i}).2^{d-i} = m_i - (m_i)_{d-i}.2^{d-i} + n_{d-i}.2^{d-i} \qquad \text{(I. 47)}$$

L'équation (I.43) peut donc être réécrite sous la forme :

$$X_i(m_i) = \sum_{n_{d-i}=0}^{1} X_{i-1}(m_i - (m_i)_{d-i}.2^{d-i} + n_{d-i}.2^{d-i}).W_N^{n_{d-i}T_{i-1}(m_i)} \qquad \text{(I. 48)}$$

Pour chaque état intermédiaire *i* (*i* allant de 1 à *d*), m_i varie de 0 à *N-1* (longueur *N* du vecteur intermédiaire) et n_{d-i} varie de 0 à *1*. En tenant compte de ces considérations, et en remplaçant m_i par *m* (*m* : 0 à *N-1*) et n_{d-i} par *n* (*n* : 0 à *1*), l'écriture de X_i est allégée, pour devenir :

$$X_i(m) = \sum_{n=0}^{1} X_{i-1}(m - m_{d-i}.2^{d-i} + n.2^{d-i}).W_N^{nT_{i-1}(m)} \qquad \text{(I. 49)}$$

avec $$T_{i-1}(m) = \left[\sum_{q=0}^{i-1} 2^q . m_{d-1-q} \right] .2^{d-i} \qquad \text{(I. 50)}$$

En regardant l'équation (I.49), on remarque que tous les éléments du vecteur intermédiaire X_i, dont les indices ne varient que par le seul sous indice m_{d-i}, dépendent des mêmes éléments du vecteur intermédiaire X_{i-1}, à des coefficients complexes $W_N^{nT_{i-1}(m)}$ près. En groupant ces éléments ensemble lors de leurs calculs permettra de réduire considérablement le nombre d'opérations complexes mises en jeu en faisant apparaître des structures semblables que l'on désigne par cellules de base. Pour regrouper ces éléments et les faire apparaître ensemble, on fait le changement de variable $r = m - m_{d-i}.2^{d-i}$, où *m* et *r* ont la même représentation binaire sauf pour le sous indice m_{d-i} qui est égal à zéro pour *r* comme on le voit ci-dessous :

$$\begin{array}{rcccccccccc} m & = & (m_{d-1} & m_{d-2} & \cdots & m_{d-(i-1)} & m_{d-i} & m_{d-(i+1)} & \cdots & m_1 & m_0)_2 \\ & & \| & \| & \cdots & \| & \neq & \| & \cdots & \| & \| \\ r & = & (r_{d-1} & r_{d-2} & \cdots & r_{d-(i-1)} & 0 & r_{d-(i+1)} & \cdots & r_1 & r_0)_2 \end{array} \qquad \text{(I. 51)}$$

En isolant le sous indice m_{d-i} dans l'expression (I.50), cela donne :

$$\begin{aligned} T_{i-1}(m) &= \left[\sum_{q=0}^{i-1} 2^q . m_{d-1-q} \right] .2^{d-i} = \left[2^{i-1} . m_{d-1-(i-1)} + \sum_{q=0}^{i-2} 2^q . m_{d-1-q} \right] .2^{d-i} \\ &= \left[\sum_{q=0}^{i-2} 2^q . m_{d-1-q} \right] .2^{d-i} + m_{d-i}.2^{d-1} = \left[\sum_{q=0}^{i-2} 2^q . r_{d-1-q} \right] .2^{d-i} + m_{d-i}.2^{d-1} \end{aligned} \qquad \text{(I. 52)}$$

Les sous indices m_{d-1-q} ont été remplacés par r_{d-1-q}, pour *q* allant de *0* à *i-2*, car ils sont les mêmes pour *m* et *r* (relation (I.51)).

En posant : $$C_i(r) = \left[\sum_{q=0}^{i-2} 2^q . r_{d-1-q} \right] .2^{d-i} \qquad \text{(I. 53)}$$

la relation (I.52) devient : $T_{i-1}(m) = C_i(r) + m_{d-i}.2^{d-1}$

Il faut remarquer que pour tout *i*, $C_i(0)=0$; et que pour $i = 1$, $C_i(r)=C_1(r)=0$.

Compte tenu de $W_N^{\alpha} = W_{R.N}^{R.\alpha}$, pour tout réel *R* différent de zéro, cela donne pour $W_N^{nT_{i-1}(m)}$:

$$W_N^{nT_{i-1}(m)} = W_N^{n(C_i(r)+m_{d-i}.2^{d-1})} = W_N^{n.C_i(r)}.W_N^{n.m_{d-i}.2^{d-1}} = W_N^{n.C_i(r)}.W_2^{n.m_{d-i}}$$

et l'équation (I.49) devient :

$$X_i(r + m_{d-i}.2^{d-i}) = \sum_{n=0}^{1} X_{i-1}(r + n.2^{d-i}).W_N^{n.C_i(r)}.W_2^{n.m_{d-i}} \qquad \text{(I. 54)}$$

Puisque on remarque que m_{d-i} varie de 0 à 1 indépendamment de *i*, l'écriture de X_i peut encore être simplifiée en remplaçant m_{d-i} par *m*, *m* variant de 0 à 1. L'équation (I.54) devient alors :

$$X_i(r + m.2^{d-i}) = \sum_{n=0}^{1} X_{i-1}(r + n.2^{d-i}).W_N^{n.C_i(r)}.W_2^{n.m} \qquad \text{(I. 55)}$$

Comme on peut le remarquer, l'équation (I.55) à la forme d'une TFD sur 2 points.

Pour chaque état intermédiaire *i* (*i* allant de 1 à *d*), *r* varie de 0 à *N*-1 avec $r_{d-i} = 0$ (la longueur du vecteur intermédiaire reste *N* puisque elle est donnée par $(r + m.2^{d-i})$). Et pour chaque valeur de *r*, *m* varie de 0 à 1. Enfin pour *i*, *r* et *m* donnés, *n* varie de 0 à 1.

L'écriture matricielle de l'équation (I.55) fait apparaître d'une manière plus explicite tous les couples de $X_i(r)$. Elle est donnée par :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{d-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & W_N^{C_i(r)} \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+2^{d-i}) \end{pmatrix} \qquad \text{(I.56)}$$

ou sous forme compacte par :

$$V_i = T_2.D_2.V_{(i-1)} \qquad \text{(I.57)}$$

où V_i, T_2, D_2 et $V_{(i-1)}$ représentent les vecteurs et matrices, dans l'ordre respectif, donnés par l'équation (I.56). Comme le montre la relation (I.56), pour une étape donnée *i*, V_i, D_2 et V_{i-1} dépendent du paramètre *r*. Connaissant la variation de *r* (*r* : 0, 1, …, *N-1* où $r_{d-i} = 0$), *(N/2)* vecteurs V_i sont donc définis pour chaque étape *i*.

Chaque élément de la matrice diagonale D_2 est donné par : $D_2(z,z) = W_N^{z.C_i(r)}$, *z* : 0… 1.

Les $C_i(r)$ se calculent selon la formule (I.53).

La transformée globale est donc décomposée récursivement en des transformées minimales sur 2 valeurs (pour chaque valeur de *r*, *m* varie de 0 à 1). Cela conduit à *(N/2)* calculs de base (variation de *r*), chacun d'entre eux étant effectué par une cellule élémentaire. Ces *(N/2)* calculs de base sont répétés sur $log_2(N)$ étapes (variation de *i*). Donc $\left(\frac{N}{2}\log_2(N)\right)$ cellules élémentaires sont nécessaires pour le calcul de la TFD en base 2.

Une écriture plus simplifiée et plus visuelle de l'algorithme nous donne :

Pour i : 1 … d

Pour r : 0 … N-1 (où $r_{d-i} = 0$)

Calcul de (I.56)

Pour obtenir le spectre final X dans l'ordre naturel, le dernier vecteur intermédiaire X_d doit être réorganisé. Cette remise en ordre des éléments du vecteur X_d est appelée cryptage/décryptage binaire. Elle consiste à faire un miroir binaire sur les indices. Le tableau I.3 donne à titre indicatif la table de transformation des différents indices pour $N=16$.

Il a donc fallu $d = log_2(N)$ itérations ou étapes pour le calcul des vecteurs intermédiaires, plus une procédure de réorganisation des éléments de l'ultime vecteur. La procédure de réorganisation qui est une étape prépondérante dans le calcul de la TFD a été investie dans plusieurs travaux de recherche [63 – 65].

Indice	Décomposition	Miroir	Indice après transformation	Indice	Décomposition	Miroir	Indice après transformation
0	0000	0000	0	8	1000	0001	1
1	0001	1000	8	9	1001	1001	9
2	0010	0100	4	10	1010	0101	5
3	0011	1100	12	11	1011	1101	13
4	0100	0010	2	12	1100	0011	3
5	0101	1010	10	13	1101	1011	11
6	0110	0110	6	14	1110	0111	7
7	0111	1110	14	15	1111	1111	15

Tableau I.3: Table de transformation des indices en base 2 (miroir binaire) correspondant à N =16.

La TFR en base 2 sera indiquée par le sigle FFT2 (Fast Fourier Transfom en base 2).

En termes de nombre d'opérations, la FFT2 qui d'après la relation (I.56) peut être écrite :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{d-i}) \end{pmatrix} = \begin{pmatrix} X_{i-1}(r) + X_{i-1}(r+2^{d-i}).W_N^{C_i(r)} \\ X_{i-1}(r) - X_{i-1}(r+2^{d-i}).W_N^{C_i(r)} \end{pmatrix} \quad \text{(I.58)}$$

montre que chaque cellule élémentaire réalise une multiplication complexe et deux additions complexes (la soustraction de la 2ème ligne est assimilée à une addition d'un nombre négatif), sauf à l'étape 1 où seulement deux additions complexes sont accomplies.

Puisque le nombre de cellules de base requis par la FFT2 est de $\left(\frac{N}{2}\log_2(N)\right)$, le nombre total d'opérations est donc de :

$$MC_{FTT2} = \frac{N}{2}\left(\log_2(N)-1\right) = \frac{N}{2}\log_2(\frac{N}{2}) \quad \text{Multiplications Complexes} \quad \text{(I.59)}$$

le (– 1) est relatif à la 1ère étape,

et $AC_{FTT2} = N.\log_2(N)$ Additions Complexes (I.60)

La FFT2 a donc un ordre de complexité O($N.log_2(N)$). En comparant cette complexité avec celle du calcul direct de la TFD qui est de O(N^2) cela donne un rapport de :

$$\frac{N^2}{N\log_2(N)} = \frac{N}{\log_2(N)} = \frac{2^d}{\log_2(2^d)} = \frac{2^d}{d}.$$

Ce rapport montre un gain *G* en temps d'exécution de la FFT2 par rapport à la TFD qui croit rapidement avec l'augmentation du nombre d'échantillons *N*. En raisonnant uniquement en nombre d'opérations, nous avons pour N =1024 (d=10) G=100, pour N =2048 (d=11) G=186, et pour N =4096 (d=12) G=341.

Chaque cellule de base représentée par (I.58) met en œuvre 2 évaluations, une pour $X_i(r)$ et l'autre pour $X_i(r+2^{d-i})$. Ces évaluations sont fonction des mêmes opérandes. Le diagramme de la figure I.1, qui par sa forme a été baptisé papillon, schématise les calculs mis en œuvre. Le nœud plus (+) représente une addition de l'entrée horizontale à l'entrée transversale, et le nœud moins (-) indique que l'entrée horizontale est soustraite de l'entrée transversale.

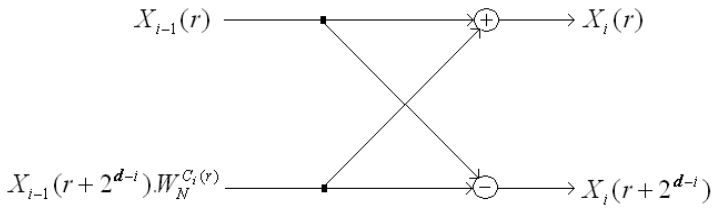


Figure I.1. Papillon de la FTT2 (E.T.).

L'évolution des calculs pour les différents vecteurs intermédiaires est bien visualisée sur les graphes de fluence qui sont dessinés à partir de tous les papillons intervenant dans une FFT. Les graphes de fluence seront d'ailleurs utilisés dans le chapitre II pour proposer les architectures parallèles implantables sur silicium. A titre d'exemples, les figures I.2, I.3 et I.4 donnent les graphes de fluence de la FFT2 pour $N=4=2^2$, $N=8=2^3$, et $N=16=2^4$ respectivement. Ils sont déduits directement des étapes de calcul de l'algorithme FFT2. Ces étapes sont clairement exposées ci-dessous. Il faut aussi remarquer qu'une réorganisation des éléments du dernier vecteur X_d, en faisant un miroir binaire sur ses indices (tableau I.3), permet de trouver le spectre final X dans l'ordre naturel.

Cas pour $N = 4 = 2^d = 2^2$; i : 1..2.

Les vecteurs intermédiaires et final sont calculés suivant la relation (I.58) réécrite ici pour N=4 :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{2-i}) \end{pmatrix} = \begin{pmatrix} X_{i-1}(r) + X_{i-1}(r+2^{2-i}).W_4^{C_i(r)} \\ X_{i-1}(r) - X_{i-1}(r+2^{2-i}).W_4^{C_i(r)} \end{pmatrix}$$

Les $C_i(r)$ sont calculés suivant (I. 53) pour d =2 : $C_i(r) = \left[\sum_{q=0}^{i-2} 2^q . r_{2-1-q}\right].2^{2-i} = \left[\sum_{q=0}^{i-2} 2^q . r_{1-q}\right].2^{2-i}$

Pour chaque étape i, $r = (r_1,r_0)_2$ varie de *0* à *3*, avec $r_{d-i} = r_{2-i} = 0$.

- Pour $i = 1$; $r = (r_1,r_0)_2$: 0, 1 (car $r_{2-i} = r_1 = 0$) et $C_1(r) = 0$, donc $W_4^{C_1(r)} = 1$;

$r = 0 = (00)_2$; $r = 1 = (01)_2$;

$$\begin{pmatrix} X_1(0) \\ X_1(2) \end{pmatrix} = \begin{pmatrix} X_0(0) + X_0(2) \\ X_0(0) - X_0(2) \end{pmatrix} \qquad \begin{pmatrix} X_1(1) \\ X_1(3) \end{pmatrix} = \begin{pmatrix} X_0(1) + X_0(3) \\ X_0(1) - X_0(3) \end{pmatrix}$$

- Pour $i = 2$; $r = (r_1,r_0)_2$: 0, 2 (car $r_{2-i} = r_0 = 0$); $C_2(r) = \left[\sum_{q=0}^{0} 2^q . r_{1-q}\right].2^{2-2} = r_1$;

$r = 0 = (00)_2$; $C_2(0) = 0$; $r = 2 = (10)_2$; $C_2(2) = 1$;

$$\begin{pmatrix} X_2(0) \\ X_2(1) \end{pmatrix} = \begin{pmatrix} X_1(0) + X_1(1).W_4^0 \\ X_1(0) - X_1(1).W_4^0 \end{pmatrix} \qquad \begin{pmatrix} X_2(2) \\ X_2(3) \end{pmatrix} = \begin{pmatrix} X_1(2) + X_1(3).W_4^1 \\ X_1(2) - X_1(3).W_4^1 \end{pmatrix}$$

On déduit le diagramme de fluence de la FFT2 pour N=4 qui est donné ci-dessous.

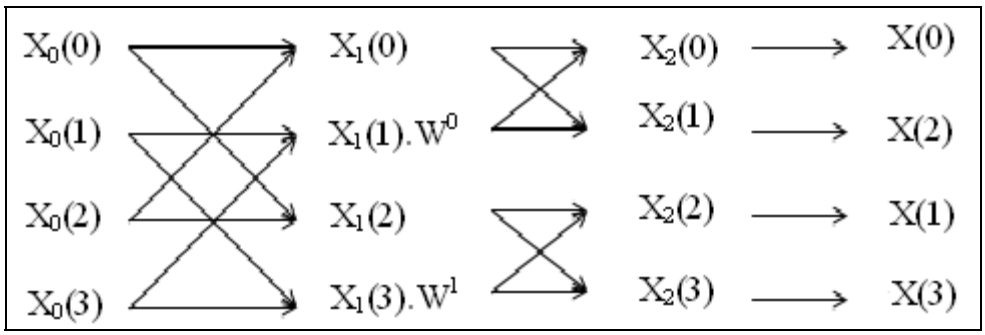


Figure I.2. Diagramme de fluence de la FFT2 (par E.T.) pour $N = 4 = 2^2$.

Cas pour $N = 8 = 2^d = 2^3$; i : 1..3.

Les vecteurs intermédiaires et final sont calculés suivant la relation (I.58) réécrite ici pour N=8 :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{3-i}) \end{pmatrix} = \begin{pmatrix} X_{i-1}(r) + X_{i-1}(r+2^{3-i}).W_8^{C_i(r)} \\ X_{i-1}(r) - X_{i-1}(r+2^{3-i}).W_8^{C_i(r)} \end{pmatrix}$$

Les $C_i(r)$ sont calculés suivant (I. 53) pour d =3 : $C_i(r) = \left[\sum_{q=0}^{i-2} 2^q . r_{3-1-q}\right].2^{3-i} = \left[\sum_{q=0}^{i-2} 2^q . r_{2-q}\right].2^{3-i}$.

Pour chaque étape i, $r = (r_2,r_1,r_0)_2$ varie de *0* à *7*, avec $r_{3-i} = 0$.

- Pour $i = 1$; $r = (r_2,r_1,r_0)_2$: 0..3 (car $r_{3-i} = r_2 = 0$) et $C_1(r) = 0$, donc $W_8^{C_1(r)} = 1$;

$r = 0 = (000)_2$; $r = 2 = (010)_2$;

$$\begin{pmatrix} X_1(0) \\ X_1(4) \end{pmatrix} = \begin{pmatrix} X_0(0) + X_0(4) \\ X_0(0) - X_0(4) \end{pmatrix} \qquad \begin{pmatrix} X_1(2) \\ X_1(6) \end{pmatrix} = \begin{pmatrix} X_0(2) + X_0(6) \\ X_0(2) - X_0(6) \end{pmatrix}$$

$r = 1 = (001)_2$; $r = 3 = (011)_2$;

$$\begin{pmatrix} X_1(1) \\ X_1(5) \end{pmatrix} = \begin{pmatrix} X_0(1) + X_0(5) \\ X_0(1) - X_0(5) \end{pmatrix} \qquad \begin{pmatrix} X_1(3) \\ X_1(7) \end{pmatrix} = \begin{pmatrix} X_0(3) + X_0(7) \\ X_0(3) - X_0(7) \end{pmatrix}$$

- Pour $i = 2$; $r = (r_2,r_1,r_0)_2$: 0, 1, 4, 5 (car $r_{3-i} = r_1 = 0$); $C_2(r) = \left[\sum_{q=0}^{0} 2^q . r_{3-1-q}\right] . 2^{3-2} = 2r_2$;

$r = 0 = (000)_2$; $C_2(0) = 0$;

$$\begin{pmatrix} X_2(0) \\ X_2(2) \end{pmatrix} = \begin{pmatrix} X_1(0) + X_1(2).W_8^0 \\ X_1(0) - X_1(2).W_8^0 \end{pmatrix}$$

$r = 1 = (001)_2$; $C_2(1) = 0$;

$$\begin{pmatrix} X_2(1) \\ X_2(3) \end{pmatrix} = \begin{pmatrix} X_1(1) + X_1(3).W_8^0 \\ X_1(1) - X_1(3).W_8^0 \end{pmatrix}$$

$r = 4 = (100)_2$; $C_2(4) = 2$;

$$\begin{pmatrix} X_2(4) \\ X_2(6) \end{pmatrix} = \begin{pmatrix} X_1(4) + X_1(6).W_8^2 \\ X_1(4) - X_1(6).W_8^2 \end{pmatrix}$$

$r = 5 = (101)_2$; $C_2(5) = 2$;

$$\begin{pmatrix} X_2(5) \\ X_2(7) \end{pmatrix} = \begin{pmatrix} X_1(5) + X_1(7).W_8^2 \\ X_1(5) - X_1(7).W_8^2 \end{pmatrix}$$

- Pour $i = 3$; $r = (r_2,r_1,r_0)_2$: 0, 2, 4, 6 (car $r_{3-i} = r_0 = 0$); $C_3(r) = \left[\sum_{q=0}^{1} 2^q . r_{3-1-q}\right] = r_2 + 2r_1$;

$r = 0 = (000)_2$; $C_3(0) = 0$;

$$\begin{pmatrix} X_3(0) \\ X_3(1) \end{pmatrix} = \begin{pmatrix} X_2(0) + X_2(1).W_8^0 \\ X_2(0) - X_2(1).W_8^0 \end{pmatrix}$$

$r = 2 = (010)_2$; $C_3(2) = 2$;

$$\begin{pmatrix} X_3(2) \\ X_3(3) \end{pmatrix} = \begin{pmatrix} X_2(2) + X_2(3).W_8^2 \\ X_2(2) - X_2(3).W_8^2 \end{pmatrix}$$

$r = 4 = (100)_2$; $C_3(4) = 1$;

$$\begin{pmatrix} X_3(4) \\ X_3(5) \end{pmatrix} = \begin{pmatrix} X_2(4) + X_2(5).W_8^1 \\ X_2(4) - X_2(5).W_8^1 \end{pmatrix}$$

$r = 6 = (110)_2$; $C_3(6) = 3$;

$$\begin{pmatrix} X_3(6) \\ X_3(7) \end{pmatrix} = \begin{pmatrix} X_2(6) + X_2(7).W_8^3 \\ X_2(6) - X_2(7).W_8^3 \end{pmatrix}$$

D'où le diagramme de fluence de la FFT2 pour N=8 :

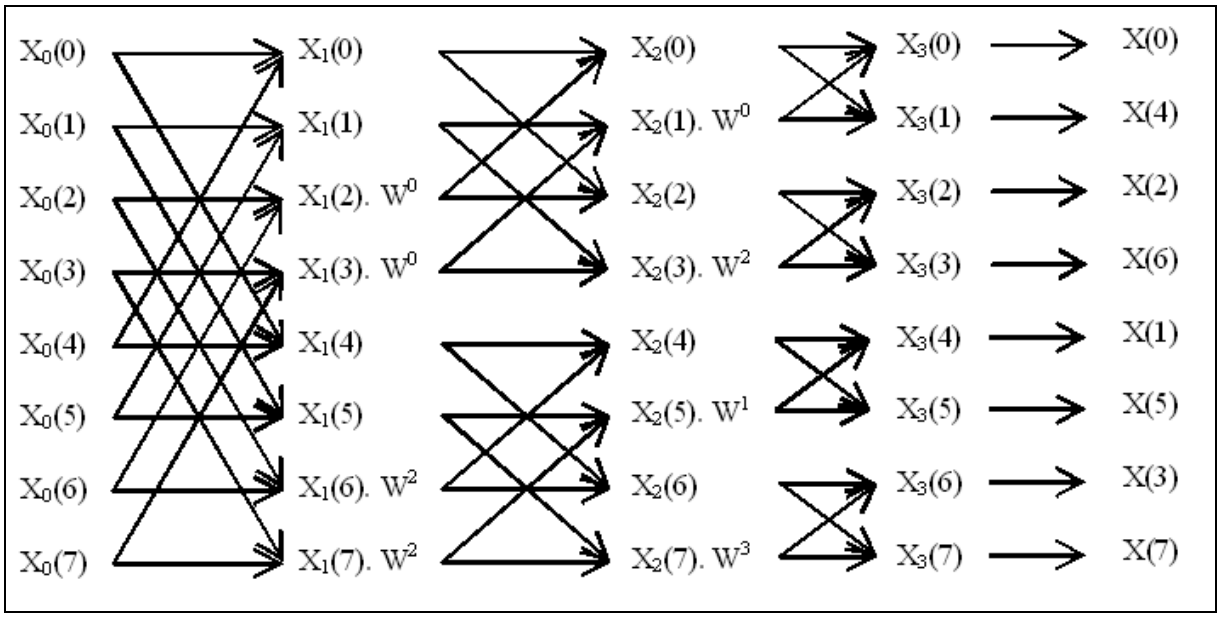


Figure I.3. Diagramme de fluence de la FFT2 (par E.T.) pour $N = 8 = 2^3$.

Cas pour $N = 16 = 2^d = 2^4$; i : 1..4.

Les vecteurs intermédiaires et final sont calculés suivant la relation (I.58) réécrite ici pour N=16 :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{4-i}) \end{pmatrix} = \begin{pmatrix} X_{i-1}(r) + X_{i-1}(r+2^{4-i}).W_{16}^{C_i(r)} \\ X_{i-1}(r) - X_{i-1}(r+2^{4-i}).W_{16}^{C_i(r)} \end{pmatrix}$$

Les $C_i(r)$ sont calculés suivant (I. 53) pour d =4 : $C_i(r) = \left[\sum_{q=0}^{i-2} 2^q.r_{4-1-q}\right].2^{4-i} = \left[\sum_{q=0}^{i-2} 2^q.r_{3-q}\right].2^{4-i}$.

Pour chaque étape i, $r = (r_3,r_2,r_1,r_0)_2$ varie de *0* à *15*, avec $r_{4-i} = 0$.

- Pour $i = 1$; $r = (r_3,r_2,r_1,r_0)_2$: 0..7 (car $r_{4-i} = r_3 = 0$) et $C_1(r)$=0, donc $W_{16}^{C_1(r)} = 1$;

$r = 0 = (0000)_2$;
$$\begin{pmatrix} X_1(0) \\ X_1(8) \end{pmatrix} = \begin{pmatrix} X_0(0) + X_0(8) \\ X_0(0) - X_0(8) \end{pmatrix}$$

$r = 1 = (0001)_2$;
$$\begin{pmatrix} X_1(1) \\ X_1(9) \end{pmatrix} = \begin{pmatrix} X_0(1) + X_0(9) \\ X_0(1) - X_0(9) \end{pmatrix}$$

$r = 2 = (0010)_2$;
$$\begin{pmatrix} X_1(2) \\ X_1(10) \end{pmatrix} = \begin{pmatrix} X_0(2) + X_0(10) \\ X_0(2) - X_0(10) \end{pmatrix}$$

$r = 3 = (0011)_2$;
$$\begin{pmatrix} X_1(3) \\ X_1(11) \end{pmatrix} = \begin{pmatrix} X_0(3) + X_0(11) \\ X_0(3) - X_0(11) \end{pmatrix}$$

$r = 4 = (0100)_2$;
$$\begin{pmatrix} X_1(4) \\ X_1(12) \end{pmatrix} = \begin{pmatrix} X_0(4) + X_0(12) \\ X_0(4) - X_0(12) \end{pmatrix}$$

$r = 5 = (0101)_2$;
$$\begin{pmatrix} X_1(5) \\ X_1(13) \end{pmatrix} = \begin{pmatrix} X_0(5) + X_0(13) \\ X_0(5) - X_0(13) \end{pmatrix}$$

$r = 6 = (0110)_2$;
$$\begin{pmatrix} X_1(6) \\ X_1(14) \end{pmatrix} = \begin{pmatrix} X_0(6) + X_0(14) \\ X_0(6) - X_0(14) \end{pmatrix}$$

$r = 7 = (0111)_2$;
$$\begin{pmatrix} X_1(7) \\ X_1(15) \end{pmatrix} = \begin{pmatrix} X_0(7) + X_0(15) \\ X_0(7) - X_0(15) \end{pmatrix}$$

- Pour $i = 2$; $r = (r_3,r_2,r_1,r_0)_2$: 0…3 et 8…11 (car $r_{4-i} = r_2 = 0$); $C_2(r) = \left[\sum_{q=0}^{0} 2^q.r_{3-q}\right].2^{4-2} = 4r_3$

$r = 0 = (0000)_2$; $C_2(0) = 0$;
$$\begin{pmatrix} X_2(0) \\ X_2(4) \end{pmatrix} = \begin{pmatrix} X_1(0) + X_1(4).W_{16}^0 \\ X_1(0) - X_1(4).W_{16}^0 \end{pmatrix}$$

$r = 1 = (0001)_2$; $C_2(1) = 0$;
$$\begin{pmatrix} X_2(1) \\ X_2(5) \end{pmatrix} = \begin{pmatrix} X_1(1) + X_1(5).W_{16}^0 \\ X_1(1) - X_1(5).W_{16}^0 \end{pmatrix}$$

$r = 2 = (0010)_2$; $C_2(2) = 0$;
$$\begin{pmatrix} X_2(2) \\ X_2(6) \end{pmatrix} = \begin{pmatrix} X_1(2) + X_1(6).W_{16}^0 \\ X_1(2) - X_1(6).W_{16}^0 \end{pmatrix}$$

$r = 3 = (0011)_2$; $C_2(3) = 0$;
$$\begin{pmatrix} X_2(3) \\ X_2(7) \end{pmatrix} = \begin{pmatrix} X_1(3) + X_1(7).W_{16}^0 \\ X_1(3) - X_1(7).W_{16}^0 \end{pmatrix}$$

$r = 8 = (1000)_2$; $C_2(8) = 4$;
$$\begin{pmatrix} X_2(8) \\ X_2(12) \end{pmatrix} = \begin{pmatrix} X_1(8) + X_1(12).W_{16}^4 \\ X_1(8) - X_1(12).W_{16}^4 \end{pmatrix}$$

$r = 9 = (1001)_2$; $C_2(9) = 4$;
$$\begin{pmatrix} X_2(9) \\ X_2(13) \end{pmatrix} = \begin{pmatrix} X_1(9) + X_1(13).W_{16}^4 \\ X_1(9) - X_1(13).W_{16}^4 \end{pmatrix}$$

$r = 10 = (1010)_2$; $C_2(10) = 4$;
$$\begin{pmatrix} X_2(10) \\ X_2(14) \end{pmatrix} = \begin{pmatrix} X_1(10) + X_1(14).W_{16}^4 \\ X_1(10) - X_1(14).W_{16}^4 \end{pmatrix}$$

$r = 11 = (1011)_2$; $C_2(11) = 4$;
$$\begin{pmatrix} X_2(11) \\ X_2(15) \end{pmatrix} = \begin{pmatrix} X_1(11) + X_1(15).W_{16}^4 \\ X_1(11) - X_1(15).W_{16}^4 \end{pmatrix}$$

- Pour $i = 3$; $r = (r_3,r_2,r_1,r_0)_2$: 0, 1, 4, 5, 8, 9, 12, 13 (car $r_{4-i} = r_1 = 0$);

$$C_3(r) = \left[\sum_{q=0}^{1} 2^q . r_{3-q} \right] .2 = 2r_3 + 4r_2$$

$r = 0 = (0000)_2$; $C_3(0) = 0$;
$$\begin{pmatrix} X_3(0) \\ X_3(2) \end{pmatrix} = \begin{pmatrix} X_2(0) + X_2(2).W_{16}^0 \\ X_2(0) - X_2(2).W_{16}^0 \end{pmatrix}$$

$r = 1 = (0001)_2$; $C_3(1) = 0$;
$$\begin{pmatrix} X_3(1) \\ X_3(3) \end{pmatrix} = \begin{pmatrix} X_2(1) + X_2(3).W_{16}^0 \\ X_2(1) - X_2(3).W_{16}^0 \end{pmatrix}$$

$r = 4 = (0100)_2$; $C_3(4) = 4$;
$$\begin{pmatrix} X_3(4) \\ X_3(6) \end{pmatrix} = \begin{pmatrix} X_2(4) + X_2(6).W_{16}^4 \\ X_2(4) - X_2(6).W_{16}^4 \end{pmatrix}$$

$r = 5 = (0101)_2$; $C_3(5) = 4$;
$$\begin{pmatrix} X_3(5) \\ X_3(7) \end{pmatrix} = \begin{pmatrix} X_2(5) + X_2(7).W_{16}^4 \\ X_2(5) - X_2(7).W_{16}^4 \end{pmatrix}$$

$r = 8 = (1000)_2$; $C_3(8) = 2$;
$$\begin{pmatrix} X_3(8) \\ X_3(10) \end{pmatrix} = \begin{pmatrix} X_2(8) + X_2(10).W_{16}^2 \\ X_2(8) - X_2(10).W_{16}^2 \end{pmatrix}$$

$r = 9 = (1001)_2$; $C_3(9) = 2$;
$$\begin{pmatrix} X_3(9) \\ X_3(11) \end{pmatrix} = \begin{pmatrix} X_2(9) + X_2(11).W_{16}^2 \\ X_2(9) - X_2(11).W_{16}^2 \end{pmatrix}$$

$r = 12 = (1100)_2$; $C_3(12) = 6$;
$$\begin{pmatrix} X_3(12) \\ X_3(14) \end{pmatrix} = \begin{pmatrix} X_2(12) + X_2(14).W_{16}^6 \\ X_2(12) - X_2(14).W_{16}^6 \end{pmatrix}$$

$r = 13 = (1101)_2$; $C_3(13) = 6$;
$$\begin{pmatrix} X_3(13) \\ X_3(15) \end{pmatrix} = \begin{pmatrix} X_2(13) + X_2(15).W_{16}^6 \\ X_2(13) - X_2(15).W_{16}^6 \end{pmatrix}$$

- Pour $i = 4$; $r = (r_3,r_2,r_1,r_0)_2$: 0, 2, 4, 6, 8, 10, 12, 14 (car $r_{4-4} = r_0 = 0$) ;

$$C_4(r) = \left[\sum_{q=0}^{2} 2^q . r_{3-q} \right] = r_3 + 2r_2 + 4r_1 \ ;$$

$r = 0 = (0000)_2$; $C_4(0) = 0$;
$$\begin{pmatrix} X_4(0) \\ X_4(1) \end{pmatrix} = \begin{pmatrix} X_3(0) + X_3(1).W_{16}^0 \\ X_3(0) - X_3(1).W_{16}^0 \end{pmatrix}$$

$r = 2 = (0010)_2$; $C_4(2) = 4$;
$$\begin{pmatrix} X_4(2) \\ X_4(3) \end{pmatrix} = \begin{pmatrix} X_3(2) + X_3(3).W_{16}^4 \\ X_3(2) - X_3(3).W_{16}^4 \end{pmatrix}$$

$r = 4 = (0100)_2$; $C_4(4) = 2$;
$$\begin{pmatrix} X_4(4) \\ X_4(5) \end{pmatrix} = \begin{pmatrix} X_3(4) + X_3(5).W_{16}^2 \\ X_3(4) - X_3(5).W_{16}^2 \end{pmatrix}$$

$r = 6 = (0110)_2$; $C_4(6) = 6$;
$$\begin{pmatrix} X_4(6) \\ X_4(7) \end{pmatrix} = \begin{pmatrix} X_3(6) + X_3(7).W_{16}^6 \\ X_3(6) - X_3(7).W_{16}^6 \end{pmatrix}$$

$r = 8 = (1000)_2$; $C_4(8) = 1$;
$$\begin{pmatrix} X_4(8) \\ X_4(9) \end{pmatrix} = \begin{pmatrix} X_3(8) + X_3(9).W_{16}^1 \\ X_3(8) - X_3(9).W_{16}^1 \end{pmatrix}$$

$r = 10 = (1010)_2$; $C_4(10) = 5$;
$$\begin{pmatrix} X_4(10) \\ X_4(11) \end{pmatrix} = \begin{pmatrix} X_3(10) + X_3(11).W_{16}^5 \\ X_3(10) - X_3(11).W_{16}^5 \end{pmatrix}$$

$r = 12 = (1100)_2$; $C_4(12) = 3$;
$$\begin{pmatrix} X_4(12) \\ X_4(13) \end{pmatrix} = \begin{pmatrix} X_3(12) + X_3(13).W_{16}^3 \\ X_3(12) - X_3(13).W_{16}^3 \end{pmatrix}$$

$r = 14 = (1110)_2$; $C_4(14) = 7$;
$$\begin{pmatrix} X_4(14) \\ X_4(15) \end{pmatrix} = \begin{pmatrix} X_3(14) + X_3(15).W_{16}^7 \\ X_3(14) - X_3(15).W_{16}^7 \end{pmatrix}$$

Le diagramme de fluence de la FFT2 pour N=16 est donné sur la figure I.4

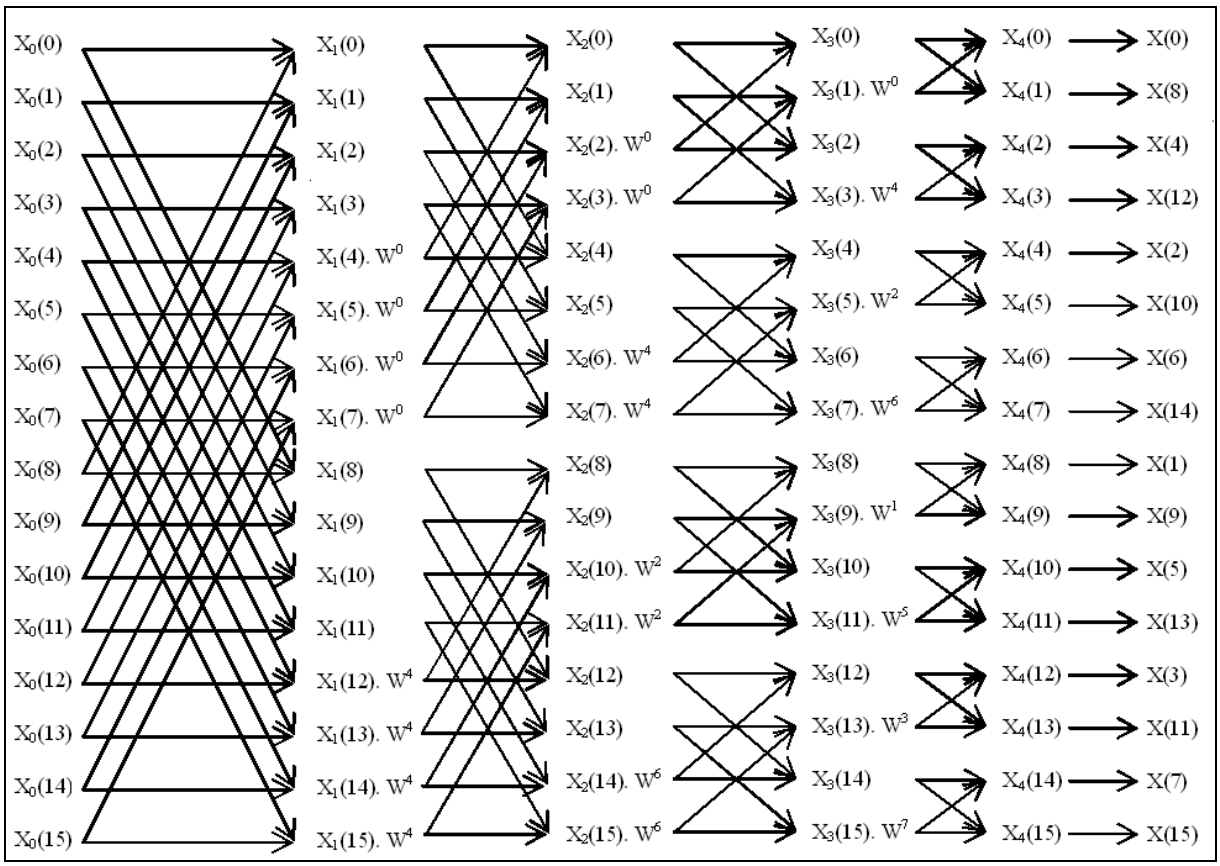


Figure I.4. Diagramme de fluence de la FFT2 (par E.T.) pour $N = 16 = 2^4$.

3.4.2. L'algorithme de Cooley–Tukey avec entrelacement temporel en base 4

La TFR en base 4 qu'on dénotera par FFT4 utilisera un développement analogue à celui de la FFT2. On démarre du fait que N est une puissance de 4 ($N=4^d$, donc $d = log_4(N)$). Les deux variables k et n, dont les représentations en base 4 sont respectivement $(k_{d-1}k_{d-2}\cdots k_1k_0)_4$ et $(n_{d-1}n_{d-2}\cdots n_1n_0)_4$, avec k_i et n_i: 0 … 3, peuvent être écrites sous forme des sommes suivantes:

$$k = k_{d-1}.4^{d-1} + k_{d-2}.4^{d-2} + \cdots + k_1.4^1 + k_0.4^0$$
$$n = n_{d-1}.4^{d-1} + n_{d-2}.4^{d-2} + \cdots + n_1.4^1 + n_0.4^0$$

Le produit kn peut alors être écrit sous la forme :

$$kn = (k_{d-1}.4^{d-1} + k_{d-2}.4^{d-2} + \cdots + k_1.4^1 + k_0.4^0)(n_{d-1}.4^{d-1} + n_{d-2}.4^{d-2} + \cdots + n_1.4^1 + n_0.4^0)$$

et l'équation (I.11) de la TFD s'écrit alors sous la forme :

$$X(k_{d-1},\cdots k_1,k_0) = \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\cdots\sum_{n_{d-1}=0}^{3} x(n_{d-1},\cdots,n_1,n_0).W_N^{(k_{d-1}.4^{d-1}+\cdots+k_1.4+k_0)(n_{d-1}.4^{d-1}+\cdots+n_1.4+n_0)} \qquad \text{(I.61)}$$

La factorisation des termes suivant les n_i permettant d'aboutir à l'entrelacement temporel est alors appliquée à l'équation (I.61) en débutant par mettre en facteur le sous indice n_{d-1}.

Toutes les étapes du développement effectuées pour aboutir à la forme de la cellule de base de la FFT2 donnée par (I.56) sont similairement appliquées ici à l'équation (I.61) pour aboutir à la cellule de base de la FFT4 donnée par :

$$\begin{pmatrix} X_i(r) \\ X_i(r+4^{d-i}) \\ X_i(r+2.4^{d-i}) \\ X_i(r+3.4^{d-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^4 & W_4^6 \\ 1 & W_4^3 & W_4^6 & W_4^9 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & W_N^{C_i(r)} & 0 & 0 \\ 0 & 0 & W_N^{2.C_i(r)} & 0 \\ 0 & 0 & 0 & W_N^{3.C_i(r)} \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+4^{d-i}) \\ X_{i-1}(r+2.4^{d-i}) \\ X_{i-1}(r+3.4^{d-i}) \end{pmatrix} \quad \text{(I. 62)}$$

ou sous forme compacte par :

$$V_i = T_4.D_4.V_{(i-1)} \quad \text{(I. 63)}$$

Chaque élément de la matrice diagonale D_4 est donné par : $D_4(z,z) = W_N^{z.C_i(r)}$, z : 0… 3.
Par analogie avec la formule (I.53), les $C_i(r)$ se calculent selon la formule (I.64).

$$C_i(r) = \left[\sum_{q=0}^{i-2} 4^q . r_{d-1-q} \right] . 4^{d-i} \quad \text{(I.64)}$$

En appliquant la propriété de la périodicité des facteurs de rotation W_4, T_4 devient :

$$T_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^4 & W_4^6 \\ 1 & W_4^3 & W_4^6 & W_4^9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix}$$

Ce qui montre que T_4 n'introduira aucune multiplication et que la cellule de base de la FFT4 a une complexité d'implémentation réduite.
La représentation matricielle de la cellule de base de la FFT4 est finalement donnée par :

$$\begin{pmatrix} X_i(r) \\ X_i(r+4^{d-i}) \\ X_i(r+2.4^{d-i}) \\ X_i(r+3.4^{d-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & W_N^{C_i(r)} & 0 & 0 \\ 0 & 0 & W_N^{2.C_i(r)} & 0 \\ 0 & 0 & 0 & W_N^{3.C_i(r)} \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+4^{d-i}) \\ X_{i-1}(r+2.4^{d-i}) \\ X_{i-1}(r+3.4^{d-i}) \end{pmatrix} \quad \text{(I.65)}$$

La transformée globale est donc décomposée récursivement en des transformées minimales sur 4 valeurs. *(N/4)* calculs de base (variation de *r*), chacun effectué par une cellule élémentaire, sont répétés sur $log_4(N)$ étapes (variation de *i*). Donc $\left(\frac{N}{4} \log_4(N) \right)$ cellules élémentaires sont nécessaires pour le calcul de la TFD en base 4.

L'écriture simplifiée de l'algorithme nous donne :

Pour i : 1 … d

Pour r : 0 … N-1 (où $r_{d-i} = 0$)

Calcul de (I.65)

De même que pour la FFT2 le dernier vecteur intermédiaire X_d de la FFT4 doit être réorganisé, en faisant un miroir sur les indices, écrits en base 4, pour donner le vecteur final X recherché.

Le tableau I.4 donne la table de transformation des différents indices pour $N=16$.

Il a donc fallu $d = log_4(N)$ itérations pour le calcul des vecteurs intermédiaires, plus une étape de réorganisation des éléments de l'ultime vecteur.

Indice		Décomposition	Miroir	Indice après transformation		Indice	Décomposition	Miroir	Indice après transformation
0		00	00	0		8	20	02	2
1		01	10	4		9	21	12	6
2		02	20	8		10	22	22	10
3		03	30	12		11	23	32	14
4		10	01	1		12	30	03	3
5		11	11	5		13	31	13	7
6		12	21	9		14	32	23	11
7		13	31	13		15	33	33	15

Tableau I.4: Table de transformation des indices en base 4 (miroir en base 4) correspondant à N =16.

En effectuant le produit matriciel du second membre de l'équation (I.65) nous obtenons :

$$\begin{pmatrix} X_i(r) \\ X_i(r+4^{d-i}) \\ X_i(r+2.4^{d-i}) \\ X_i(r+3.4^{d-i}) \end{pmatrix} = \begin{pmatrix} [X_{i-1}(r)+X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]+[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}+X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)-X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]-j[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}-X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)+X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]-[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}+X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)-X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]+j[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}-X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \end{pmatrix} \quad \text{(I. 66)}$$

Cela représente en termes d'opérations complexes : 3 multiplications ($X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}$, $X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}$ et $X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}$) suivies de 4 additions (termes entre crochets), suivies d'une multiplication par j (non comptée : le produit par j est traité autrement), et enfin 4 additions (une addition pour chaque ligne du vecteur). Cela fait en tout 3 multiplications complexes et 8 additions complexes. Puisque le nombre de cellules de base nécessaire pour la FFT4 est de $\left(\frac{N}{4}\log_4(N)\right)$, le nombre total d'opérations est donc de :

$$MC_{FTT4} = 3\frac{N}{4}\left(\log_4(N)-1\right) = \frac{3}{4}N.\log_4(\frac{N}{4}) = \frac{3}{8}N.\log_2(\frac{N}{4}) \text{ multiplications complexes} \quad \text{(I. 67)}$$

Le (-1) est dû à la 1ère étape qui ne fait pas intervenir de multiplications.

$$AC_{FTT4} = 8.\frac{N}{4}.\log_4(N) = 2.N.\log_4(N) = N.\log_2(N) \quad \text{additions complexes} \quad \text{(I. 68)}$$

La FFT4 a donc un ordre de complexité O($N.log_4(N)$).

Le diagramme de la figure I.5 schématise le papillon de la FTT4 (les mentions -1, j et –j dénotent une multiplication de l'élément correspondant par l'un de ces facteurs).

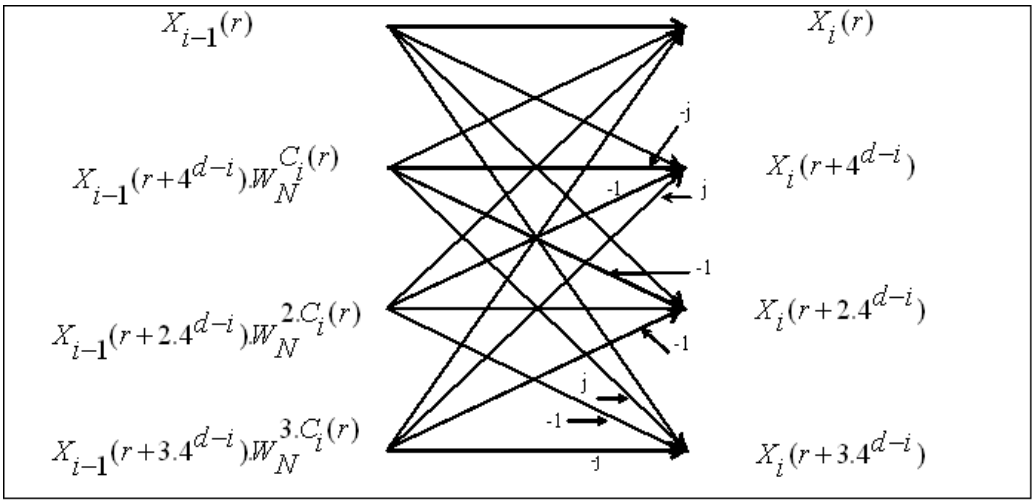


Figure I.5. Papillon de la FTT4 (par E.T.).

Comme exemples d'application, l'algorithme de la FFT4 est appliqué dans ce qui suit pour les cas où $N = 4^1$ et $N=16=4^2$ qui nous permettent de déduire respectivement les graphes de fluence représentés sur les figures I.6 et I.7. Comme on le voit sur la figure I.7 pour $N=16$, les éléments du dernier vecteur X_p sont réorganisés pour donner le spectre final X dans l'ordre naturel.

Cas pour $N = 4 = 4^d = 4^1$; $i = 1$; $r = (r_0)_4 = 0$ ($r_{d-i} = r_{1-i} = r_0 = 0$) ; $C_1(r) = 0$; $W_4^{C_1(r)} = 1$

La relation (I.65) donne :

$$\begin{pmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} X_0(0) \\ X_0(1) \\ X_0(2) \\ X_0(3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_0(0) \\ X_0(1) \\ X_0(2) \\ X_0(3) \end{pmatrix}$$

Ce qui permet d'aboutir au diagramme de fluence de la figure 1.6.

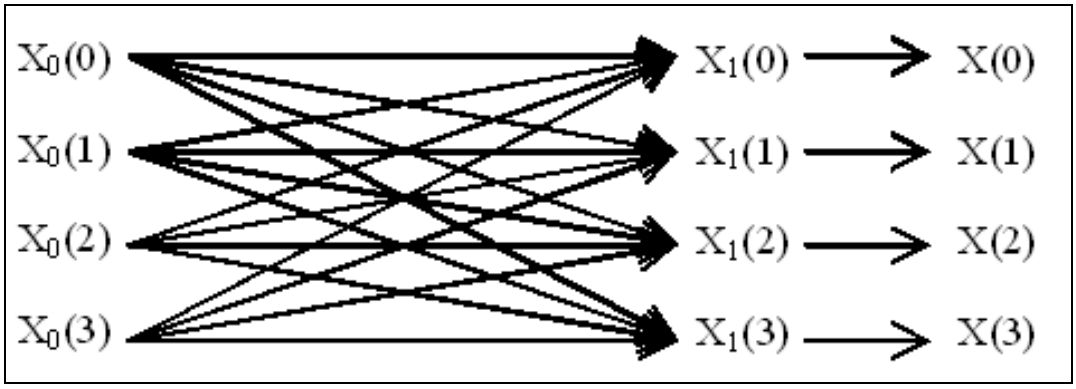


Figure I.6. Graphe de fluence de la FFT4 (par E.T.) pour $N = 4$.

Cas pour $N = 16 = 4^d = 4^2$; i : 1..2.

La relation (I.65) donne :

$$\begin{pmatrix} X_i(r) \\ X_i(r+4^{2-i}) \\ X_i(r+2.4^{2-i}) \\ X_i(r+3.4^{2-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+4^{2-i}).W_{16}^{C_i(r)} \\ X_{i-1}(r+2.4^{2-i}).W_{16}^{2.C_i(r)} \\ X_{i-1}(r+3.4^{2-i}).W_{16}^{3.C_i(r)} \end{pmatrix};$$

Les $C_i(r)$ sont calculés suivant (I. 64) pour d =2: $C_i(r) = \left[\sum_{q=0}^{i-2} 4^q . r_{1-q} \right] .4^{2-i}$.

Pour chaque étape i, $r = (r_1,r_0)_4$ varie de *0* à *15*, où $r_{2-i} = 0$.

- Pour $i = 1$; $r = (r_1,r_0)_4$: 0 à 3 ($r_{2-i} = r_1 = 0$) et $C_1(r) = 0$, donc $W_{16}^{\alpha.C_1(r)} = 1$, α : 1..3 ;

$r = 0 = (00)_4$;

$$\begin{pmatrix} X_1(0) \\ X_1(4) \\ X_1(8) \\ X_1(12) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_0(0) \\ X_0(4) \\ X_0(8) \\ X_0(12) \end{pmatrix}$$

$r = 2 = (02)_4$;

$$\begin{pmatrix} X_1(2) \\ X_1(6) \\ X_1(10) \\ X_1(14) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_0(2) \\ X_0(6) \\ X_0(10) \\ X_0(14) \end{pmatrix}$$

$r = 1 = (01)_4$;

$$\begin{pmatrix} X_1(1) \\ X_1(5) \\ X_1(9) \\ X_1(13) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_0(1) \\ X_0(5) \\ X_0(9) \\ X_0(13) \end{pmatrix}$$

$r = 3 = (03)_4$;

$$\begin{pmatrix} X_1(3) \\ X_1(7) \\ X_1(11) \\ X_1(15) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_0(3) \\ X_0(7) \\ X_0(11) \\ X_0(15) \end{pmatrix}$$

- Pour $i = 2$; $r = (r_1,r_0)_4$: 0, 4, 8, 12 (car $r_{2-i} = r_0 = 0$) ; $C_2(r) = \left[\sum_{q=0}^{0} 4^q . r_{1-q} \right] .4^0 = r_1$;

$r = 0 = (00)_4$; $C_2(r) = r_1 = 0$;

$$\begin{pmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \\ X_2(3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{pmatrix}$$

$r = 8 = (20)_4$; $C_2(r) = r_1 = 2$;

$$\begin{pmatrix} X_2(8) \\ X_2(9) \\ X_2(10) \\ X_2(11) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_1(8).W_{16}^0 \\ X_1(9).W_{16}^2 \\ X_1(10).W_{16}^4 \\ X_1(11).W_{16}^6 \end{pmatrix}$$

$r = 4 = (10)_4$; $C_2(r) = r_1 = 1$;

$$\begin{pmatrix} X_2(4) \\ X_2(5) \\ X_2(6) \\ X_2(7) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_1(4).W_{16}^0 \\ X_1(5).W_{16}^1 \\ X_1(6).W_{16}^2 \\ X_1(7).W_{16}^3 \end{pmatrix}$$

$r = 12 = (30)_4$; $C_2(r) = r_1 = 3$;

$$\begin{pmatrix} X_2(12) \\ X_2(13) \\ X_2(14) \\ X_2(15) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \times \begin{pmatrix} X_1(12).W_{16}^0 \\ X_1(13).W_{16}^3 \\ X_1(14).W_{16}^6 \\ X_1(15).W_{16}^9 \end{pmatrix}$$

Cela nous permet d'avoir le diagramme de fluence suivant :

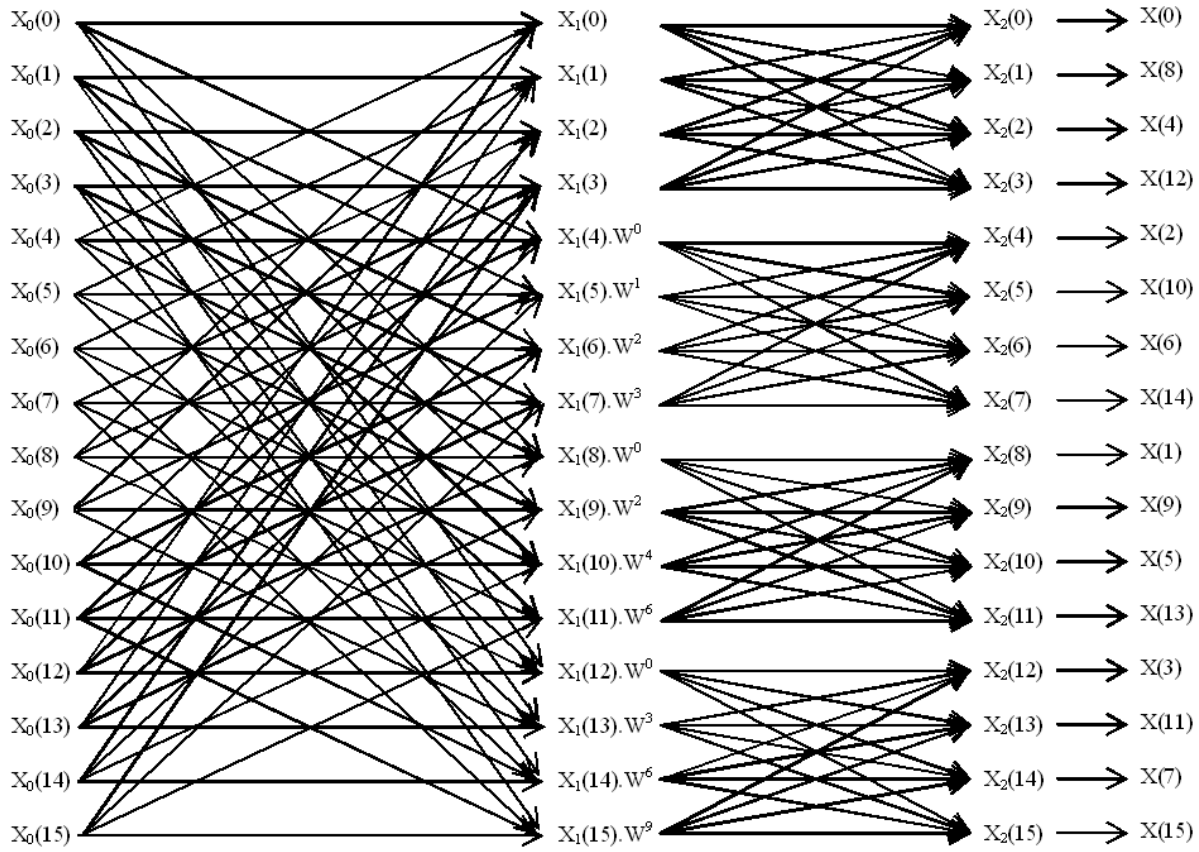


Figure I.7. Graphe de fluence de la FFT4 (par E.T.) pour $N = 16 = 4^2$.

4. Conclusion

Dans ce chapitre nous avons introduit la transformée de Fourier. Nous avons ensuite donné sa définition, présenté son aspect dimensionnel, décrit son adaptation sur des calculateurs numériques et ensuite exposé son importance reflétée par le vaste domaine de ses différentes applications. Un développement de la TFR en base arbitraire *B* suivant l'entrelacement fréquentiel et basé sur l'ACT s'en est suivi [49]. Dans [49] nous avons apporté une amélioration significative à l'ACT et qui permet de réduire, presque de 50 %, le calcul ou la lecture à partir d'une 'lookup table' des coefficients de rotation *W*. Ceci accentuera encore plus la réduction du temps d'exécution d'une TFR. Le développement de la TFR suivant l'entrelacement temporel a été ensuite arboré en utilisant les bases 2 et 4 qui sont les plus fréquemment utilisées car elles présentent des calculs simples et efficaces. Les diagrammes de fluence résultant de ces calculs formeront les fondements de plusieurs architectures parallèles calculant une TFR et qui seront présentées dans le chapitre suivant.

Chapitre II

Implémentation de la FFT sur des architectures parallèles très performantes

1. Introduction

Le temps de réponse étant un facteur capital dans la plupart des applications du traitement du signal, trois choix possibles qui affectent directement la vitesse des calculs exécutés, donc le temps de réponse, sont éventuellement proposés pour traiter une certaine application. Ce sont l'utilisation d'un ordinateur à usage général, l'utilisation d'un système à base d'un processeur de traitement du signal (D.S.P. pour Digital Signal Processor) ou l'utilisation d'une implantation dédiée à la fonction à calculer. Comparée aux deux premières alternatives, cette dernière est la solution qui permet le temps de traitement le plus court. Une implantation dédiée n'est autre que la mise en œuvre sur silicium d'un algorithme déjà défini. Elle doit être efficace dans son fonctionnement, c.à.d. permettre un taux élevé d'utilisation de ses différentes composantes et plus particulièrement son ou ses unités arithmétiques. Les progrès technologiques réalisés en matière d'intégration permettent actuellement la conception de circuits VLSI spécialisés très complexes, dont ceux dédiés au calcul de la FFT [66 – 70].

Dans bon nombre de cas, plusieurs mises en œuvre d'un certain algorithme peuvent être envisagées, dépendant des flux de données à traiter, de l'organisation des calculs et des différents agencements possibles des processeurs et circuits de l'implantation. Ceci est notamment le cas des algorithmes FFT [71 – 75].

Bien que des progrès énormes aient été faits pendant les trois dernières décennies en matière d'intégration, des contraintes subsistent encore, aussi bien au niveau des communications entre les différents blocs au sein même du circuit intégré, qu'au niveau de l'intégration elle-même. Il faut néanmoins se dire que ces limites sont quotidiennement repoussées plus loin [76, 77].

Ceci dit, différentes implémentations des algorithmes FFT développés dans le chapitre précédent feront l'objet de ce deuxième chapitre. Plusieurs architectures parallèles seront proposées et une étude comparative sera faite. Les premières architectures proposées seront issues directement des graphes de fluence présentés. On apportera à ces architectures des modifications comme la réduction des processeurs élémentaires et l'homogénéisation des réseaux d'interconnexion qui permettront de contourner les problèmes liés aux contraintes technologiques.

2. Implémentation de la FFT sur des architectures parallèles étalées

L'algorithme de la FFT2 développé dans le chapitre I, décompose récursivement, en $log_2(N)$ étapes, la transformée globale à N échantillons en des transformées minimales sur 2 valeurs. Ces dernières, au nombre de ($N/2$) par étape, sont au total au nombre de $(N/2)log_2(N)$. Cette décomposition récursive, où chaque étape de calcul ne dépend que des données fournies par l'étape précédente, permet l'implémentation de la FFT2 sur des architectures parallèles où le calcul se fera sur place. Pour la FFT4 qui utilise un développement similaire à la FFT2, le nombre d'étapes est égal à $log_4(N)$ et les transformées minimales qui sont sur 4 valeurs sont au nombre total de $(N/4)log_4(N)$. Le choix des bases 2 et 4, comme expliqué dans le chapitre I, est justifié par l'aspect d'une complexité de mise en œuvre réduite pour ces bases par rapport aux autres bases possibles. Les diagrammes de fluence qui ont été déduits de ces algorithmes sont simples et mènent à différentes implémentations FFT2 et FFT4 [78, 85]. C'est d'ailleurs sur la base de ces diagrammes de fluence que seront proposées, dans ce qui suit, différentes architectures parallèles qui calculent la FFT. Cependant, avant d'aborder ces différentes architectures, quelques notions sur l'addition et la multiplication complexes qui sont les opérations qui interviennent dans le calcul d'une FFT sont introduites ci-dessous.

L'addition et la soustraction complexes qui sont en réalité composées respectivement de 2 additions et 2 soustractions réelles seront notées indistinctement par AC (la soustraction est en fait une addition d'un nombre négatif). La multiplication complexe, quant à elle notée MC, est composée de 4 multiplications et de 2 additions réelles. En effet, pour quatre nombres complexes A, B, C et D, tels que :

$A = A_r + j\,A_i$; $B = B_r + j\,B_i$; $C = C_r + j\,C_i$; $D = D_r + j\,D_i$

on a l'AC : $C = A + B = C_r + j\,C_i = (A_r + B_r) + j\,(A_i + j\,B_i)$

et la MC : $D = A*B = D_r + j\,D_i = (A_r*B_r - A_i*B_i) + j\,(A_r*B_i + A_i*B_r)$

Les implémentations de ces 2 opérations, représentées, avec leurs symboles, sur les figures II.1. et II.2., sont donc à base d'additionneurs et de multiplieurs réels.

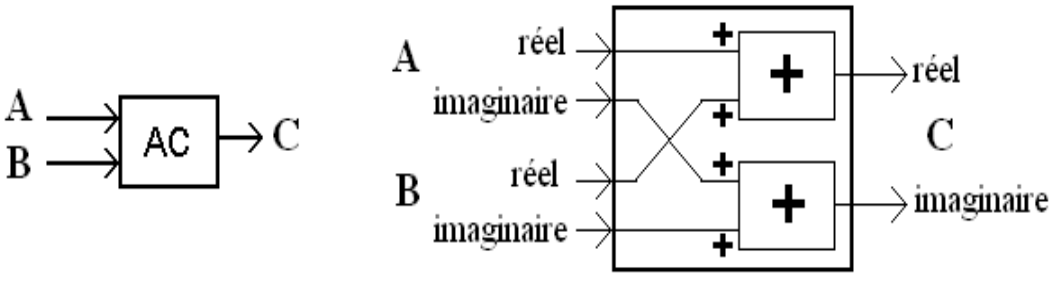


Figure II.1. Symbole et structure d'un additionneur complexe à base d'opérateurs réels.

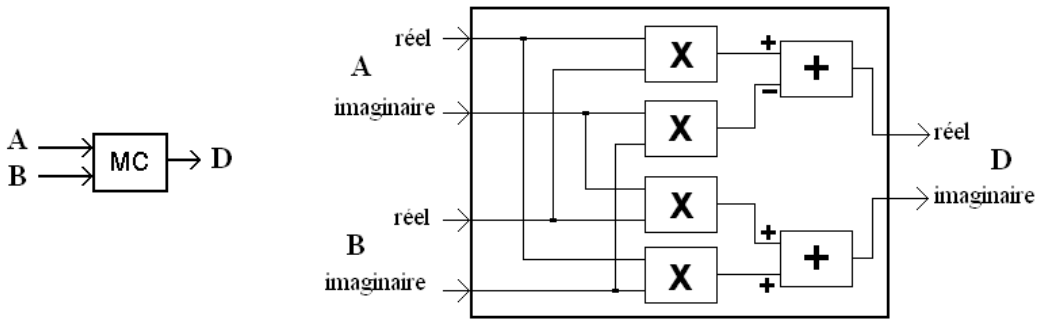


Figure II.2. Symbole et structure d'un multiplieur complexe à base d'opérateurs réels.

2.1. Implémentation de la FFT2 sur des architectures parallèles étalées

L'architecture de la FFT2 parallèle et étalée présente une structure en pipeline et est directement déduite des graphes de fluence de celle-ci [78, 85]. Pour un nombre d'échantillons temporels *N*, la FFT2 étalée peut être implémentée selon la structure du schéma bloc donné sur la figure II.3. Elle est étalée sur $log_2(N)$ rangées. Les échantillons arrivent en parallèle et en même temps sur les entrées des cellules de la rangée limitrophe de gauche et les composantes spectrales sont récupérées simultanément aux sorties des cellules de la rangée frontalière de droite. Des réseaux d'interconnexion réguliers et identiques sont insérés entre les différentes colonnes adjacentes des cellules ou processeurs élémentaires (PE).

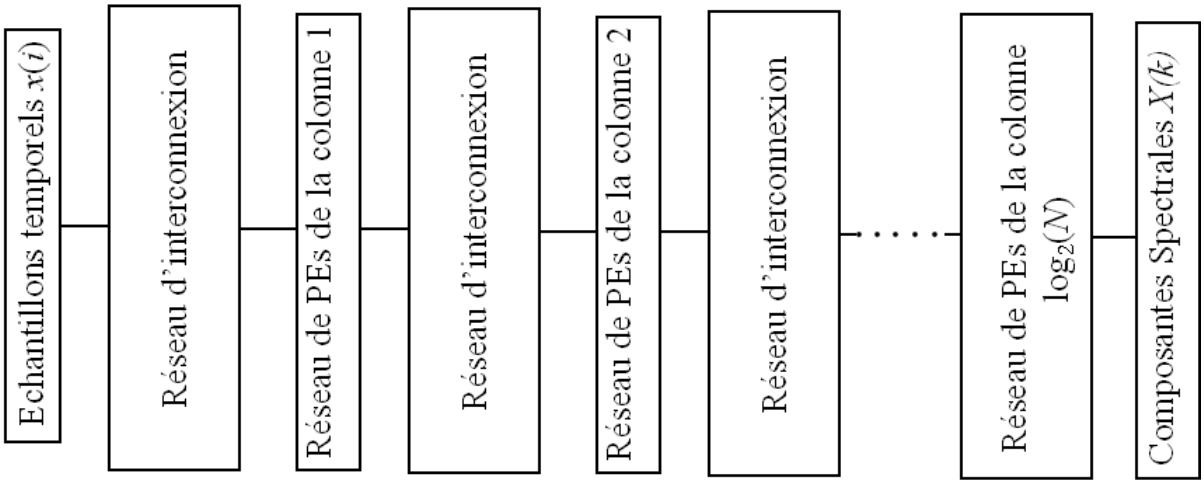


Figure II.3. Schéma bloc de l'architecture parallèle, pipelinée et étalée de la FFT2.

Le nombre de rangées de PEs, de PEs par rangée et la structure des réseaux d'interconnexion étant directement liés au nombre d'échantillons *N*, quelques implémentations FFT2, pour différentes valeurs de *N*, sont présentées et données en exemples dans ce qui suit pour clarifier le schéma bloc de la figure II.3.

La première architecture de la FFT2 proposée est pour un nombre d'échantillons N = 8. Elle est basée directement sur le graphe de fluence de la figure I.3 et peut être implémentée selon la structure donnée sur la figure II.4. Elle est étalée sur $log_2(8)$ = 3 rangées. Sur la figure II.4, les états intermédiaires ont été détaillées pour les 3 étapes, afin de pouvoir suivre l'évolution des calculs jusqu'au vecteur spectre final.

On remarque que chaque PE fait une opération de multiplication (e_2.W) suivie d'une opération d'accumulation (e_1+e_2.W ou e_1-e_2.W) où e_1 et e_2 représentent les données présentes sur ses 2 entrées.

Sur le circuit de la figure II.4, on remarque la présence d'une homogénéité et régularité des réseaux d'interconnexion entre les processeurs élémentaires des colonnes adjacentes. Ces réseaux d'interconnexion ont été adéquatement ordonnancés pour permettre une redistribution simple et parfaite des résultats intermédiaires. Cette redistribution est appelée en anglais « Perfect Shuffle Exchange » [79 – 80].

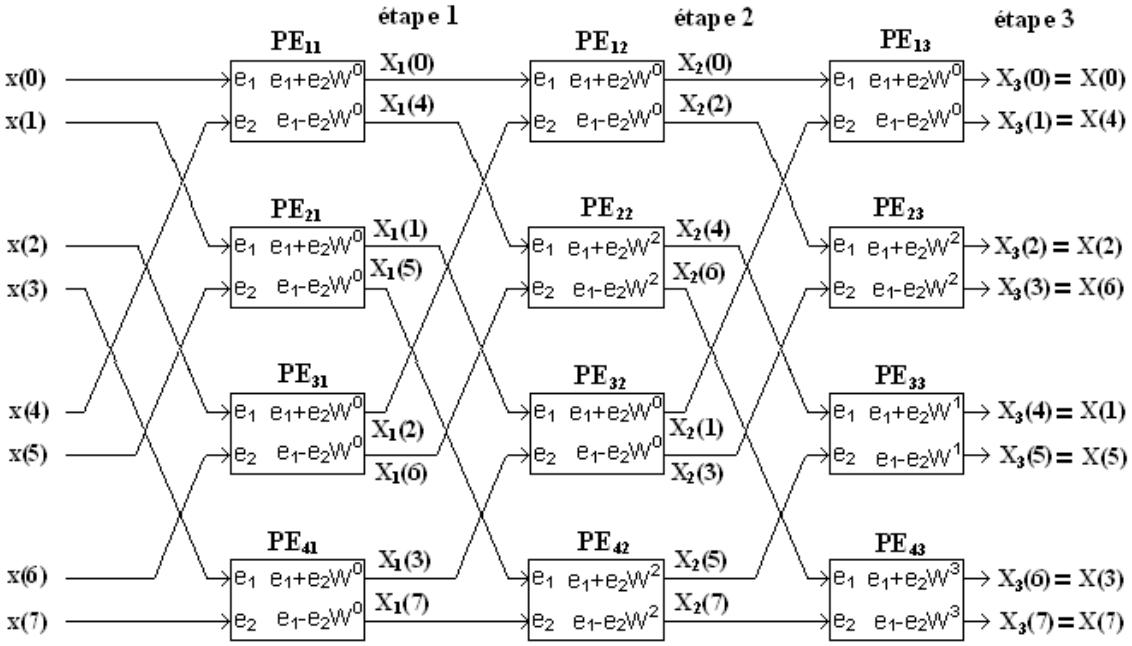


Figure II.4. Architecture parallèle, pipelinée, étalée de la FFT2 pour N = 8.

Pour une structure figée, calculant répétitivement la même FFT sur un flux continuel de trames de N=8 échantillons, les coefficients W restent les mêmes et peuvent alors être pré-calculés et chargés chacun dans un registre interne au niveau de chaque PE.

Chaque PE, qu'on notera désormais par PE_FFT2, effectue 3 opérations complexes : une multiplication et deux additions. La structure interne d'un tel processeur est donnée sur la figure II.5. Le papillon, présent sur les graphes de fluence de la FFT2, est représenté par le bloc de droite regroupant les deux additionneurs complexes.

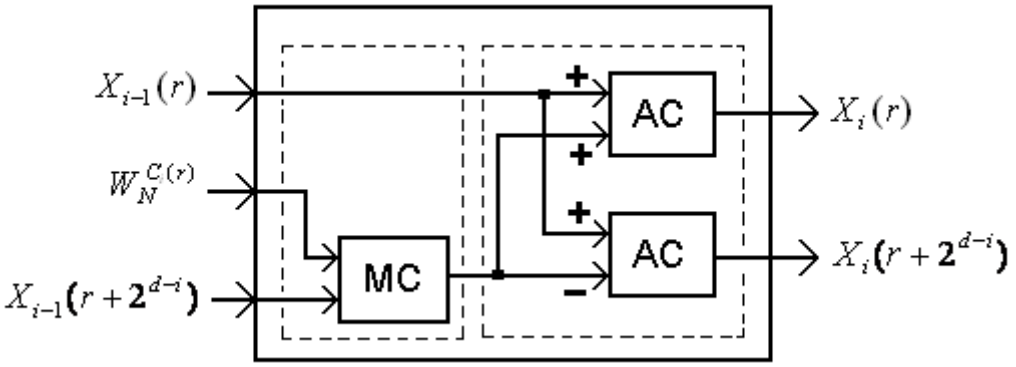


Figure II.5. Structure interne du PE_FFT2.

Comme il a été mentionné plus haut, les opérations complexes sont faites à partir d'opérateurs réels. Il serait donc intéressant de donner, à partir de ces opérateurs réels, la structure interne du PE_FFT2. Cela permettra alors d'avoir une idée claire sur la complexité de la mise en œuvre sur silicium de ce dernier. Pour arriver à cette structure, quelques passages élémentaires de calcul sont à effectuer. En effet les expressions à calculer, données par les formules (I.56) et (I.58) et qui sont reprises dans l'expression suivante :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{d-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & W_N^{C_i(r)} \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+2^{d-i}) \end{pmatrix} = \begin{pmatrix} X_{i-1}(r) + X_{i-1}(r+2^{d-i}).W_N^{C_i(r)} \\ X_{i-1}(r) - X_{i-1}(r+2^{d-i}).W_N^{C_i(r)} \end{pmatrix}$$

peuvent être décomposées en des expressions simples menant à l'implémentation recherchée. Pour bien visualiser les calculs, les variables complexes sont exprimées par leurs parties réelles et imaginaires, avec :

a_r = partie réelle de $X_{i-1}(r)$ — a_i = partie imaginaire de $X_{i-1}(r)$

b_r = partie réelle de $X_{i-1}(r+2^{d-i})$ — b_i = partie imaginaire de $X_{i-1}(r+2^{d-i})$

c_r = partie réelle de $W_N^{C_i(r)}$ — c_i= partie imaginaire de $W_N^{C_i(r)}$

A_r = partie réelle de $X_i(r)$ — A_i = partie imaginaire de $X_i(r)$

B_r = partie réelle de $X_i(r+2^{d-i})$ — B_i = partie imaginaire de $X_i(r+2^{d-i})$

Les expressions à calculer sont donc de la forme :

$(A_r + j\ A_i) = (a_r + j\ a_i) + (b_r + j\ b_i)\ (c_r + j\ c_i) = [a_r + (b_r.c_r - b_i.c_i)] + j\ [a_i + (b_r.c_i + b_i.c_r)]$

$(B_r + j\ B_i) = (a_r + j\ a_i) - (b_r + j\ b_i)\ (c_r + j\ c_i) = [a_r - (b_r.c_r - b_i.c_i)] + j\ [a_i - (b_r.c_i + b_i.c_r)]$

Cela nécessite alors quatre multiplieurs et six additionneurs réels. La structure interne du PE_FFT2 avec des opérateurs réels est donnée sur la figure II.6.

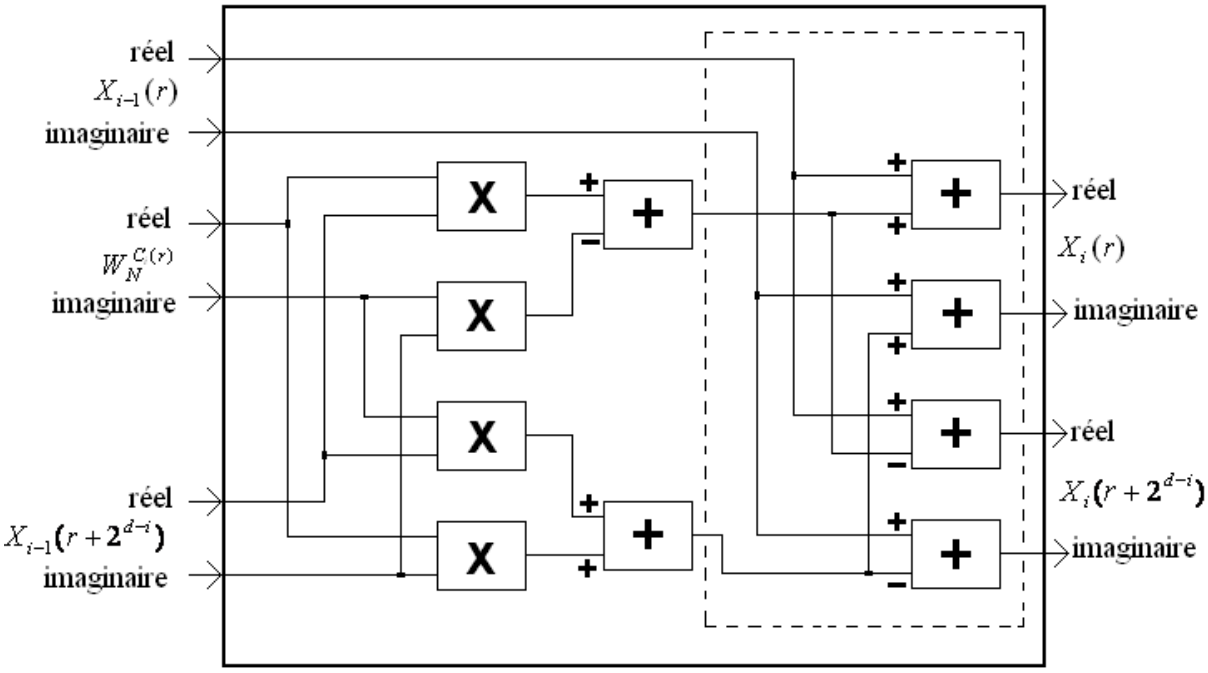


Figure II.6. Structure interne du PE_FFT2 avec des opérateurs réels.

Pour un nombre d'échantillons *N*, le nombre total d'opérateurs réels pour une implémentation FFT2 étalée est de :

(N/2) $log_2(N)$*4 multiplieurs + (N/2) $log_2(N)$*6 additionneurs =

(2N) $log_2(N)$ multiplieurs + (3N) $log_2(N)$ additionneurs (II.1)

Un facteur important à mettre en évidence est le temps de réponse (TR) du processeur. En notant t_m et t_a les temps de réponse de respectivement le multiplieur et l'additionneur réels, le temps de réponse du PE_FFT2 est de :

$$TR\_PE\_FFT2 = t_m + 2.t_a \quad \text{(II.2)}$$

Pour un nombre d'échantillons *N* plus élevé, l'architecture de l'implémentation peut être similairement déduite de celle à $N = 8$. Le nombre de PEs et celui des d'opérateurs réels peuvent facilement être calculés à partir des formules déjà évoquées. Le réseau d'interconnexion doit être aussi adapté en conséquence. A titre d'exemple, la figure II.7 donne l'architecture parallèle de la FFT2 pour $N = 16$. Les coefficients $W_{16}^{C_i(r)}$ correspondant à chacun des PEs y sont reportés leurs entrées.

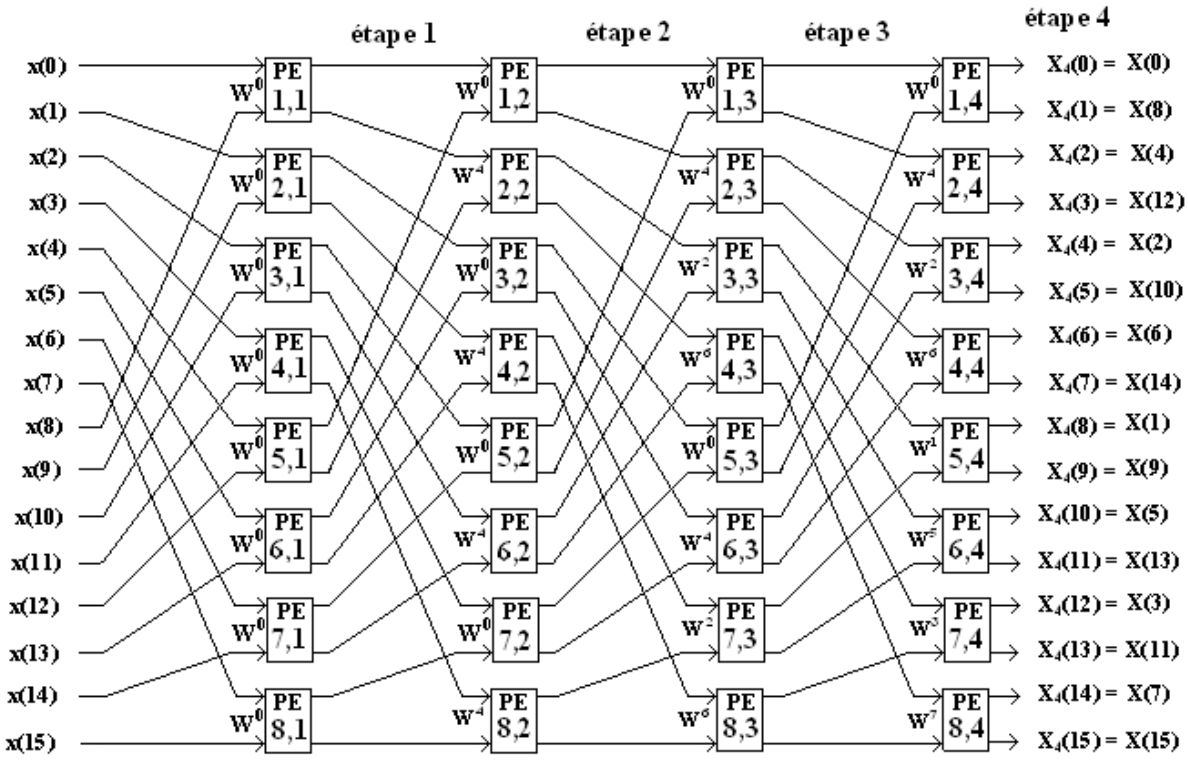


Figure II.7. Architecture parallèle, pipelinée et étalée de la FFT2 pour $N = 16$.

2.2. Implémentation de la FFT4 sur des architectures parallèles étalées

Pour un nombre d'échantillons donné $N = 4^d$, la FFT4 est la décomposition récursive de la transformée globale en $log_4(N)$ étapes avec $(N/4)$ transformées minimales par étape, chacune sur 4 valeurs. On peut déduire alors son implémentation étalée, similairement à celle de la FFT2, qui nécessitera $(N/4)log_4(N)$ PEs. La figure II.8 donne, comme exemple, l'architecture parallèle, pipelinée et étalée de la FFT4 pour $N = 16$. Elle est dérivée du graphe de fluence donné sur la figure I.7. Deux étages avec 4 PEs chacun sont alors nécessaires. Les coefficients de transformation $W_{16}^{C_i(r)}$ correspondant à chaque PE y sont reportés au niveau de chacun d'eux. Si plusieurs FFT4 sont à calculer les différents $W_{16}^{C_i(r)}$ peuvent être pré-calculés et stockés dans des registres au niveau des PEs pour être réutilisés.

Comme ça a été montré dans le chapitre I (Section 3.4.2), chaque PE de la FFT4, et qu'on notera par PE_FFT4, effectue 11 opérations complexes : 3 multiplications et 8 additions. La structure interne du PE_FFT4 est donnée sur la figure II.9. Le papillon, présent sur les graphes de fluence de la FFT4, est délimité par le bloc de droite englobant les 8 additionneurs complexes et l'opérateur **j**.

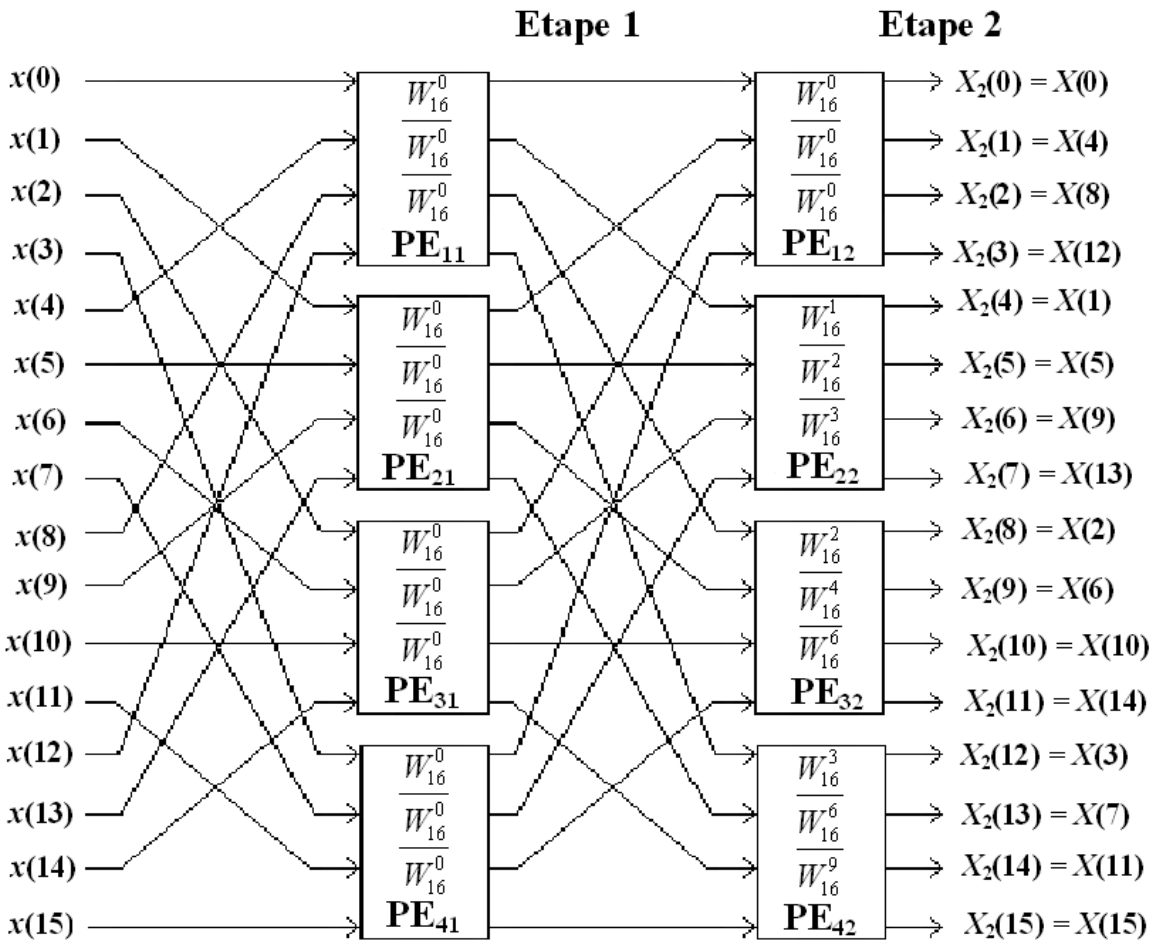


Figure II.8. Architecture parallèle, pipelinée et étalée de la FFT4 pour N =16.

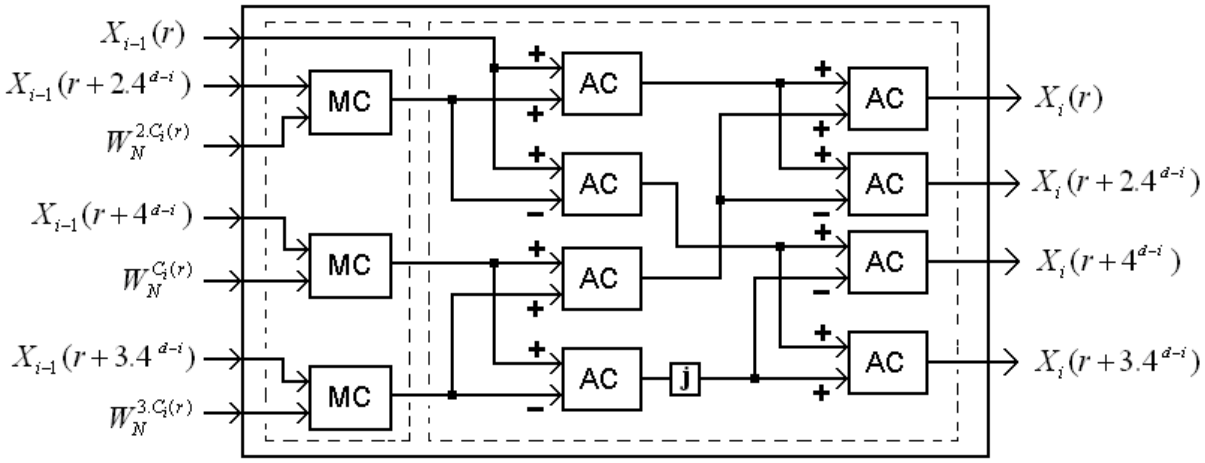


Figure II.9. Structure interne du PE_FFT4.

Comme pour le PE_FFT2, le PE_FFT4 est en réalité conçu à partir d'opérateurs réels. Quelques passages de calculs très simples, similaires à ceux élaborés pour aboutir au PE_FFT2, à base d'opérateurs réels sont donnés ci-dessous. Les résultats de ces calculs permettent de dégager la structure du PE_FFT4 donnée sur la figure II.10. Cette structure comporte 12 multiplieurs réels et 22 additionneurs réels.

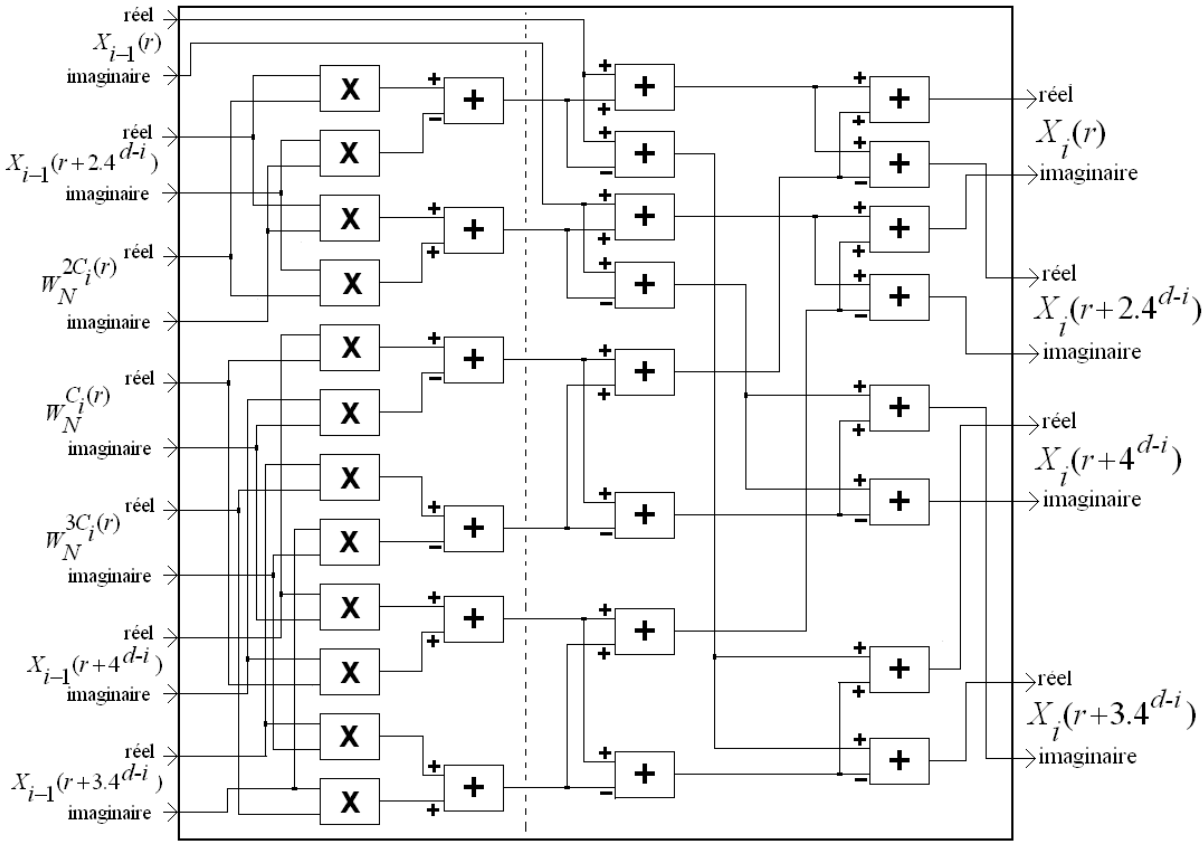


Figure II.10. Structure interne du PE_FFT4 à base d'opérateurs réels.

Pour arriver à cette structure, nous avons utilisé la formule I.66 qu'on réécrit ici :

$$\begin{pmatrix} X_i(r) \\ X_i(r+4^{d-i}) \\ X_i(r+2.4^{d-i}) \\ X_i(r+3.4^{d-i}) \end{pmatrix} = \begin{pmatrix} [X_{i-1}(r)+X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]+[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}+X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)-X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]-j[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}-X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)+X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]-[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}+X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \\ [X_{i-1}(r)-X_{i-1}(r+2.4^{d-i}).W_N^{2.C_i(r)}]+j[X_{i-1}(r+4^{d-i}).W_N^{C_i(r)}-X_{i-1}(r+3.4^{d-i}).W_N^{3.C_i(r)}] \end{pmatrix}$$

En posant :

A_r = partie réelle de $X_i(r)$
A_i = partie imaginaire de $X_i(r)$
B_r = partie réelle de $X_i(r+4^{d-i})$
B_i = partie imaginaire de $X_i(r+4^{d-i})$
C_r = partie réelle de $X_i(r+2.4^{d-i})$
C_i = partie imaginaire de $X_i(r+2.4^{d-i})$
D_r = partie réelle de $X_i(r+3.4^{d-i})$
D_i = partie imaginaire de $X_i(r+3.4^{d-i})$
a_r = partie réelle de $X_{i-1}(r)$
a_i = partie imaginaire de $X_{i-1}(r)$
b_r = partie réelle de $X_{i-1}(r+4^{d-i})$
b_i = partie imaginaire de $X_{i-1}(r+4^{d-i})$
c_r = partie réelle de $X_{i-1}(r+2.4^{d-i})$
c_i = partie imaginaire de $X_{i-1}(r+2.4^{d-i})$
d_r = partie réelle de $X_{i-1}(r+3.4^{d-i})$
d_i = partie imaginaire de $X_{i-1}(r+3.4^{d-i})$
e_r = partie réelle de $W_N^{C_i(r)}$
e_i = partie imaginaire de $W_N^{C_i(r)}$
f_r = partie réelle de $W_N^{2C_i(r)}$
f_i = partie imaginaire de $W_N^{2C_i(r)}$
g_r = partie réelle de $W_N^{3C_i(r)}$
g_i = partie imaginaire de $W_N^{3C_i(r)}$

on arrive aux formules suivantes :

$$(A_r + j\,A_i) = [(a_r + j\,a_i) + (c_r + j\,c_i)(f_r + j\,f_i)] + [(b_r + j\,b_i)(e_r + j\,e_i) + (d_r + j\,d_i)(g_r + j\,g_i)]$$
$$(A_r + j\,A_i) = [a_r + (c_r.f_r - c_i.f_i) + j\,\{a_i + (c_r.f_i + c_i.f_r)\}] + [\{(b_r.e_r - b_i.e_i) + (d_r.g_r - d_i.g_i)\} + j\,\{(b_r.e_i + b_i.e_r) + (d_r.g_i + d_i.g_r)\}]$$

$$(B_r + j\,B_i) = [(a_r + j\,a_i) - (c_r + j\,c_i)(f_r + j\,f_i)] - j\,[(b_r + j\,b_i)(e_r + j\,e_i) - (d_r + j\,d_i)(g_r + j\,g_i)]$$
$$(B_r + j\,B_i) = [a_r - (c_r.f_r - c_i.f_i) + j\,\{a_i - (c_r.f_i + c_i.f_r)\}] - j\,[\{(b_r.e_r - b_i.e_i) - (d_r.g_r - d_i.g_i)\} + j\,\{(b_r.e_i + b_i.e_r) - (d_r.g_i + d_i.g_r)\}]$$

$$(C_r + j\,C_i) = [(a_r + j\,a_i) + (c_r + j\,c_i)(f_r + j\,f_i)] - [(b_r + j\,b_i)(e_r + j\,e_i) + (d_r + j\,d_i)(g_r + j\,g_i)]$$
$$(C_r + j\,C_i) = [a_r + (c_r.f_r - c_i.f_i) + j\,\{a_i + (c_r.f_i + c_i.f_r)\}] - [\{(b_r.e_r - b_i.e_i) + (d_r.g_r - d_i.g_i)\} + j\,\{(b_r.e_i + b_i.e_r) + (d_r.g_i + d_i.g_r)\}]$$

$$(D_r + j\,D_i) = [(a_r + j\,a_i) - (c_r + j\,c_i)(f_r + j\,f_i)] + j\,[(b_r + j\,b_i)(e_r + j\,e_i) - (d_r + j\,d_i)(g_r + j\,g_i)]$$
$$(D_r + j\,D_i) = [a_r - (c_r.f_r - c_i.f_i) + j\,\{a_i - (c_r.f_i + c_i.f_r)\}] + j\,[\{(b_r.e_r - b_i.e_i) - (d_r.g_r - d_i.g_i)\} + j\,\{(b_r.e_i + b_i.e_r) - (d_r.g_i + d_i.g_r)\}]$$

qui déterminent ainsi la structure du PE_FFT4 de la figure II.10.

Pour un nombre d'échantillons N donné, le nombre total d'opérateurs réels pour implémenter la FFT4 est de :

$(N/4).log_4(N)$.12 multiplieurs + $(N/4).log_4(N)$.22 additionneurs =

$(3/2).N.log_2(N)$ multiplieurs + $(11/4).N.log_2(N)$ additionneurs (II.3)

D'après la structure du PE_FFT4, Le temps de réponse est donné par :

$$TR\_PE\_FFT4 = t_m + 3.t_a \quad \text{(II.4)}$$

t_m et t_a sont les temps de réponse de respectivement les multiplieur et additionneur réels.

Une concise comparaison entre les architectures étalées de la FFT2 et de la FFT4 nous permet de tirer quelques conclusions :

- Pour un même nombre d'échantillons N, le nombre d'étages de la FFT4 est égal à la moitié du nombre d'étages de la FFT2 et le nombre de PEs de la FFT4 par étage est égal à la moitié de celui da la FFT2. A titre d'exemple, pour N=256, la FFT4 a 4 étages et 64 PE_FFT4 par étage (4*64=256 PE_FFT4 pour toute la FFT4), alors que la FFT2 possède 8 étages et requiert 128 PE_FFT2 par étage (8*128=1024 PE_FFT2 pour toute la FFT2). Le nombre total des PE_FFT2 est donc 4 fois supérieur à celui des PE_FFT4. Cet important gain de la FFT4 par rapport à la FFT2 a été obtenu au dépend d'une complexité d'implémentation beaucoup plus grande du PE_FFT4 par rapport à celle de la FFT2 comme on le voit sur la figures II.6 et II.10.

- En surface d'implantation sur silicium, la FFT4 est meilleure que la FFT2 : un gain considérable est acquis sur la surface des réseaux d'interconnexion entre étages, plus le gain sur les surfaces occupées par les opérateurs réels. En effet, si **Sm** et **Sa** sont les surfaces occupées respectivement par un multiplieur et un additionneur réels, les surfaces occupées par les implémentations du point de vue de ces opérateurs réels sont :
  - pour la FFT2 : **S_** FFT2 = $2.N.log_2(N).$**Sm** + $3.N.log_2(N).$**Sa**
  - pour la FFT4 : **S_** FFT4 = $(3/2).N.log_2(N).$**Sm** + $(11/4).N.log_2(N).$**Sa**

Ce qui fait un gain en surface de 25% pour les multiplieurs et d'à peu près 8,5% pour les additionneurs de l'implémentation de la FFT4 par rapport à celle de la FFT2.

- Pour le temps de réponse qui un autre facteur très important et qu'on cherche toujours à améliorer, l'architecture de la FFT4 est de loin supérieure à celle de la FFT2. En effet, en appliquant les formules (II.2) et (II.4), les temps de réponse sont :
  - pour la FFT2 : TR_FFT2 = $(t_m + 2.t_a).log_2(N)$
  - pour la FFT4 : TR_FFT4 = $(t_m + 3.t_a).log_4(N) = [(t_m + 3.t_a).log_2(N)]/2$

Etant donné que la multiplication consomme plus de temps que l'addition, on peut conclure que le temps de réponse de l'architecture FFT4 est à peu près la moitié de celui de l'architecture FFT2.

A partir de cette comparaison, on peut conclure que pour des architectures parallèles étalées, l'implémentation de l'algorithme FFT4 est préférable à celle de l'algorithme FFT2 car les gains en temps de réponse et en surface d'implantation sont considérables.
Un gain en surface peut encore être apporté pour les deux types architectures FFT2 et FFT4 en implémentant les PEs de la 1ère colonne sans multiplieurs car les coefficients $W_N^{C_i(r)}$ (pour la FFT2) et $W_N^{C_i(r)}$, $W_N^{2C_i(r)}$ et $W_N^{3C_i(r)}$ (pour la FFT4) sont égaux à $W_N^0=1$.

Remarque importante : Les architectures parallèles trouvent leur importance lorsque le volume des données à traiter dépasse un certain seuil. Si le calcul est fait sur un débit de données continu, c'est-à-dire si plusieurs FFTs sont à calculer sur des trames de données adjacentes, chaque trame ayant $N=B^d$ données (B = 2 ou 4), le premier spectre relatif à la première trame est obtenu en $log_2(N)$ étapes pour une FFT2 et $log_4(N)$ étapes pour une FFT4. Il sera suivi d'un spectre par étape pour chaque nouvelle trame de données. Le spectre de la 2ème trame sera donc obtenu à l'étape $(1+d)$ et celui de la $i$ème trame à l'étape : $(i-1) + d$.

3. Implémentation de la FFT sur des architectures parallèles rebouclées ou cycliques

Le nombre de PEs est directement lié au nombre d'échantillons *N*. Quand ce dernier devient très grand, le nombre de PEs le devient encore plus. Par exemple, pour calculer une FFT à 16 millions d'échantillons ($N = 2^{24} = 4^{12}$ =16777216), presque 200 millions de PE_FFT2 ou 50 millions de PE_FFT4 seront nécessaires pour l'implémentation de l'un ou l'autre des deux algorithmes. La technologie VLSI est encore limitée pour pouvoir implanter ce nombre de cellules sur la même plaque de silicium. Des alternatives ont été étudiées pour remédier à ce problème. L'une d'elles est de n'utiliser qu'une seule rangée de PEs et de reboucler la structure sur elle-même pour permettre la réutilisation des résultats intermédiaires en les redirigeant des sorties du réseau vers ses entrées [78, 85]. Un système de contrôle dirigé par le système hôte est alors nécessaire pour fournir les commandes qui permettent :

- à la première étape, d'injecter au réseau les *N* échantillons à traiter.
- aux étapes qui suivent, sauf à la dernière, de reboucler les sorties du réseau, qui sont les états intermédiaires, vers ses entrées. Des registres doivent être prévus à la sortie des PEs pour mémoriser les résultats intermédiaires.
- à la fin de la dernière étape, de fournir le spectre final.

Les coefficients *W* doivent être appliqués d'une manière appropriée pour chaque étape de calcul. La figure II.11 présente le schéma bloc d'une telle structure.

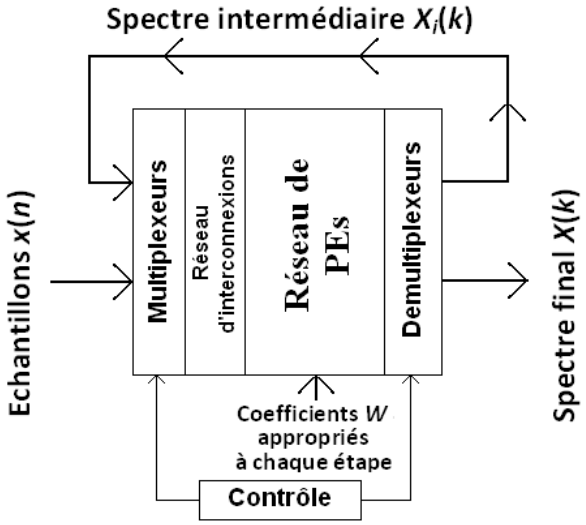


Figure II.11. Structure générale d'une architecture parallèle rebouclée.

La figure II.12 donne la structure d'une architecture parallèle rebouclée pour le calcul de la FFT2 pour un nombre d'échantillons $N = 8$. Les multiplexeurs, qui se trouvent à l'entrée du réseau de processeurs, permettent de sélectionner les données externes (début de l'étape 1) ou les composantes spectrales intermédiaires (débuts des étapes 2 et 3). A la fin de l'étape 3,

le spectre final est récupéré à la sortie des démultiplexeurs qui, à la fin des étapes 1 et 2, rebouclaient les résultats intermédiaires vers l'entrée du réseau de processeurs.

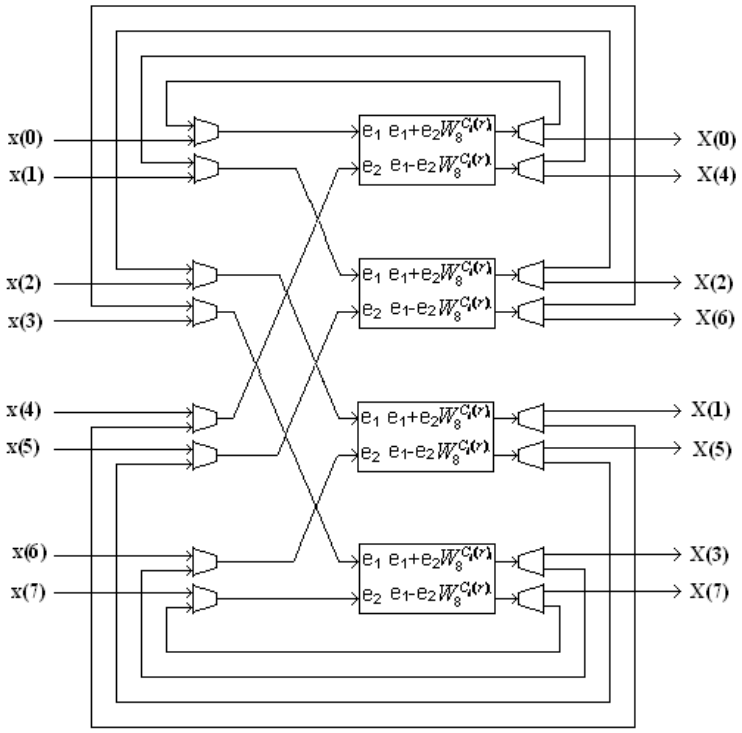


Figure II.12. Architecture parallèle, pipelinée et rebouclée de la FFT2 pour $N = 8$.

Le nombre total de PEs utilisés a donc été divisé par le nombre d'étapes (3 pour $N = 8$, $log_2(N)$ pour le cas général). Ce gain a son importance en surface d'implantation si N devient important. Par exemple, pour $N = 1024 = 2^{10}$, 5120 PEs sont nécessaires pour une architecture étalée, alors que seulement 512 PEs le sont pour une architecture rebouclée. Cela fait un gain de 90% sur aussi bien le nombre de PEs que sur le nombre de réseaux d'interconnexion, et donc sur la surface silicium totale d'implantation. Il faut juste noter que bien que N multiplexeurs et N démultiplexeurs soient ajoutés au circuit, le gain en surface d'implantation est remarquable.

Comme pour le cas d'une architecture étalée, le nombre d'étapes pour le calcul d'une FFT2 est égal à $log_2(N)$, mais celui de m FFT2s sur m trames de données adjacentes sera égal à $m.log_2(N)$. Cela est dû au fait qu'une nouvelle FFT2 ne peut être entamée avant la fin de la précédente. Il faut aussi noter que la boucle de retour avec ses multiplexeurs et ses démultiplexeurs apporte une latence additionnelle : la durée d'une étape dans le cas d'une architecture rebouclée est donc plus importante que celle d'une architecture étalée.

Pour une architecture FFT4, le même principe peut être appliqué et les mêmes remarques peuvent être exprimées. Le nombre de processeurs PE_FFT4 est alors divisé par $log_4(N)$ par rapport à une architecture étalée et le nombre d'étapes pour le calcul de m FFT4s

successives sera de $m.log_4(N)$. La figure II.13 donne la structure d'une architecture parallèle rebouclée pour le calcul de la FFT4 pour un nombre d'échantillons $N = 16$.

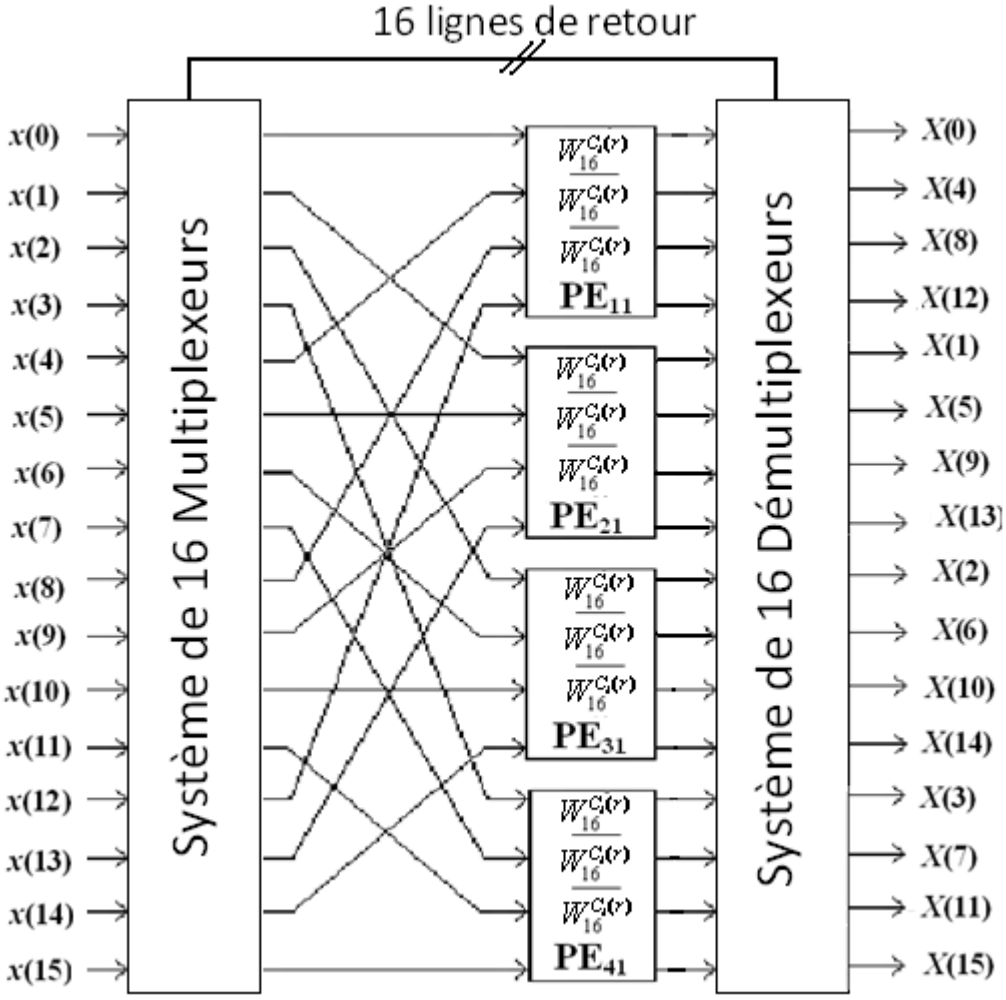


Figure II.13. Architecture parallèle, pipelinée et rebouclée de la FFT4 pour $N = 16$.

4. Implémentation de la FFT sur des architectures pipelinées linéaires

4.1. Implémentation de la FFT2 sur des architectures pipelinées linéaires

L'architecture de la FFT parallèle et cyclique permet de réduire considérablement le nombre de PEs par rapport à l'architecture étalée. Les architectures pipelinées et linéaires, sujet de plusieurs travaux de recherche [81 – 85], permettent de réduire encore beaucoup plus ce nombre. Quelques architectures FFT2 trouvées dans la littérature [86] et qu'on présente succinctement ci-dessous montrent les qualités et avantages résultant de telles architectures.

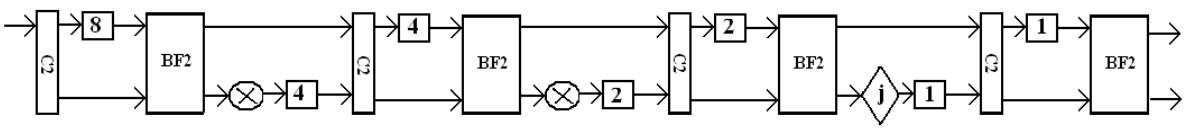


Figure II.14. Architecture pipelinée et linéaire de la FFT2 pour N =16 avec commutateurs et registres à décalage [86].

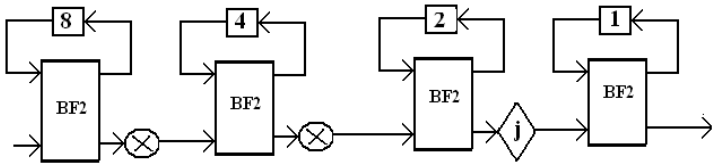


Figure II.15. Architecture pipelinée et linéaire de la FFT2 pour N =16 avec registres à décalage en Feedback [86].

Ces architectures permettent de ne garder qu'une seule ligne de PEs, composée de $log_2(N)$ PEs. Le nombre total de PEs utilisés dans une architecture FFT2 étalée est donc divisé par $(N/2)$. Cela représente un gain en nombre de PEs encore beaucoup plus important que celui apporté par l'architecture rebouclée. Si on reprend l'exemple de $N = 1024 = 2^{10}$, 10 PEs seulement seront nécessaires pour une architecture FFT2 linéaire contre 512 PEs pour une architecture rebouclée et 5120 PEs pour une architecture étalée. En ne prenant en compte que les PEs, l'architecture FFT2 linéaire présente alors un gain en surface silicium de 99.8% par rapport à l'architecture étalée et 98% par rapport à l'architecture rebouclée.

Le circuit de la figure II.14. désigné par «Radix-2 Multi-path Delay Commutator (R2MDC) », pour réseau FFT2 à chemin multiple avec commutateurs et éléments de retard, est l'approche directe d'une implémentation pipelinée et linéaire où la séquence d'entrée est répartie, grâce au 1er commutateur C2, en deux flux ou suites d'échantillons. Ces derniers vont arriver sur les papillons BF2 (butterfly FFT2) avec des temps adéquatement synchrones grâce à des éléments de retard qui sont des registres à décalage en nombre de 8 avant le 1er BF2, 4 avant le 2ème BF2 et ainsi de suite comme montré sur la figure II.14. La figure II.15 présente le même réseau avec un retour ou feedback des registres à décalage sur les entrées des papillons BF2. Ceci est fait dans un souci de réduire le nombre des registres à décalage et éliminer les commutateurs, d'où l'appellation «Radix-2 Single-path Delay Feedback (R2SDF)», pour réseau FFT2 à chemin unique et feedback des éléments de retard.

Sur les deux circuits présentés, les papillons BF2, les multiplieurs complexes et l'opérateur j sont définis séparément contrairement aux réseaux qu'on proposera où les PEs englobent tous ces éléments.

La structure proposée sur la figure II.16.a est un exemple d'architecture pipelinée et linéaire de la FFT2 qui travaille sur un nombre d'échantillons $N = 8$. N est choisi limité à 8 afin de pouvoir suivre clairement et bien comprendre l'évolution des calculs. Elle utilise des PE_FFT2 dont la structure est donnée sur la figure II.6. Les figures II.16.b et II.16.c représentent les modes de fonctionnement des commutateurs C_1, C_2 et C_3.

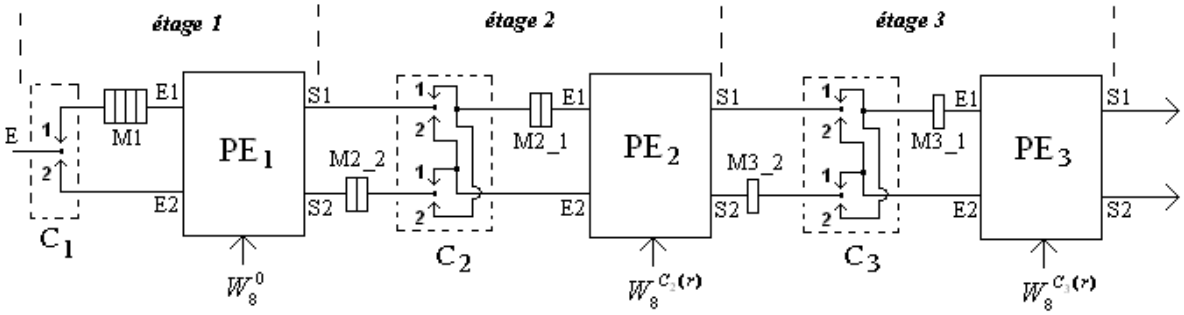


Figure II.16.a. Architecture pipelinée et linéaire de la FFT2 pour N =8.

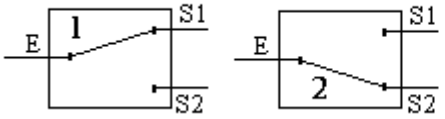


Figure II.16.b.
Etats possibles du commutateur C_1.

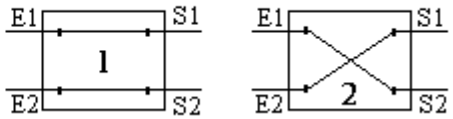


Figure II.16.c.
Etats possibles des commutateurs C_2 et C_3.

Pour N = 8, le circuit comporte 3 PEs, un module M1 de 4 registres, 2 autres modules, M2_1 et M2_2, de 2 registres chacun et 2 simples registres notés M3_1 et M3_2. Le rôle des registres est de faire introduire des retards permettant la synchronisation des paires d'échantillons appropriés aux entrées des différents PEs. A chaque étape, chaque registre fait transférer sa donnée vers l'élément se trouvant à sa droite et est chargé par la donnée présente à son entrée où elle est retenue jusqu'à la prochaine étape. Le circuit possède aussi 3 commutateurs C_1, C_2 et C_3, qui travaillent selon l'une des 2 positions données sur les figures II.16.b et II.16.c, et qui permettent une transmission adéquate des données vers les entrées des PEs. Le choix de l'une ou l'autre des 2 positions est contrôlé par un signal de commande. Les coefficients W correspondants à chaque étape sont injectés aux processeurs aux moments adéquats.

Fonctionnement du réseau :

- Les échantillons se présentent séquentiellement à l'entrée E du réseau dans l'ordre x(0), x(1), x(2), … x(7)
- Pendant les 4 premières étapes, le commutateur C_1 est en position 1. Les quatre premières données sont alors stockées au fur et à mesure dans le module des registres M1.
- Pendant les 4 prochaines étapes (de la 5ème à la 8ème étape), le commutateur C_1 bascule vers la position 2, et les actions suivantes sont effectuées :

- $X_1(0)$ et $X_1(4)$ sont calculés ($E1_{PE1} = x(0)$ et $E2_{PE1} = x(4)$) et sont enregistrés dans respectivement M2_1 et M2_2, Puis un décalage d'une position vers la gauche se fait au niveau de M1.

- Les étapes 6, 7 et 8 vont permettre le calcul du reste du vecteur intermédiaire X_1 ($X_1(1)$ et $X_1(5)$ [étape 6] ; $X_1(2)$ et $X_1(6)$ [étape 7]; $X_1(3)$ et $X_1(7)$ [étape 8]).

- Puisque tous les processeurs sont en train de travailler en pipeline et en même temps, le calcul de X_2 commence pendant l'étape 7 et celui de X_3 pendant l'étape 8.
- Le spectre intermédiaire X_2 est fini d'être calculé à l'étape 10, et le spectre X_3 à l'étape 11. X_3 nécessite un miroir binaire de ses composantes pour avoir le spectre final X.

Les tableaux II.1 permettent de visualiser clairement toutes les étapes de calcul. Les entrées des PEs sont notées par $E1_{PEj}$ et $E2_{PEj}$, et leurs sorties par $S1_{PEj}$ et $S2_{PEj}$, j : 1…3. Les sorties des registres sont mentionnées par S_{M1} , S_{M2_1}, S_{M2_2}, S_{M3_1} et S_{M3_2}.

Etape 4 : (C1 en 1)

M1				S_{PE1}
x(3)	x(2)	x(1)	x(0)	

Etape 5 : (C1 en 2) (C2 en 1)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}
	x(3)	x(2)	x(1)	$X_1(0)$			$X_1(0)$		
				$X_1(4)$	$X_1(4)$				

Etape 6 : (C1 en 2) (C2 en 1)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}
		x(3)	x(2)	$X_1(1)$			$X_1(1)$	$X_1(0)$	
				$X_1(5)$	$X_1(5)$	$X_1(4)$			

Etape 7 : (C1 en 2) (C2 en 2) (C3 en 1)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
			x(3)	$X_1(2)$			$X_1(4)$	$X_1(1)$	$X_2(0)$		$X_2(0)$	
				$X_1(6)$	$X_1(6)$	$X_1(5)$			$X_2(2)$	$X_2(2)$		

Etape 8 : (C1 en 2) (C2 en 2) (C3 en 2)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
				$X_1(3)$			$X_1(5)$	$X_1(4)$	$X_2(1)$		$X_2(2)$	$X_3(0)$
				$X_1(7)$	$X_1(7)$	$X_1(6)$			$X_2(3)$	$X_2(3)$		$X_3(1)$

Etape 9 : (C1 en 1) (C2 en 1) (C3 en 1)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
								$X_1(5)$	$X_2(4)$		$X_2(4)$	$X_3(2)$
						$X_1(7)$			$X_2(6)$	$X_2(6)$		$X_3(3)$

Etape 10 : (C1 en 1) (C2 en 1) (C3 en 2)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
									$X_2(5)$		$X_2(6)$	$X_3(4)$
									$X_2(7)$	$X_2(7)$		$X_3(5)$

Etape 11 : (C1 en 1) (C2 en 2) (C3 en 1)

M1				S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
												$X_3(6)$
												$X_3(7)$

Tableaux II.1. Sorties des PEs et contenus des différents registres M*i* pendant les différentes étapes de calcul pour l'architecture pipelinée et linéaire de la FFT2 pour N =8.

Pour $N = 8$, il a donc fallu 11 étapes pour calculer le spectre X.

Le circuit de la figure II.16.a peut être généralisé pour un nombre quelconque d'échantillons $N = 2^d$. Le nombre d'étages est $log_2(N)$ et chaque étage est constitué d'un commutateur, d'un ensemble de registres et d'un PE_FFT2. Le 1er étage est composé du commutateur C_1, du module de registres M1 formé de (N/2) registres et de PE1. Le $i^{ème}$ étage, i : 2 … $log_2(N)$, est formé du commutateur C_i, de 2 modules de registres Mi_1 et Mi_2, formés chacun de ($N/2^i$) registres, et de PEi.

Remarques importantes :

- Les différents $log_2(N)$ étages sont de structures différentes vis-à-vis des modules de registres et nécessitent donc des signaux de commande différents pour les commutateurs, d'où l'absence du facteur modularité recherché dans les architectures parallèles VLSI.
- Si plusieurs FFT2 sont à calculer sur des trames de données adjacentes les PEs ne sont actifs que pendant la moitié du temps. On peut voir cela en reprenant l'exemple donné en haut pour $N = 8$: PE_1, par exemple, est inactif pendant les 4 premières périodes, actif pendant les 4 prochaines (périodes 5 à 8) puis inactif pendant les 4 suivantes, donc attendre la période 13 pour redevenir actif si une nouvelle FFT2 est à calculer, et ainsi de suite. Ceci implique qu'entre 2 calculs consécutifs de FFT2, rien n'est récupéré à la sortie du réseau pendant 4 périodes (N/2 périodes pour le cas général). En considérant un débit en données important à l'entrée du réseau, un système de mémoires peut être inséré à l'entrée du réseau pour retenir les données et les fournir à ce dernier, à la cadence adéquate et dans l'ordre donné sur la figure II.17, où C_1 et M1 ont été enlevés.

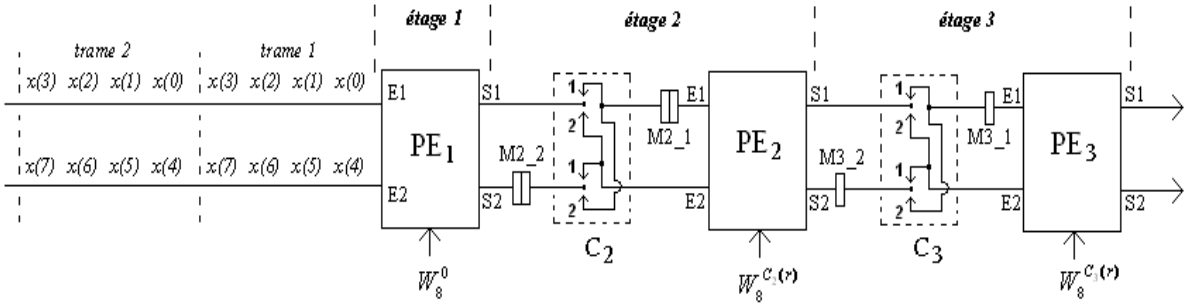


Figure II.17. Architecture pipelinée et linéaire de la FFT2 pour $N = 8$, à débit amélioré.

Les PEs seront alors tout le temps actifs et deux composantes spectrales sont obtenues à la sortie du réseau à chaque étape. Les tableaux suivants exposent les différentes étapes de calculs pour 3 trames consécutives traitées par le réseau.

Flux ininterrompu de données à l'entrée du réseau

Trame3				Trame 2				Trame 1				S_{PE1}
x(3)	x(2)	x(1)	x(0)	x(3)	x(2)	x(1)	x(0)	x(3)	x(2)	x(1)	x(0)	
x(7)	x(6)	x(5)	x(4)	x(7)	x(6)	x(5)	x(4)	x(7)	x(6)	x(5)	x(4)	

Etape 1 : 1ère trame (C2 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}
$X_1(0)$			$X_1(0)$		
$X_1(4)$	$X_1(4)$				

Etape 2 : (C2 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}
$X_1(1)$			$X_1(1)$	$X_1(0)$	
$X_1(5)$	$X_1(5)$	$X_1(4)$			

Etape 3 : (C2 en 2) (C3 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(2)$			$X_1(4)$	$X_1(1)$	$X_2(0)$		$X_2(0)$	
$X_1(6)$	$X_1(6)$	$X_1(5)$			$X_2(2)$	$X_2(2)$		

Etape 4 : (C2 en 2) (C3 en 2)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(3)$			$X_1(5)$	$X_1(4)$	$X_2(1)$		$X_2(2)$	$X_3(0)$
$X_1(7)$	$X_1(7)$	$X_1(6)$			$X_2(3)$	$X_2(3)$		$X_3(1)$

Etape 5 : 2ème trame (C2 en 1) (C3 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(0)$			$X_1(0)$	$X_1(5)$	$X_2(4)$		$X_2(4)$	$X_3(2)$
$X_1(4)$	$X_1(4)$	$X_1(7)$			$X_2(6)$	$X_2(6)$		$X_3(3)$

Etape 6 : (C2 en 1) (C3 en 2)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(1)$			$X_1(1)$	$X_1(0)$	$X_2(5)$		$X_2(6)$	$X_3(4)$
$X_1(5)$	$X_1(5)$	$X_1(4)$			$X_2(7)$	$X_2(7)$		$X_3(5)$

Etape 7 : (C2 en 2) (C3 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(2)$			$X_1(4)$	$X_1(1)$	$X_2(0)$		$X_2(0)$	$X_3(6)$
$X_1(6)$	$X_1(6)$	$X_1(5)$			$X_2(2)$	$X_2(2)$		$X_3(7)$

Etape 8 : (C2 en 2) (C3 en 2)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(3)$			$X_1(5)$	$X_1(4)$	$X_2(1)$		$X_2(2)$	$X_3(0)$
$X_1(7)$	$X_1(7)$	$X_1(6)$			$X_2(3)$	$X_2(3)$		$X_3(1)$

Etape 9 : 3ème trame (C2 en 1) (C3 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(0)$			$X_1(0)$	$X_1(5)$	$X_2(4)$		$X_2(4)$	$X_3(2)$
$X_1(4)$	$X_1(4)$	$X_1(7)$			$X_2(6)$	$X_2(6)$		$X_3(3)$

Etape 10 : (C2 en 1) (C3 en 2)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(1)$			$X_1(1)$	$X_1(0)$	$X_2(5)$		$X_2(6)$	$X_3(4)$
$X_1(5)$	$X_1(5)$	$X_1(4)$			$X_2(7)$	$X_2(7)$		$X_3(5)$

Etape 11 : (C2 en 2) (C3 en 1)

S_{PE1}	M2_2		M2_1		S_{PE2}	M3_2	M3_1	S_{PE3}
$X_1(2)$			$X_1(4)$	$X_1(1)$	$X_2(0)$		$X_2(0)$	$X_3(6)$
$X_1(6)$	$X_1(6)$	$X_1(5)$			$X_2(2)$	$X_2(2)$		$X_3(7)$

Tableaux II.2. Sorties des PEs et contenus des registres M*i* lors des différentes étapes de calcul pour l'architecture pipelinée et linéaire de la FFT2 pour $N = 8$, sur un flux continu de trames d'échantillons.

Le spectre de la 1ère trame est obtenu à la 7ème étape, celui de la 2ème à la 11ème étape, c'est-à-dire après 4 étapes, et ainsi de suite pour les spectres des trames suivantes.

Dans le cas général, le spectre de la 1ère trame est obtenu à l'étape *k*, telle que :

$$k = \frac{N}{2} + \sum_{i=2}^{\log_2(N)} \frac{N}{2^i} = \frac{N}{2} + \left(\frac{N}{2} - 1\right) = N - 1 .$$

$\frac{N}{2}$ représentent le nombre d'étapes nécessaires pour que la dernière colonne d'échantillons de la 1ère trame entre dans le premier PE_FFT2 du circuit et $\left(\sum_{i=2}^{\log_2(N)} \frac{N}{2^i}\right)$ est le nombre d'étapes requises pour traverser l'ensemble des registres et être introduite dans le dernier PE_FFT2 du circuit. Un spectre correspondant à une nouvelle trame d'échantillons est obtenu toutes les $\left(\frac{N}{2}\right)$ étapes suivantes, avec deux composantes spectrales obtenues à la sortie du réseau à chaque étape.

4.2. Implémentation de la FFT4 sur des architectures pipelinées linéaires

L'implémentation de la FFT4 sur des architectures pipelinées et linéaires, sujet aussi de plusieurs travaux [82, 86 – 88], peut être conçue selon un modèle similaire à celui adopté pour l'architecture pipelinée et linéaire de la FFT2. La figure II.18.a. montre une telle structure pour $N = 4^2 = 16$. La figure II.18.b. donne les modes de fonctionnement des commutateurs C_1 et C_2 pour les différentes étapes de calcul.

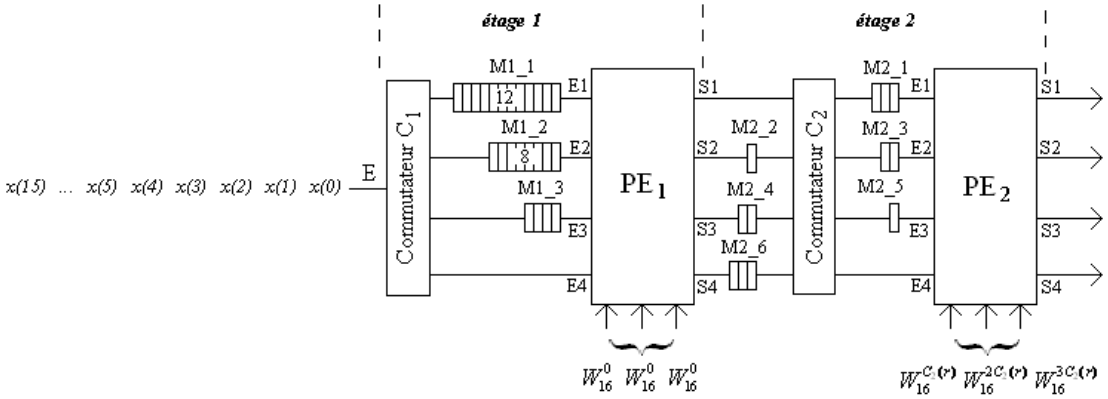


Figure II.18.a. Architecture pipelinée et linéaire de la FFT4 pour $N = 16$.

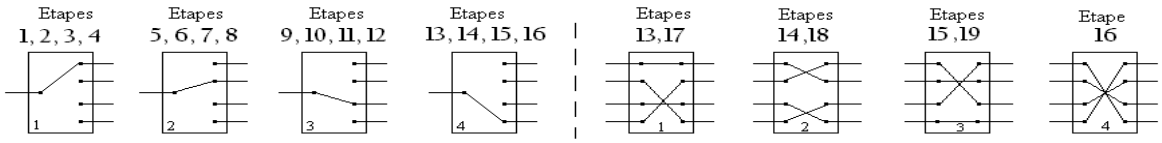


Figure II.18.b. Positions des commutateurs C_1 et C_2 lors des différentes étapes de calcul.

Le fonctionnement du circuit avec les différents passages et les états des commutateurs pendant chaque étape de calcul est donné sur les tableaux II.3.

Etape 4 (C1 en 1)

M1_1	x(3)	x(2)	x(1)	x(0)								
M1_2												
M1_3												

Etape 8 (C1 en 2)

M1_1					x(3)	x(2)	x(1)	x(0)				
M1_2					x(7)	x(6)	x(5)	x(4)				
M1_3												

Etape 12 (C1 en 3)

M1_1									x(3)	x(2)	x(1)	x(0)
M1_2									x(7)	x(6)	x(5)	x(4)
M1_3									x(11)	x(10)	x(9)	x(8)

Etape13 (C1 en 4) et (C2 en 1)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5			S_{PE2}
$X_1(0)$				$X_1(0)$			
$X_1(4)$			$X_1(4)$				
$X_1(8)$		$X_1(8)$					
$X_1(12)$	$X_1(12)$						

Etape14 (C1 en 4) et (C2 en 2)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5			S_{PE2}
$X_1(1)$				$X_1(4)$	$X_1(0)$		
$X_1(5)$			$X_1(5)$		$X_1(1)$		
$X_1(9)$		$X_1(9)$	$X_1(8)$				
$X_1(13)$	$X_1(13)$	$X_1(12)$					

Etape15 (C1 en 4) et (C2 en 3)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5			S_{PE2}
$X_1(2)$				$X_1(8)$	$X_1(4)$	$X_1(0)$	
$X_1(6)$			$X_1(6)$		$X_1(5)$	$X_1(1)$	
$X_1(10)$		$X_1(10)$	$X_1(9)$			$X_1(2)$	
$X_1(14)$	$X_1(14)$	$X_1(13)$	$X_1(12)$				

Etape16 (C1 en 4) et (C2 en 4)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5			S_{PE2}
$X_1(3)$				$X_1(12)$	$X_1(8)$	$X_1(4)$	$X_2(0)$
$X_1(7)$			$X_1(7)$		$X_1(9)$	$X_1(5)$	$X_2(1)$
$X_1(11)$		$X_1(11)$	$X_1(10)$			$X_1(6)$	$X_2(2)$
$X_1(15)$	$X_1(15)$	$X_1(14)$	$X_1(13)$				$X_2(3)$

Etape17 (C1 en 4) et (C2 en 1)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5		S_{PE2}
				$X_1(12)$	$X_1(8)$	$X_2(4)$
				$X_1(13)$	$X_1(9)$	$X_2(5)$
			$X_1(11)$		$X_1(10)$	$X_2(6)$
		$X_1(15)$	$X_1(14)$			$X_2(7)$

Etape18 (C1 en 4) et (C2 en 2)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5		S_{PE2}
					$X_1(12)$	$X_2(8)$
					$X_1(13)$	$X_2(9)$
					$X_1(14)$	$X_2(10)$
			$X_1(15)$			$X_2(11)$

Etape19 (C1 en 4) et (C2 en 3)

S_{PE1}	M2_2, M2_4, M2_6			M2_1, M2_3, M2_5		S_{PE2}
						$X_2(12)$
						$X_2(13)$
						$X_2(14)$
						$X_2(15)$

Tableaux II.3. Sorties des PEs et contenus des différents registres M*i* pendant les différentes étapes de calcul pour l'architecture pipelinée et linéaire de la FFT4 pour $N = 16$.

Le spectre est totalement calculé après 19 étapes. Les échantillons se présentent séquentiellement à l'entrée E du réseau dans l'ordre donné sur la figure II.18.a. Initialement à la position 1, le commutateur C_1, avec ses 4 états possibles, bascule vers un nouvel état toutes les 4 étapes. Le commutateur C_2, lui aussi à 4 états possibles, change vers un nouvel état à chaque étape et ne commence à travailler qu'à la 13ème étape. La figure II.18.b. donne les positions des commutateurs C_1 et C_2 pour chaque étape de calcul. Les registres M1_*i* (*i* = 1..3) et M2_*j* (*j* = 1..6) permettent de faire introduire les retards nécessaires pour synchroniser les 4 échantillons adéquats aux entrées des PEs appropriés au moment convenable. Les coefficients $W_{16}^{C_i(r)}, W_{16}^{2C_i(r)}$ et $W_{16}^{3C_i(r)}$ correspondant à chaque étape et à chaque PE_FFT4 sont fournis à celui-ci au moment qui convient.

Une généralisation de l'architecture peut être déduite pour $N = 4^d$. Le nombre de PE_FFT4 sera égal à $log_4(N)$. Chaque étage aura son ensemble de vecteurs de registres distribués de la manière suivante :

- Pour le 1er étage : M1_1 = 3.*N*/4 registres ; M1_2 = 2.*N*/4 registres et M1_3 = *N*/4 registres.
- Pour le *i*ème étage, avec *i* allant de 2 à *d* : M*i*_1 = M*i*_6 = 3.*N*/4^i registres ;

M*i*_3 = Mi_4 = 2.*N*/4^i registres et M*i*_2 = M*i*_5 = *N*/4^i registres.

Les commutateurs C_i, pour $i > 1$, auront tous la même structure.

A titre d'exemple, la figure II.19 donne l'architecture pipelinée et linéaire de la FFT4 pour *N* = 64, avec pour les différents vecteurs de registres:

M1_1 = 48 registres ; M1_2 = 32 registres ; M1_3 = 16 registres

M2_1 = M2_6 = 12 registres ; M2_3 = M2_4 = 8 registres ; M2_2 = M2_5 = 4 registres.

M3_1 = M3_6 = 3 registres ; M2_3 = M2_4 = 2 registres ; M2_2 = M2_5 = 1 registre.

Le commutateur C_1 changera d'état toutes les 16 étapes, C_2 toutes les 4 étapes et C_3 à chaque étape. Pour le cas général C_i basculera vers un nouvel état toutes les (N/4^i) étapes.

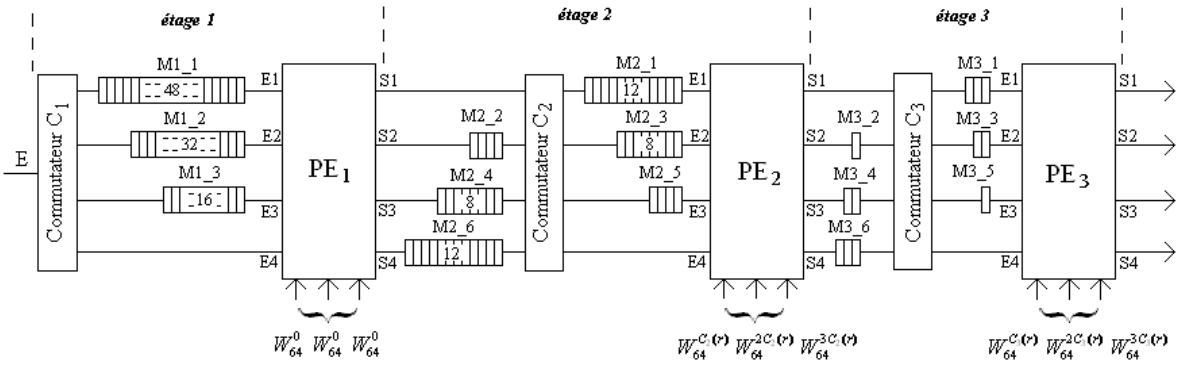


Figure II.19. Architecture pipelinée et linéaire de la FFT4 pour *N* = 64.

Remarques importantes : Les mêmes inconvénients soulevés pour l'architecture pipelinée et linéaire de la FFT2, sont valables pour celle de la FFT4 :

- Les étages sont à structures différentes vis-à-vis du nombre de registres par étage et les signaux de commande des commutateurs diffèrent d'un étage à l'autre, d'où l'absence du facteur modularité.
- Les PEs ne sont actifs que seulement le quart du temps. En effet on peut remarquer cela en reprenant l'exemple où N =16 et analyser les tableaux II.3. On remarque, par exemple, que PE_1 n'est actif que pendant les étapes 13 à 16 et doit attendre 12 nouvelles étapes pour travailler sur une nouvelle trame d'échantillons. Cela implique que pendant 12 étapes sur 16 rien n'est récupéré à la sortie du circuit. Si donc plusieurs FFT4 sont à calculer sur un tel circuit et qu'on insère un système de mémoires à son entrée pour retenir les et les lui fournir à la cadence adéquate suivant l'ordre donné sur la figure II.20. (C_1 et les M1_j, j :1..3, ont été éliminés), les PEs seront actifs tout le temps et le débit en données sera ainsi quadruplé. Quatre composantes seront disponibles à la sortie du réseau à chaque étape, et un spectre complet sera calculé toutes les 4 étapes.

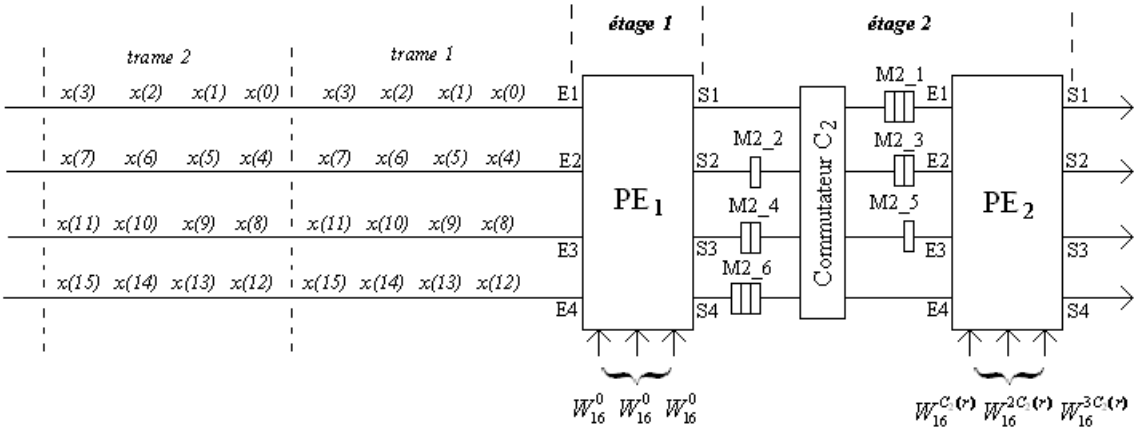


Figure II.20. Architecture pipelinée et linéaire de la FFT4 pour N =16, à débit amélioré.

Dans le cas général pour N échantillons, le spectre de la 1ère trame est obtenu à l'étape k, telle que :

$$k = \frac{N}{4} + 3.\sum_{i=2}^{\log_4(N)} \frac{N}{4^i} = \frac{N}{4} + \left(\frac{N}{4} - 1\right) = \frac{N}{2} - 1$$

où $\frac{N}{4}$ représente le nombre d'étapes nécessaires pour que la dernière colonne de la 1ère trame entre dans le premier PE_FFT4 du circuit et $\left(3.\sum_{i=2}^{\log_4(N)}\frac{N}{4^i}\right)$ est le nombre d'étapes requises pour traverser l'ensemble des registres et être introduite dans le dernier PE_FFT4 du circuit. Les spectres des prochaines trames sont obtenus toutes les $\left(\frac{N}{4}\right)$ étapes suivantes, avec 4 composantes spectrales obtenues à la sortie du réseau à chaque étape.

5. Comparaison entre les différentes architectures FFT étudiées

Le tableau suivant résume les caractéristiques des différentes architectures proposées.

	Nombre de processeurs	Nombre de multiplieurs réels	Nombre d'additionneurs réels	Nombre d'étapes pour le calcul du 1er spectre	Nombre d'étapes pour la sortie du spectre suivant
FFT2 étalée	$(N/2).log_2(N)$	$(2N).log_2(N)$	$(3N).log_2(N)$	$log_2(N)$	1
FFT4 étalée	$(N/8).log_2(N)$	$(3/2).N.log_2(N)$	$(11/4).N.log_2(N)$	$(1/2).log_2(N)$	1
FFT2 rebouclée	$N/2$	$2N$	$3N$	$log_2(N)$	$log_2(N)$
FFT4 rebouclée	$N/4$	$3N$	$(11/2).N$	$(1/2).log_2(N)$	$(1/2).log_2(N)$
FFT2 linéaire (sans C_1 ni M1)	$log_2(N)$	$4.log_2(N)$	$6.log_2(N)$	N-1	$N/2$
FFT4 linéaire (sans C_1 ni M1)	$(1/2).log_2(N)$	$6.log_2(N)$	$11.log_2(N)$	$(N/2)$ - 1	$N/4$

Tableau II.4. Comparaison entre les différentes architectures FFT proposées.

Un exemple concret de N=1024 permet de mieux situer les écarts entre les différentes architectures. Ceci est montré sur le tableau suivant qui reprend les formules du tableau II.4.

	Nombre de processeurs	Nombre de multiplieurs réels	Nombre d'additionneurs réels	Nombre d'étapes pour le calcul du 1er spectre	Nombre d'étapes pour la sortie du spectre suivant
FFT2 étalée	5120	20480	30720	10	1
FFT4 étalée	1280	15360	28160	5	1
FFT2 rebouclée	512	2048	3072	10	10
FFT4 rebouclée	256	3072	5632	5	5
FFT2 linéaire (sans C_1 ni M1)	10	40	60	1023	512
FFT4 linéaire (sans C_1 ni M1)	5	60	110	511	256

Tableau II.5. Comparaison entre les différentes architectures FFT proposées pour N=1024.

Même si une étape FFT2 est légèrement inférieure en temps d'exécution à celle d'une FFT4 ($TR\_PE\_FFT2 = t_m + 2.t_a$ inferieur à $TR\_PE\_FFT4 = t_m + 3.t_a$), on peut conclure d'après le tableau II.4 que :

- Pour un choix d'une certaine architecture (étalée, rebouclée ou linéaire) celle utilisant la base 4 est meilleure que celle utilisant la base 2 sous tous les aspects possibles (temps de réponse et surface implantable), même si le PE_FFT4 présente une complexité de mise en œuvre plus importante par rapport au PE_FFT2.
- Pour la même base utilisée (2 ou 4), plusieurs considérations devraient être prises en compte pour choisir l'architecture à implémenter : la faisabilité, le coût, les contraintes auxquelles est soumise l'application, etc… La FFT étalée est celle qui permet le meilleur temps de réponse surtout si des calculs devaient se faire sur des trames ininterrompues d'échantillons. Mais vu le nombre de PEs qui la composent, elle pose problème vis-à-vis de son implantation VLSI pour des nombres d'échantillons N élevés. Ce problème pourrait être résolu en optant pour l'architecture rebouclée au détriment du temps de réponse qui est rallongé, surtout pour les trames d'échantillons ultérieures à la première. Un gain considérable en surface silicium est apporté par l'architecture linéaire par rapport aux deux premières architectures, mais ceci est fait au détriment du temps de réponse qui sera encore plus rallongé.

6. Conclusion

Dans ce chapitre, plusieurs structures parallèles ont été proposées pour le calcul de la FFT. Les différences et les préférences entre ces architectures ont été largement discutées. Les facteurs puissance de calcul, modularité, régularité et localité des structures ont été recherchés et ont été atteints à différents niveaux qui dépendent de l'architecture choisie. Le facteur modularité, par exemple, est absent de la structure linéaire, mais le facteur localité, ou interconnexion locale entre étages, y est présent mais avec un certain nombre de registres intercalés entre les différents PEs. Pour les architectures FFT étalée et rebouclée, les facteurs puissance de calcul, modularité et régularité existent, mais le facteur localité, primordial pour une implantation VLSI, est manquant. Ceci est dû à la structure même des algorithmes FFT. Pour l'obtenir, il faudrait revenir à l'équation initiale qui définit la TFD et l'implémenter sur des structures spéciales nommés Réseaux Systoliques. C'est dans ce contexte que ces derniers seront proposés dans le prochain chapitre et où les facteurs modularité, régularité et localité, tant recherchés dans les structures parallèles et synonymes de puissance de calcul et possibilité d'implantation VLSI seront atteints.

Chapitre III

Implémentation de la TFD sur des réseaux systoliques

1. Introduction

Dans le chapitre précédent nous avons présenté et étudié plusieurs architectures parallèles performantes pour le calcul de la TFR mais auxquelles manquait le facteur primordial qui est la localité des interconnexions entre les différents PEs. Ce problème, comme nous l'avons signalé dans la conclusion du chapitre précédent, peut être efficacement résolu par l'utilisation d'une classe spéciale d'architectures parallèles qui sont les réseaux systoliques.

Les réseaux systoliques, qui sont des réseaux de processeurs organisés suivant une structure régulière et qui sont qualifiés pour des applications invoquant un nombre important de calculs, sont la forme de parallélisme qui sera étudiée dans la suite de ce chapitre. Tout d'abord nous ferons une présentation des réseaux systoliques, puis nous aborderons leur besoin dans les applications relatives au domaine du traitement du signal, ensuite nous décrirons leurs structures et leurs caractéristiques. Nous finirons par des exemples d'application qui nous permettront de proposer des réseaux systoliques performants pour l'implémentation de la transformée de Fourier discrète dans ce même chapitre, et d'autres réseaux pour l'implémentation des filtres numériques récursifs RII dans le chapitre suivant.

2. Présentation des réseaux systoliques

De nombreux algorithmes travaillant sur un grand nombre de données et présentant des formes simplement récursives sont par leurs aspects évalués séquentiellement. Par contre, d'autres algorithmes exhibant un certain degré de parallélisme plus ou moins important peuvent exploiter celui-ci pour réduire considérablement le temps de calcul. Ces algorithmes sont alors implémentés sur des architectures parallèles. Pour une architecture computationnelle massivement parallèle, deux propriétés doivent être retenues : une flexibilité vis-à-vis des applications et une efficacité de calcul satisfaisante [89]. Cela veut dire que cette architecture doit être reconfigurable pour exploiter les parallélismes dépendant de l'application, programmable en langage de haut niveau pour un contrôle pratique des tâches, extensible facilement à de nombreuses applications et capable de supporter des organisations SIMD (Single Instruction Multiple Data : Exécution d'une instruction unique sur un flux de données multiples) et des organisations MIMD (Multiple Instruction Multiple Data : Exécution de plusieurs instructions sur un flux de données multiples) pour pouvoir exploiter les exigences du parallélisme non homogène. Les machines SIMD et MIMD sont d'ailleurs bien expliquées

dans la taxonomie de Flynn qui donne une classification détaillée des différents types de machines possibles [90].

Les réseaux systoliques sont des architectures parallèles très performantes et sont idéalement qualifiés pour des applications intensives en calcul qui exploitent efficacement le parallélisme massif. Classés dans une zone entre les ordinateurs vectoriels (très adaptés pour le traitement des tableaux) et les ordinateurs massivement parallèles (calculateurs qui possèdent un très grand nombre de processeurs), les réseaux systoliques combinent idéalement des communications locales intensives et un calcul avec un parallélisme décentralisé dans un ensemble condensé. Ils trouvent leur grand intérêt quand ils sont utilisés dans des processus réguliers, modulaires, rythmiques, synchrones, concurrents et nécessitant un calcul intensif et répétitif [89, 91].

Alors que les réseaux systoliques étaient à l'origine utilisés pour des architectures fixes ou spéciales, le concept systolique a été étendu ces dernières années aux architectures SIMD et MIMD générales.

Il faut noter que la prolifération des réseaux systoliques n'a été possible que grâce à l'avancée technologique et économique (c.à.d. diminution des prix) des circuits VLSI durant pratiquement les quatre décennies passées. Cette avancée technologique s'est traduite essentiellement par l'augmentation d'une façon draconienne de la densité des portes logiques au niveau des circuits intégrés qui a permis de fabriquer des processeurs plus complexes en fonctions, plus performants, et ainsi encouragé et vulgarisé le parallélisme.

3. Le besoin des réseaux systoliques dans le domaine du traitement du signal

Les applications en traitement de signal sont le plus souvent des applications en temps réel avec un nombre de données élevé à traiter. Elles présentent par conséquent un calcul intensif à effectuer. Cela requiert des calculateurs très performants et très développés, présentant ou ayant des systèmes multiprocesseurs et utilisant des techniques de programmation et de calcul parallèle. Cependant, de part leur nature d'usage général, ces supercalculateurs ne sont pas adaptés à toutes les applications et surtout à celles qui exigent un débit en données élevé. De ce fait, les chercheurs ont été incités à axer leurs recherches sur le développement de systèmes processeurs dédiés. Ces derniers se présentent sous deux formes. L'une de ces formes représente la catégorie des processeurs inflexibles et exclusivement dédiés aux applications qui leur sont spécifiées. On peut citer par exemples les processeurs dédiés et

câblés qui ont une vitesse de traitement très élevée, et donc adaptés pour le traitement du signal en temps réel. Leur inconvénient majeur vient du fait de leur durée de développement excessive et donc de leur coût élevé. L'autre forme regroupe la catégorie de processeurs autorisant une certaine flexibilité vis-à-vis de la programmabilité et de la reconfigurabilité [89, 92 – 94].

L'avancée fulgurante de la technologie VLSI a permis la faisabilité et la concrétisation d'un grand nombre d'algorithmes sur des circuits VLSI dédiés ou bien reconfigurables. Lors de la conception de systèmes basés sur des processeurs dédiés ou reconfigurables les chercheurs ont essayé et essayent d'ailleurs toujours d'innover en matière d'algorithmes et d'architectures [28, 29, 95 – 104]. Deux aspects importants, qui dépendent en grande partie des contraintes technologiques, doivent néanmoins être pris en considération : la communication, fonction de l'interconnexion entre processeurs, et la dépendance des flots de données. Ces deux éléments font que pour un algorithme récursif, dont un bon nombre d'algorithmes en traitement de signal, une grande importance soit donnée à la localité de celui-ci lors de son implantation VLSI. Cela est réalisé par l'utilisation de chemins courts pour la communication de données, tout en essayant de répartir équitablement la charge des calculs entre tous les processeurs. Ces considérations liées à la technologie VLSI ont conduit à la conception des réseaux systoliques. C'est d'ailleurs dans ce contexte qu'en 1978 H.T. Kung et C.E. Leiserson proposèrent, à l'université Carnegie-Melon, un modèle de système parallèle qu'ils appelèrent "Réseau systolique pour VLSI" et qui fera l'objet en 1980 d'une publication intitulée "Systolic Arrays (for VLSI)" [32]. Avec sa deuxième publication "Why Systolic Architectures?" [105], en 1982, H.T. Kung, illustra le besoin de tels réseaux pour les applications sollicitant des calculs volumineux et le gain en performance à acquérir en les intégrant dans un système hôte. Le terme 'systolique' dans l'appellation de ces réseaux n'a pas été improvisé. En effet, en physiologie, le terme systolique décrit les contractions (systoles) du cœur, qui d'une façon régulière et rythmique propulse le sang dans toutes les cellules du corps humain à travers artères, veines et capillaires. Par analogie, un réseau calculateur systolique propulse le flot de données à traiter et exécute les opérations ou calculs d'une manière rythmée, régulière et répétitive. Le débit du flot de données à traiter est conditionné par les opérations d'entrée/sortie du réseau calculateur, de la même manière que le cœur contrôle la quantité du sang propulsé vers les cellules puisqu'il est la source et la destination de tout le sang.

4. Définition d'un réseau systolique

Bien qu'il n'y ait pas une définition standard pour décrire exactement un réseau systolique, on peut partir de la définition suivante :

Un réseau systolique est un système composé d'un ensemble de cellules processeurs, appelées aussi processeurs élémentaires (PE), interconnectées localement suivant une architecture régulière et homogène, chacune capable d'effectuer une certaine opération sur les données présentes à son ou ses entrées et de transmettre éventuellement ces mêmes données, les résultats ou les deux en même temps, aux cellules contigües un cycle de temps plus tard. La transmission des données se fait d'une manière pipeline, rythmique et synchrone à travers tout le réseau. Les communications entre le réseau et l'extérieur, c'est à dire une mémoire externe ou un ordinateur hôte, se fait par l'intermédiaire des cellules situées aux frontières du réseau.

5. Organisation d'un réseau systolique

Bien qu'il y ait une multitude de topologies de réseaux systoliques, nous présentons sur les figures III.1.a et III.1.b un exemple d'organisation générale d'un réseau systolique planaire à connexions orthogonales et une structure possible d'une cellule de base. Chaque ligne/colonne représente un bus de communication et chaque intersection une cellule processeur. Les registres horizontal et vertical permettent de mémoriser les entrées correspondantes pour un traitement au niveau de l'unité de calcul. Le registre accumulateur sert à retenir les résultats intermédiaires pendant les différentes étapes de calcul et le résultat final après la dernière étape.

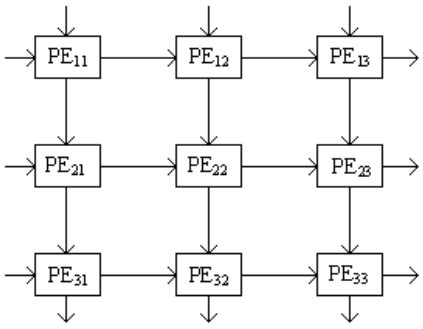


Figure III.1.a. Réseau systolique planaire à connexion orthogonales.

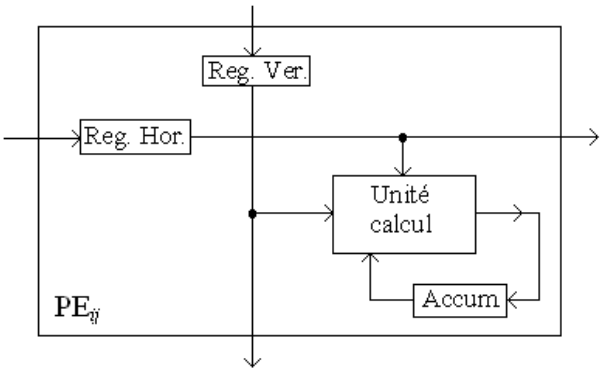


Figure III.1.b. Cellule élémentaire.

Toutefois, il faut remarquer que les réseaux systoliques sont plus que des matrices de processeurs qui exécutent des algorithmes systoliques. Les réseaux systoliques sont constitués d'éléments qui prennent l'une des formes suivantes:

- une cellule spéciale avec des fonctions câblées,
- une cellule processeur avec une unité de décodage d'instructions et une unité de traitement,
- un processeur complet avec une unité de contrôle et une unité de traitement.

Dans tous les cas, les éléments ou cellules systoliques sont caractérisés pour des communications locales intensives et un parallélisme décentralisé. Parce qu'un réseau systolique se compose de cellules d'un seul ou, de seulement quelques types, il a des caractéristiques régulières et simples. Le réseau systolique est généralement extensible avec un minimum de difficulté.

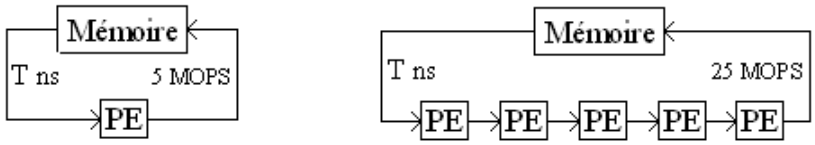
6. Propriétés caractéristiques des réseaux systoliques

Les réseaux systoliques présentent un certain nombre de propriétés spécifiques et très intéressantes qui les rendent très attrayants pour la réalisation de structures de calcul à très grande échelle d'intégration (VLSI). Parmi ces propriétés, on peut citer les plus importantes [29, 89, 106, 107] :

- La régularité qui veut dire l'utilisation de cellules simples et uniformes. En effet, les structures systoliques sont formées d'un seul type ou, au pire des cas, d'un nombre réduit de types de cellules processeurs. Cela induit une simplicité et une régularité au niveau de leur contrôle et permet surtout l'extensibilité modulaire et l'adaptation du réseau à la taille du problème à traiter. Pour le cas de la TFD ou des filtres numériques RII, le réseau ne sera pas limité par le nombre d'échantillons N et pourra en conséquence être adapté à celui-ci.
- La localité ou l'interconnexion locale entre les cellules qui permet, d'une part d'intensives communications locales entres processeurs, et d'autre part la réduction des surfaces consacrées au routage dans le dessin de masque du circuit VLSI. Ceci induira une augmentation significative de la densité d'intégration [106, 108, 109]. Cette propriété de localité serait pleinement exploitée par un nombre important d'algorithmes de traitement de signal localement récursifs. Un algorithme est dit récursif s'il effectue d'une façon répétitive la même séquence d'opérations sur des données disponibles de manière séquentielle à son entrée. Dans un algorithme récursif parallèle, localement communicatif, les données d'entrée et de sortie sont exprimées en fonction du temps et de l'espace. Leurs

indices de temps et d'espace sont reliés selon une forme régulière : au niveau d'une cellule, l'indice de temps des sorties est égal à celui des entrées incrémenté de un (Entrée(t), Sortie(t+1)), et les indices d'espace des données à l'entrée, impliquées dans une même opération récursive, sont séparés d'une quantité inférieure à une certaine limite donnée.

- La puissance de calcul : Un débit de données à traiter plus élevé est obtenu sans pour autant augmenter la bande passante de la mémoire. Ce principe est schématisé par les circuits donnés en exemple sur la figure III.2.



(a) Processeur séquentiel. (b) Réseau systolique.

Figure III.2. Principe en termes de puissance de calcul d'un réseau systolique.

Une fois que la donnée est injectée au réseau à partir de la mémoire, elle est concrètement utilisée au niveau de chaque cellule en étant 'pompée' d'une cellule à l'autre le long du réseau. Sur la figure III.2.a, le processeur séquentiel effectue par exemple au plus 5 millions d'opérations par seconde (MOPS), limité pour cela essentiellement par le temps d'accès à la mémoire égal à T ns. Sur la figure III.2.b, le réseau systolique, composé de 5 processeurs identiques et connectés à la même mémoire autoriserait 25 MOPS. Les structures systoliques sont très efficaces lorsque la quantité d'opérations à réaliser est supérieure d'au moins un ordre de grandeur au volume de données en entrée [29].

7. L'importante progression des réseaux systoliques

Trois facteurs essentiels ont contribué à la fabuleuse progression des réseaux systoliques pour les mettre sur l'avant scène des systèmes pouvant résoudre des applications scientifiques qui font appel à un nombre important de données et de calculs [89] : l'avancée technologique VLSI, le traitement concurrent et la demande de plus en plus grandissante des applications scientifiques, dont celles du traitement du signal, en termes de volume de données à traiter.

a) L'avancée technologique VLSI

Cette avancée a permis, grâce à deux facteurs importants, à donner un envol considérable pour le développement des réseaux systoliques. Premièrement, le degré d'intégration des circuits VLSI, de plus en plus accentué, permet de fabriquer des processeurs avec plus de portes logiques, donc avec des fonctions plus compliquées pouvant travailler sur des données ou mots plus larges en termes de nombre de bits. Plus grande densité implique

aussi que les données ont des chemins plus courts à parcourir et par conséquent les communications sur le circuit sont beaucoup plus rapides. Un deuxième facteur qu'on peut considérer comme primordial est le facteur économique. En effet les processus modernes de conception et de fabrication permettent la réalisation de circuits systoliques moins chers même en quantité réduite. Des outils CAO de synthèse et de simulation avancés permettent une conception plus efficace. Une cellule systolique peut ainsi être complètement simulée avant la fabrication, réduisant ainsi le risque de mal fonctionnement après sa réalisation. Avec les techniques de simulation très avancées de nos jours, des cellules peuvent être complètement testées, facilement dupliquées et arrangées pour former d'une manière modulaire et régulière un réseau systolique. Plus les circuits VLSI deviennent complexes dans leurs fonctions et donc dans leur conception, plus évident devient le besoin de les 'systoliser'. En effet toutes les précautions prises vis-à-vis de la tolérance aux défauts au niveau d'une cellule sont extensibles à toutes les autres cellules.

b) Le traitement concurrent

Beaucoup d'efforts ont été dépensés dans les architectures conventionnelles Von Neumann pour rendre leurs traitements de données plus concurrents. Cela a été possible par l'ajout de coprocesseurs, le développement d'unités multiprocesseurs, ou le pipelining aussi bien des données que des instructions. Les réseaux systoliques, eux, par une architecture massivement parallèle, peuvent combiner toutes ces qualités en regroupant un nombre important de processeurs homogènes suivant une structure en pipeline à *n* dimensions. Bien que le pipelining des données diminue les appels d'entrée/sortie (les mêmes données sont utilisées par les cellules adjacentes), la nouveauté apportée par les réseaux systoliques est le pipelining dans le traitement de ces données. En effet, et suivant le format de l'algorithme spécifié, chaque cellule dans la chaîne participe au calcul d'une partie du résultat global qui sera, à la fin du processus, interprété par le processeur hôte.

c) L'exigence des applications scientifiques

Durant les trois dernières décennies, le développement de plus en plus croissant des techniques de calcul, par l'émergence de calculateurs très puissants, a permis d'aborder les applications qui présentent d'énormes volumes de données à traiter. En traitement de signal, la transformée de Fourier rapide, le filtrage, la convolution, la corrélation et bien d'autres sont des exemples concrets de ce type d'applications. Elles invoquent très souvent le produit matriciel dans leurs calculs. En analysant ces applications, on remarque qu'elles présentent un traitement

parallèle, massif et régulier, avec un haut débit en données et où l'utilisation des réseaux systoliques est tout à fait indiquée. Dans ce contexte, une multitude d'architectures systoliques ont d'ailleurs été proposées, où le calcul matriciel y est largement traité [29, 33, 110 – 112].

8. Considérations de mise en œuvre

Lors de la conception d'un réseau systolique, les concepteurs doivent tenir compte de plusieurs points importants dont :

a) Considérations algorithmiques et architecturales

Les concepteurs doivent être très familiers avec les algorithmes qu'ils doivent implémenter sur les réseaux systoliques, car concevoir un réseau systolique d'une manière heuristique comme ça a été souvent le cas dans le passé, est lent et induit à des erreurs. Ainsi, automatiser la synthèse des réseaux systoliques et pouvoir les simuler par des outils CAO est plus que nécessaire. En effet, L'automatisation garantit à la fois un système sûr et un temps de conception court. C'est d'ailleurs un important domaine de recherche et beaucoup de travaux ont été consacrés à ce sujet [113 – 120].

b) Intégration dans des systèmes existants

Généralement un réseau systolique est intégré dans un système hôte pour le complémenter comme système de calcul. Cette intégration pose des fois un problème vis-à-vis du haut débit en données à traiter par le réseau par rapport à la largeur de la bande réduite de la chaîne entrées/sorties de celui-ci. Un système de mémoires est donc très souvent incorporé entre le processeur hôte et le réseau systolique pour permettre l'accès aux données et leurs éventuels multiplexage et démultiplexage. Les figures III.3. et III.4. donnent respectivement les structures des machines SIMD et MIMD intégrant chacune un réseau systolique linéaire [89]. Le réseau systolique linéaire peut être remplacé par un réseau systolique planaire ou à autre topologie.

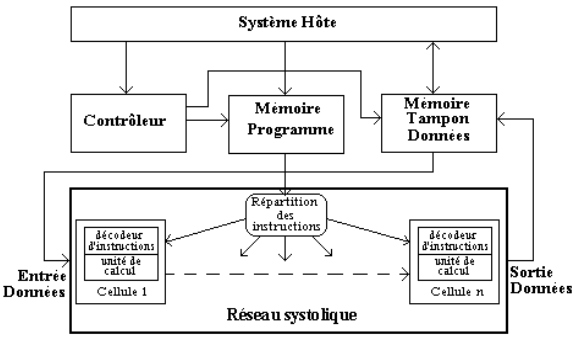


Figure III.3. Organisation générale d'une machine SIMD

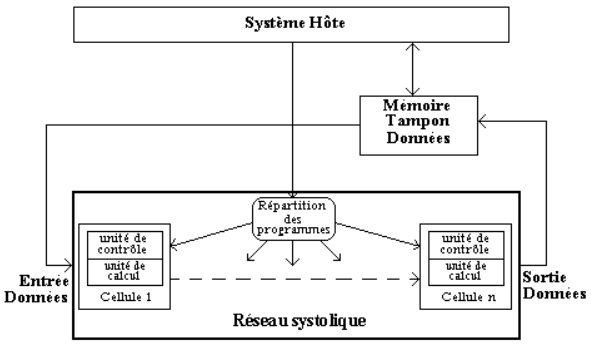


Figure III.4. Organisation générale d'une machine MIMD

programmable comportant un réseau systolique linéaire programmable comportant un réseau systolique linéaire.

c) Extensibilité

Etant donné que les réseaux systoliques sont formés de blocks ou modules de cellules élémentaires, ces dernières doivent permettre, par leur conception, un certain degré de flexibilité pour permettre leur extension, leur utilisation dans un large éventail de topologies et leur implantation sur différents substrats.

d) Synchronisation de l'horloge

La différence dans la longueur des lignes du bus horloge aussi bien à l'intérieur qu'à l'extérieur du réseau peut influer sur la synchronisation globale du système. Une attention particulière doit donc être donnée à la synchronisation des exécutions de l'ensemble des processeurs pour éviter un risque d'incohérence des données. Ce problème est absent pour les réseaux à front d'onde qui utilisent des communications asynchrones entre cellules mais qui, en contre partie, présentent une topologie beaucoup plus complexe [27, 121].

e) Fiabilité

Par de simples lois de probabilité on peut montrer que plus le nombre de processeurs augmente dans un réseau et moins fiable serait celui-ci. Afin d'assurer un bon rendement lors du processus de fabrication, les concepteurs doivent donc tenir compte d'un facteur important : la tolérance aux défauts. Ils doivent ainsi inclure dans leur processus de conception des procédures de testabilité afin d'assurer le bon fonctionnement du système. Pour qu'un réseau systolique soit tolérant aux défauts, une certaine redondance ou répétition des processeurs élémentaires doit être prévue ainsi que des mécanismes de reconfigurabilité [122 – 127].

La fabrication de tels réseaux systoliques implique donc les étapes suivantes :

- La détection des erreurs causées par les cellules défectueuses ;
- La localisation des cellules défectueuses ;
- La reconfiguration du réseau défectueux en un réseau valide en contournant les cellules défectueuses en les remplaçant par d'autres cellules qui sont en double.
- Faire les calculs dans le réseau reconfiguré.

9. Exemples d'algorithmes systoliques appliqués au calcul de la TFD

Une multitude d'algorithmes systoliques traitant divers problèmes existent. Les quelques exemples suivants illustrent parfaitement la notion d'architecture systolique. Ces exemples ne sont pas donnés ici fortuitement mais comme indiqué dans l'introduction ils seront utilisés et adaptés à l'implémentation de la TFD dans la suite du présent chapitre et à

l'implémentation des filtres récursifs RII dans le chapitre suivant. Les réseaux présentés ci-dessous sont à topologies linéaire et planaire orthogonale vu que leur implantation sur silicium est plus appropriée que celle des réseaux à autres topologies, par exemple la topologie 3D.

9.1. Evaluation d'un polynôme suivant le schéma de Horner sur un réseau systolique linéaire

L'évaluation d'un polynôme de degré *d*, d'une manière récursive, peut se faire suivant le schéma de Horner [128]. Plusieurs architectures systoliques pour l'implémentation du schéma de Horner sont présentées dans [129 – 131]. Dans ce qui suit nous présentons une implémentation sur un réseau systolique linéaire qui s'avère très performante si une série de polynômes $\{\boldsymbol{y_i} \;\; i : 1...k\}$ sont évalués sur une suite de données continues $\{\boldsymbol{x_i} \;\; i : 1...k\}$.

La méthode de calcul adoptée suivant le schéma de Horner utilise la factorisation suivante :

$$y_i = \sum_{j=0}^{n} a_j x_i^j = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \ldots + a_n x_i^n = a_0 + x_i(a_1 + x_i(a_2 + x_i(a_3 + \ldots + x_i a_n))) \quad \text{(III.1)}$$

Par exemple, pour un polynôme de degré d=3, y_i est donné par :

$$y_i = \sum_{j=0}^{3} a_j x_i^j = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 = a_0 + x_i(a_1 + x_i(a_2 + x_i a_3)) \quad \text{(III.2)}$$

Les calculs étant faits séquentiellement à partir des parenthèses les plus internes, on remarque que seulement une opération MAC (Multiplication-Accumulation: $r = r+a*x$), est effectuée à chaque étape de calcul. En notant f(i,j) le facteur de $\boldsymbol{x_i}$, l'évaluation du polynôme peut donc s'exprimer par le système d'équations récurrentes suivantes :

$\mathrm{f}(i,0) = a_n$

$\mathrm{f}(i,j) = \mathrm{f}(i,j\text{-}1) * x_i + a_{n-j} \quad \{ j : 1..n \}$

$y_i = \mathrm{f}(i,n)$

Le réseau systolique linéaire de la figure III.5.a, utilisant des cellules dont le mode de fonctionnement est donné sur la figure III.5.b, a une structure linéaire en pipeline. Il illustre une mise en œuvre de cet algorithme pour des polynômes de degré n=3. Un coefficient du polynôme est attribué à chaque cellule suivant l'ordre indiqué. Les valeurs $\boldsymbol{x_i}$ entrent une à une par la gauche du réseau et se propagent, après chaque étape de calcul, vers la droite en même temps que les résultats partiels.

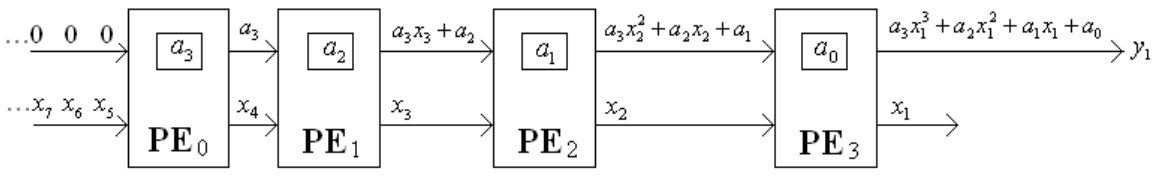
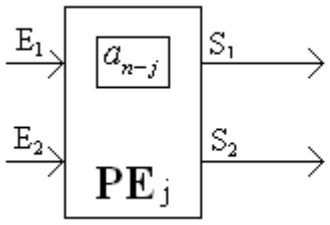


Figure III.5.a. Réseau systolique linéaire pour l'évaluation d'un polynôme, de degré d=3, suivant le schéma de Horner. (Etats à l'étape 4).



$$\begin{cases} S_1(t+1) \leftarrow E_1(t) * E_2(t) + a_{n-j} \\ S_2(t+1) \leftarrow E_2(t) \end{cases}$$

Figure III.5.b. Représentation et fonctionnement d'une cellule élémentaire pour le schéma de Horner

Durant l'étape (i+j), la cellule j reçoit la donnée $\boldsymbol{x_i}$ et la valeur calculée f(i,j-1) de sa voisine de gauche, effectue le calcul { f(i,j-1) * x_i + a_{n-j} } et transmet à son tour, lors de l'étape suivante, à sa voisine de droite, le résultat du calcul ainsi que la valeur $\boldsymbol{x_i}$. Le résultat final du calcul de $\boldsymbol{y_i}$ est fourni à l'étape (i+n) par la dernière cellule n.

Pour une parfaite modularité du réseau, les cellules frontières sont réalisées identiques aux cellules internes. Dans ce contexte, PE_0 reçoit sur son entrée E_1 un zéro et la sortie S_2 de PEn est ignorée pendant toutes les étapes de calcul.

Durant une étape, chaque PE ne participe qu'à un calcul partiel du calcul global du polynôme. Ce même calcul se répète à l'étape suivante sur une nouvelle entrée, et ainsi de suite.

Le régime de calcul recherché est obtenu après le temps de latence correspondant à la propagation de la première donnée x_1 à travers tout le réseau ((1+n) étapes). Le réseau délivrera à partir de ce moment un nouveau résultat y_i à chaque étape ou cycle systolique.

Le tableau III.1. montre l'évolution des calculs au niveau de chaque PE du réseau, pour une trame de quatre données x_i successives.

Etape	PE_0	PE_1	PE_2	PE_3
1	$(S_1)_0 \leftarrow (0*x_1)+a_3 = a_3$			
2	$(S_1)_0 \leftarrow (0*x_2)+a_3 = a_3$	$(S_1)_1 \leftarrow ((E_1)_1*x_1)+a_2 = a_3*x_1+a_2$		
3	$(S_1)_0 \leftarrow (0*x_3)+a_3 = a_3$	$(S_1)_1 \leftarrow ((E_1)_1*x_2)+a_2 = a_3*x_2+a_2$	$(S_1)_2 \leftarrow ((E_1)_2*x_1)+a_1 = a_3*x_1^2+a_2*x_1+a_1$	
4	$(S_1)_0 \leftarrow (0*x_4)+a_3 = a_3$	$(S_1)_1 \leftarrow ((E_1)_1*x_3)+a_2 = a_3*x_3+a_2$	$(S_1)_2 \leftarrow ((E_1)_2*x_2)+a_1 = a_3*x_2^2+a_2*x_2+a_1$	$((S_1)_3 \leftarrow ((E_1)_3*x_1)+a_0 = a_3*x_1^3+a_2*x_1^2+a_1*x_1+a_0 = y_1$
5		$(S_1)_1 \leftarrow ((E_1)_1*x_4)+a_2 = a_3*x_4+a_2$	$(S_1)_2 \leftarrow ((E_1)_2*x_3)+a_1 = a_3*x_3^2+a_2*x_3+a_1$	$((S_1)_3 \leftarrow ((E_1)_3*x_2)+a_0 = a_3*x_2^3+a_2*x_2^2+a_1*x_2+a_0 = y_2$
6			$(S_1)_2 \leftarrow ((E_1)_2*x_4)+a_1 = a_3*x_4^2+a_2*x_4+a_1$	$((S_1)_3 \leftarrow ((E_1)_3*x_3)+a_0 = a_3*x_3^3+a_2*x_3^2+a_1*x_3+a_0 = y_3$
7				$((S_1)_3 \leftarrow ((E_1)_3*x_4)+a_0 = a_3*x_4^3+a_2*x_4^2+a_1*x_4+a_0 = y_4$

Tableau III.1. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire pour l'évaluation d'un polynôme suivant le schéma de Horner.

9.2. Implémentation de la TFD suivant le schéma de Horner sur un réseau systolique linéaire

Les différentes composantes $X(i)$ de la TFD sont en réalité des polynômes de degré d=N-1 en W_N^k, N étant le nombre d'échantillons. En effet, cela est montré par l'équation définissant la TFD et réécrite ici :

$$X(k) = \sum_{n=0}^{N-1} x(n).W_N^{kn} = x(0).W_N^{0.k} + x(1).W_N^k + x(2).W_N^{2k} + \cdots + x(N-1).W_N^{k(N-1)} \qquad \text{(III.3)}$$

pour k : 0..N-1.

Maintenant, en posant $a = W_N^k$ et en sachant que $a^0 = a^N$ (propriété de la périodicité : $W_N^0 = W_N^N = 1$), la relation III.3 peut se mettre sous la forme :

$$X(k) = x(1).a + x(2).a^2 + \cdots + x(N-1).a^{(N-1)} + x(0).a^N \qquad \text{(III.4)}$$

Cette relation a la forme d'un polynôme en a de degré N qui peut alors être évalué suivant le schéma de Horner et implémenté sur le réseau systolique linéaire de la figure III.5.a et qui a d'ailleurs été à la base de plusieurs implémentations de la TFD [72, 132 – 136].

L'utilisation du schéma de Horner impose la factorisation de l'équation (III.4) suivant la forme suivante :

$$X(k) = a\Big(x(1) + a\Big(x(2) + \cdots + a(x(N-1) + a.x(0))\cdots\Big)\Big) \qquad \text{(III.5)}$$

Les calculs seront alors faits séquentiellement à partir des parenthèses les plus internes.

Le réseau systolique linéaire donné sur la figure III.6.a, dont la structure de ses cellules de base est représentée sur la figure III.6.b, permet de calculer la TFD suivant la relation (III.5) pour N = 4. En effet l'application de la relation (III.5) au cas N = 4 donne (avec $a = W_4^k$) :

$$X(k) = a\Big(x(1) + a\Big(x(2) + a(x(3) + a.x(0))\Big)\Big) \qquad \text{pour } k : 0..3$$

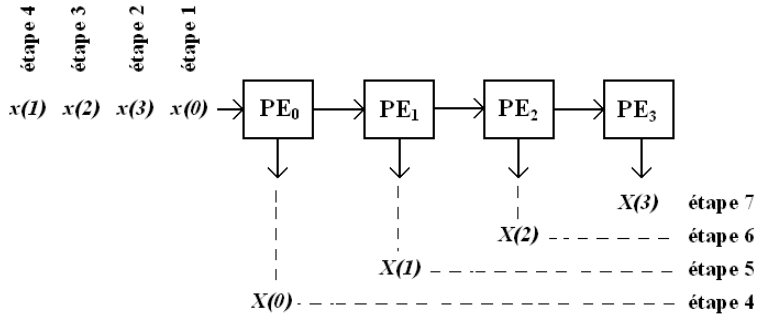


Figure III.6.a. Réseau systolique linéaire pour le calcul de la TFD, suivant le schéma de Horner, pour N = 4.

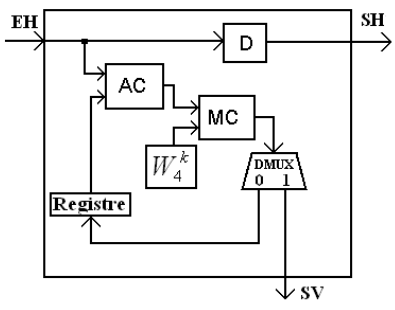


Figure III.6.b. Cellule de base du réseau systolique linéaire.

Pour le cas général, le nombre de PEs nécessaires pour le calcul des N composantes $X(k)$ est égal à N car chaque composante $X(k)$ sera calculée au niveau d'un processeur PE_k. Pour le cas étudié (N=4), 4 PEs sont nécessaires. Chaque PE, comme le montre la figure III.6.b, comporte un multiplieur complexe, un additionneur complexe, un registre/accumulateur, un deuxième registre contenant le facteur de transformation W_4^k, un élément de retard noté D (pour Delay element) et un démultiplexeur.

Le fonctionnement d'un PE est défini par ses entrées/sorties (EH/SH : entrée/sortie horizontales, SV : sortie verticale) :

Entrées des PEs :

Pour $k = 0$ (PE_0)

- $EH_0 \leftarrow$ données injectées au réseau suivant la séquence de la figure III.6.a

Pour $k > 0$ (PEs suivants)

- $EH_k \leftarrow SH_{(k-1)}$

Etat du registre/accumulateur et sorties des PEs :

- $SH_k(t+1) \leftarrow EH_k(t)$ (un élément de retard est introduit entre l'entrée et la sortie)
- $Reg_k(t+1) \leftarrow [Reg_k(t) + EH_k(t)] * W_4^k$ (pour les 3 1ères étapes, (N-1 pour le cas général))
- $SV_k(t+1) \leftarrow [Reg_k(t) + EH_k(t)] * W_4^k$ (à la dernière étape)

{étapes propres à chaque PE}

Chaque PE_k nécessite quatre étapes (N étapes dans le cas général) pour calculer sa composante $X(k)$. Le registre/accumulateur est initialisé à 0 avant tout calcul. Pendant les trois 1ères étapes le calcul $Reg_k(t+1) \leftarrow [Reg_k(t) + EH_k(t)] * W_4^k$ est effectué puis, à la quatrième et dernière étape, le même calcul est fait sauf l'accumulation au niveau du registre : la composante $X(k)$ étant complètement calculée, elle est disponible sur la sortie verticale du PE. Un bit de contrôle peut précéder la première donnée pour initialiser l'accumulateur avant tout calcul et un autre peut suivre la dernière donnée pour diriger le résultat, grâce au démultiplexeur, vers la sortie. L'ensemble des calculs aux différentes étapes, pour chaque PE, sont clairement explicités sur le tableau III.2.

Etape	PE_0	PE_1	PE_2	PE_3
1	$R_0 \leftarrow (R_0+x(0))*1$			
2	$R_0 \leftarrow (R_0+x(3))*1$	$R_1 \leftarrow (R_1+x(0))*W^1$		
3	$R_0 \leftarrow (R_0+x(2))*1$	$R_1 \leftarrow (R_1+x(3))*W^1$	$R_2 \leftarrow (R_2+x(0))*W^2$	
4	$SV_0 \leftarrow (R_0+x(1))*1=$***X(0)***	$R_1 \leftarrow (R_1+x(2))*W^1$	$R_2 \leftarrow (R_2+x(3))*W^2$	$R_3 \leftarrow (R_3+x(0))*W^3$
5		$SV_1 \leftarrow (R_1+x(1))*W^1=$***X(1)***	$R_2 \leftarrow (R_2+x(2))*W^2$	$R_3 \leftarrow (R_3+x(3))*W^3$
6			$SV_2 \leftarrow (R_2+x(1))*W^2=$***X(2)***	$R_3 \leftarrow (R_3+x(2))*W^3$
7				$SV_3 \leftarrow (R_3+x(1))*W^3=$***X(3)***

Tableau III.2. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire pour le calcul de la TFD, suivant le schéma de Horner, pour N = 4.

Si chaque PE nécessite N étapes pour calculer sa composante $X(k)$, $(2N-1)$ étapes sont requises pour le calcul de tout le spectre (7 étapes pour N=4). Ce nombre peut être facilement calculé en suivant la progression du dernier terme entré c'est-à-dire $x(1)$.
Si plusieurs TFDs sont à calculer sur des trames d'échantillons continus et adjacents, un nouveau spectre sera calculé toutes les N étapes. Ce nombre peut s'avérer élevé si le nombre d'échantillons N est important. Les réseaux systoliques planaires, sujet de la partie 9.4, permettent, comme on le verra, un débit en données beaucoup plus élevé et sont donc plus aptes à traiter des volumes plus importants de données. Ils sont aussi parfaitement adaptés pour le calcul matriciel (partie 9.3), omniprésent dans les applications du traitement du signal.

9.3. Implémentation du produit matriciel sur des réseaux systoliques planaires

Il a été vérifié que beaucoup d'applications dans des domaines tels que le traitement du signal et en particulier le traitement d'image utilisent des représentations matricielles et où le produit matriciel est quasiment omniprésent lors des calculs à effectuer. C'est le cas d'ailleurs de l'évaluation de la transformée de Fourier discrète et du filtrage numérique. Il est donc impératif d'étudier l'implémentation du produit matriciel sur un réseau systolique. Plusieurs algorithmes permettant une telle implémentation et utilisant différentes topologies ont été proposés dans [33, 91, 110 – 112, 137 – 142], mais il a été prouvé que la topologie planaire et orthogonale, appelée aussi topologie en grille, reste la mieux appropriée. Les algorithmes proposés ci-après utilisent donc un réseau systolique à topologie planaire orthogonale. Il faut noter aussi que pour cette même topologie, plusieurs variantes du même algorithme sont possibles. Ces variantes dépendent de la façon dont sont fournis les éléments des matrices au réseau. La structure du réseau systolique utilisé restant la même, il faudrait alors faire des modifications sur les processeurs élémentaires ou leur inclure une certaine reconfigurabilité pour les adapter aux calculs relatifs à chacune de ces variantes.

Deux variantes pour le calcul du produit de deux matrices sur un réseau systolique à topologie orthogonal et deux autres pour le produit d'une matrice par un vecteur sur un réseau systolique linéaire seront exposées ci-dessous. Le produit d'une matrice par un vecteur reste un cas particulier du produit de deux matrices et peut être implémenté sur un réseau systolique linéaire. Les deux variantes pour le produit de deux matrices permettraient, en les combinant sur un réseau planaire à cellules reconfigurables, de déduire un algorithme performant pour le calcul du produit de trois matrices qu'on retrouve souvent dans les applications du traitement d'images.

9.3.1. 1er réseau systolique planaire pour le calcul du produit de 2 matrices

L'algorithme présenté ici effectue le produit C = A.B, où A et B sont deux matrices carrées de taille (n x n). Ce produit n'est autre que le calcul de n^2 produits scalaires. Si à chaque produit scalaire est associé un processeur élémentaire PE, n^2 PEs seront alors nécessaires. La répartition de ces n^2 PEs est donnée sur la figure III.7.a où, à titre d'exemple n = 4. Le temps d'exécution obtenu est linéaire en n.

Chaque processeur PE_{ij} aura à effectuer le calcul suivant l'algorithme:

$$c_{ij} \leftarrow 0$$
$$Pour \quad k = 1 \quad à \quad n$$
$$c_{ij} \leftarrow c_{ij} + a_{ik} \times b_{kj} \quad (opération\ MAC)$$

La façon et l'ordre avec lesquels sont fournis les éléments des deux matrices A et B au réseau sont importants. Ceci est parfaitement illustré sur la figure III.7.a. La figure III.7.b. représente un PE. Chaque PE_{ij} possède un multiplieur, un additionneur et un registre interne, noté Reg_{ij}, jouant le rôle d'accumulateur et servant à stocker le résultat c_{ij}. Tous les registres sont initialisés à zéro avant le début du processus c'est-à-dire avant la première étape.

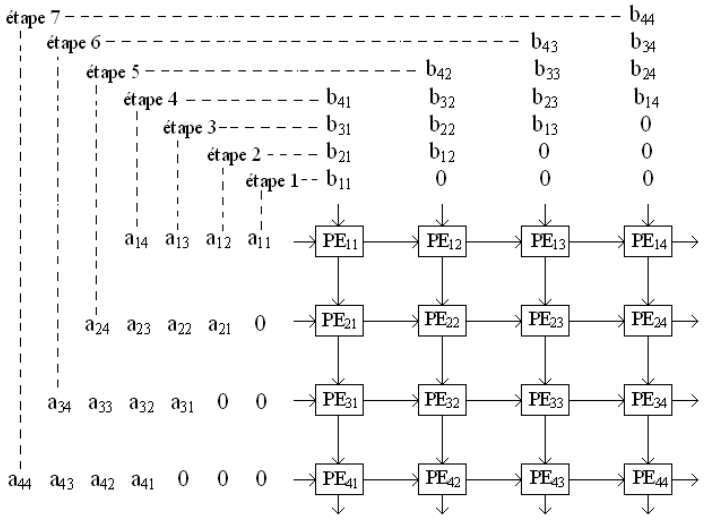


Figure III.7.a. Réseau systolique planaire à connexions orthogonales, avec vidage, pour le calcul du produit de deux matrices carrées de taille (4x4).

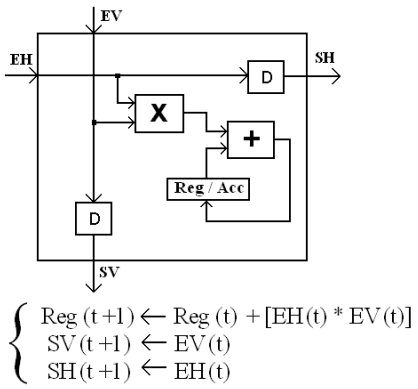


Figure III.7.b. Structure de la cellule élémentaire.

A chaque top d'horloge, toutes les cellules du réseau sont activées simultanément. Le processeur, dont l'accumulateur Reg_{ij} a la valeur $c_{ij}(t)$, reçoit les données $a_{ik}(t)$ et $b_{kj}(t)$ présentes sur ses entrées, effectue le calcul suivant l'algorithme ci-dessus, et met à jour son registre accumulateur avec la nouvelle valeur $c_{ij}(t+1)$. Il transmet alors les données $a_{ik}(t)$ et $b_{kj}(t)$ inchangées, mais avec un retard d'une étape, grâce aux éléments de retard D, aux PEs

voisins respectivement de droite et d'en bas. Les équations représentant le mode de fonctionnement d'un PE, en fonction de ses entrées, de ses sorties et du contenu de son accumulateur, sont donnés sur la figure III.7.b. Les entrées et sorties horizontales et verticales sont dénotées par leurs initiales. A la fin des calculs, les éléments de la matrice résultat C se trouvent au niveau des registres des PEs. Une procédure de vidage, qui prendra n étapes, et qui se fera par le biais des cellules frontières de la dernière colonne ou de la dernière ligne, seules aptes à communiquer avec le système hôte, doit être alors prévue.

Le nombre total d'étapes pour effectuer le produit de deux matrices carrées (n x n) sur un réseau (n x n) est égal à (3n-2). Ce résultat peut être aisément trouvé en suivant l'évolution des coefficients a_{nn} ou b_{nn} (respectivement a_{44} ou b_{44} sur la figure III.7.a.), derniers à rentrer dans le réseau. En raisonnant sur a_{nn} (dernière ligne), il faut compter (n-1) étapes dues au décalage des lignes, (n-1) étapes pour les autres coefficients à rentrer dans le réseau, et n étapes à a_{nn} pour rentrer et traverser les n cellules, soit un total de (3n-2). Cela donne ((3.4)-2) = 10 étapes dans le cas de l'exemple de la figure III.7.a.

Le comportement des différents PEs du réseau de la figure III.7.a, pendant les dix différentes étapes de calcul, est clairement montré sur les quatre tableaux III.3. Chacun des tableaux correspond aux processeurs d'une des quatre colonnes du réseau.

PEs de la colonne 1

Etape	PE_{11}	PE_{21}	PE_{31}	PE_{41}
1	$R_{11} \leftarrow R_{11}+a_{11}*b_{11}$			
2	$R_{11} \leftarrow R_{11}+a_{12}*b_{21}$	$R_{21} \leftarrow R_{21}+a_{21}*b_{11}$		
3	$R_{11} \leftarrow R_{11}+a_{13}*b_{31}$	$R_{21} \leftarrow R_{21}+a_{22}*b_{21}$	$R_{31} \leftarrow R_{31}+a_{31}*b_{11}$	
4	$R_{11} \leftarrow R_{11}+a_{14}*b_{41}=\mathbf{c_{11}}$	$R_{21} \leftarrow R_{21}+a_{23}*b_{31}$	$R_{31} \leftarrow R_{31}+a_{32}*b_{21}$	$R_{41} \leftarrow R_{41}+a_{41}*b_{11}$
5		$R_{21} \leftarrow R_{21}+a_{24}*b_{41}=\mathbf{c_{21}}$	$R_{31} \leftarrow R_{31}+a_{33}*b_{31}$	$R_{41} \leftarrow R_{41}+a_{42}*b_{21}$
6			$R_{31} \leftarrow R_{31}+a_{34}*b_{41}=\mathbf{c_{31}}$	$R_{41} \leftarrow R_{41}+a_{43}*b_{31}$
7				$R_{41} \leftarrow R_{41}+a_{44}*b_{41}=\mathbf{c_{41}}$

PEs de la colonne 2

Etape	PE_{12}	PE_{22}	PE_{32}	PE_{42}
2	$R_{12} \leftarrow R_{12}+a_{11}*b_{12}$			
3	$R_{12} \leftarrow R_{12}+a_{12}*b_{22}$	$R_{22} \leftarrow R_{22}+a_{21}*b_{12}$		
4	$R_{12} \leftarrow R_{12}+a_{13}*b_{32}$	$R_{22} \leftarrow R_{22}+a_{22}*b_{22}$	$R_{32} \leftarrow R_{32}+a_{31}*b_{12}$	
5	$R_{12} \leftarrow R_{12}+a_{14}*b_{42}=\mathbf{c_{12}}$	$R_{22} \leftarrow R_{22}+a_{23}*b_{32}$	$R_{32} \leftarrow R_{32}+a_{32}*b_{22}$	$R_{42} \leftarrow R_{42}+a_{41}*b_{12}$
6		$R_{22} \leftarrow R_{22}+a_{24}*b_{42}=\mathbf{c_{22}}$	$R_{32} \leftarrow R_{32}+a_{33}*b_{32}$	$R_{42} \leftarrow R_{42}+a_{42}*b_{22}$
7			$R_{32} \leftarrow R_{32}+a_{34}*b_{42}=\mathbf{c_{32}}$	$R_{42} \leftarrow R_{42}+a_{43}*b_{32}$
8				$R_{42} \leftarrow R_{42}+a_{44}*b_{42}=\mathbf{c_{42}}$

PEs de la colonne 3

Etape	PE_{13}	PE_{23}	PE_{33}	PE_{43}
3	$R_{13} \leftarrow R_{13}+a_{11}*b_{13}$			
4	$R_{13} \leftarrow R_{13}+a_{12}*b_{23}$	$R_{23} \leftarrow R_{23}+a_{21}*b_{13}$		
5	$R_{13} \leftarrow R_{13}+a_{13}*b_{33}$	$R_{23} \leftarrow R_{23}+a_{22}*b_{23}$	$R_{33} \leftarrow R_{33}+a_{31}*b_{13}$	
6	$R_{13} \leftarrow R_{13}+a_{14}*b_{43}=\mathbf{c_{13}}$	$R_{23} \leftarrow R_{23}+a_{23}*b_{33}$	$R_{33} \leftarrow R_{33}+a_{32}*b_{23}$	$R_{43} \leftarrow R_{43}+a_{41}*b_{13}$
7		$R_{23} \leftarrow R_{23}+a_{24}*b_{43}=\mathbf{c_{23}}$	$R_{33} \leftarrow R_{33}+a_{33}*b_{33}$	$R_{43} \leftarrow R_{43}+a_{42}*b_{23}$
8			$R_{33} \leftarrow R_{33}+a_{34}*b_{43}=\mathbf{c_{33}}$	$R_{43} \leftarrow R_{43}+a_{43}*b_{33}$
9				$R_{43} \leftarrow R_{43}+a_{44}*b_{43}=\mathbf{c_{43}}$

PEs de la colonne 4

Etape	PE_{14}	PE_{24}	PE_{34}	PE_{44}
4	$R_{14} \leftarrow R_{14}+a_{11}*b_{14}$			
5	$R_{14} \leftarrow R_{14}+a_{12}*b_{24}$	$R_{24} \leftarrow R_{24}+a_{21}*b_{14}$		
6	$R_{14} \leftarrow R_{14}+a_{13}*b_{34}$	$R_{24} \leftarrow R_{24}+a_{22}*b_{24}$	$R_{34} \leftarrow R_{34}+a_{31}*b_{14}$	
7	$R_{14} \leftarrow R_{14}+a_{14}*b_{44}=\mathbf{c_{14}}$	$R_{24} \leftarrow R_{24}+a_{23}*b_{34}$	$R_{34} \leftarrow R_{34}+a_{32}*b_{24}$	$R_{44} \leftarrow R_{44}+a_{41}*b_{14}$
8		$R_{24} \leftarrow R_{24}+a_{24}*b_{44}=\mathbf{c_{24}}$	$R_{34} \leftarrow R_{34}+a_{33}*b_{34}$	$R_{44} \leftarrow R_{44}+a_{42}*b_{24}$
9			$R_{34} \leftarrow R_{34}+a_{34}*b_{44}=\mathbf{c_{34}}$	$R_{44} \leftarrow R_{44}+a_{43}*b_{34}$
10				$R_{44} \leftarrow R_{44}+a_{44}*b_{44}=\mathbf{c_{44}}$

Tableaux III.3. Evolution des calculs au niveau de chaque PE du réseau systolique planaire, avec vidage, pour le calcul du produit de deux matrices C = A*B.

9.3.2. 2ème réseau systolique planaire pour le calcul du produit de 2 matrices

Ce deuxième réseau systolique, équivalent par son architecture au premier réseau, mais dont la structure interne des cellules est différente, permet aussi de réaliser le produit C=A*B. Néanmoins les éléments de la matrice résultat C seront directement disponibles à la sortie des cellules frontières, donc sans nécessiter la procédure de vidage du premier réseau proposé. Il faut pourtant remarquer que cet avantage n'est acquis que par l'ajout d'une procédure de pré chargement des éléments d'une des deux matrices dans les cellules correspondantes du réseau. On peut donc dire, qu'en termes de nombres d'opérations totales, les deux réseaux seront équivalents.

En considérant que la matrice B soit pré stockée dans le réseau, chaque élément b_{ij} est donc dans son PE_{ij} correspondant. Les éléments de la matrice A sont fournis au réseau suivant l'ordre donné sur la figure III.8.a., c'est-à-dire la colonne *j* de la matrice A injectée à la ligne *i* du réseau tel que $i = j$. Les composantes de la matrice C seront disponibles à la sortie du réseau par l'intermédiaire des cellules frontières de la dernière ligne. La figure III.8.b. représente la structure d'un PE. Chaque PE_{ij} possède un multiplieur, un additionneur et un registre interne,

noté Reg_{ij}, servant à stocker l'élément b_{ij}. Un élément de retard est introduit entre les entrée et sortie horizontales.

Toutes les cellules du réseau étant activées simultanément, à chaque top d'horloge, le processeur PE_{ij} effectue une multiplication de la donnée présente sur son entrée horizontale par le contenu de son registre Reg_{ij}, additionne le résultat à la donnée présente sur son entrée verticale, puis transmet ce dernier résultat à sa cellule voisine d'en bas. Il transmet aussi la donnée présente sur son entrée horizontale, avec un retard d'une étape, à sa cellule voisine de droite. Les équations régissant ce fonctionnement, en fonction des entrées, des sorties et du contenu du registre de la cellule, sont donnés sur la figure III.8.b. Pendant toutes les étapes de calcul, les entrées verticales des cellules de la première ligne sont à zéro. Le nombre total d'étapes pour effectuer le produit de deux matrices carrées (n x n) est le même que celui du 1er réseau présenté, c'est-à-dire (3n-2) étapes (10 étapes dans le cas de notre exemple). Ce résultat peut être facilement trouvé en suivant l'évolution du coefficient a_{nn}, dernier à rentrer dans le réseau. Il faut juste remarquer qu'à partir de l'étape n (4 dans notre cas d'exemple), des composantes de la matrice C commencent à être disponibles à la sortie du réseau : l'élément c_{ij} (i, j : 1..n) sera disponible à la sortie de la cellule PE_{nj} à l'étape [(n-2)+(i+j)]. Le premier élément c_{11} sera donc calculé à l'étape n, et le dernier élément c_{nn} le sera à l'étape (3n-2).

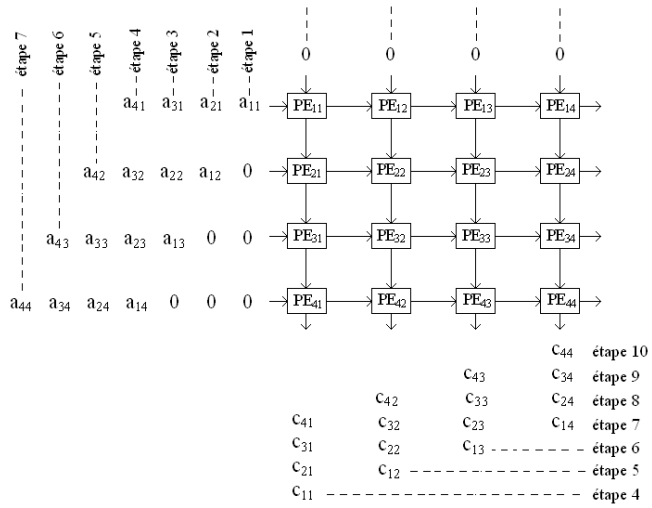


Figure III.8.a. Réseau systolique planaire à connexions orthogonales, sans vidage, pour le calcul du produit de deux matrices carrées de taille (4x4).

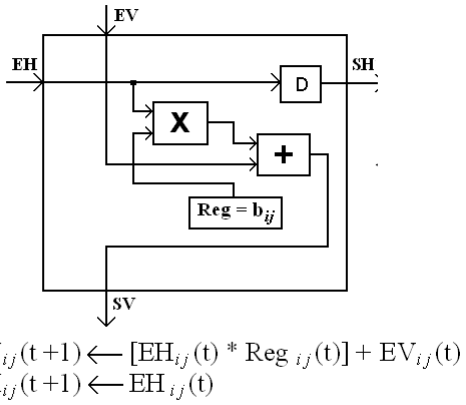


Figure III.8.b. Structure de la cellule élémentaire.

Les différentes étapes de calcul au niveau de chaque PE sont données sur les tableaux III.4. Chaque tableau correspond aux processeurs d'une colonne du réseau.

PEs de la colonne 1

Etape	PE_{11}	PE_{21}	PE_{31}	PE_{41}
1	$SV_{11} \leftarrow 0+a_{11}*b_{11}$			
2	$SV_{11} \leftarrow 0+a_{21}*b_{11}$	$SV_{21} \leftarrow EV_{21}+a_{12}*b_{21}$		
3	$SV_{11} \leftarrow 0+a_{31}*b_{11}$	$SV_{21} \leftarrow EV_{21}+a_{22}*b_{21}$	$SV_{31} \leftarrow EV_{31}+a_{13}*b_{31}$	
4	$SV_{11} \leftarrow 0+a_{41}*b_{11}$	$SV_{21} \leftarrow EV_{21}+a_{32}*b_{21}$	$SV_{31} \leftarrow EV_{31}+a_{23}*b_{31}$	$SV_{41} \leftarrow EV_{41}+a_{14}*b_{41} = \mathbf{c_{11}}$
5		$SV_{21} \leftarrow EV_{21}+a_{42}*b_{21}$	$SV_{31} \leftarrow EV_{31}+a_{33}*b_{31}$	$SV_{41} \leftarrow EV_{41}+a_{24}*b_{41} = \mathbf{c_{21}}$
6			$SV_{31} \leftarrow EV_{31}+a_{43}*b_{31}$	$SV_{41} \leftarrow EV_{41}+a_{34}*b_{41} = \mathbf{c_{31}}$
7				$SV_{41} \leftarrow EV_{41}+a_{44}*b_{41} = \mathbf{c_{41}}$

PEs de la colonne 2

Etape	PE_{12}	PE_{22}	PE_{32}	PE_{42}
2	$SV_{12} \leftarrow 0+a_{11}*b_{12}$			
3	$SV_{12} \leftarrow 0+a_{21}*b_{12}$	$SV_{22} \leftarrow EV_{22}+a_{12}*b_{22}$		
4	$SV_{12} \leftarrow 0+a_{31}*b_{12}$	$SV_{22} \leftarrow EV_{22}+a_{22}*b_{22}$	$SV_{32} \leftarrow EV_{32}+a_{13}*b_{32}$	
5	$SV_{12} \leftarrow 0+a_{41}*b_{12}$	$SV_{22} \leftarrow EV_{22}+a_{32}*b_{22}$	$SV_{32} \leftarrow EV_{32}+a_{23}*b_{32}$	$SV_{42} \leftarrow EV_{42}+a_{14}*b_{42} = \mathbf{c_{12}}$
6		$SV_{22} \leftarrow EV_{22}+a_{42}*b_{22}$	$SV_{32} \leftarrow EV_{32}+a_{33}*b_{32}$	$SV_{42} \leftarrow EV_{42}+a_{24}*b_{42} = \mathbf{c_{22}}$
7			$SV_{32} \leftarrow EV_{32}+a_{43}*b_{32}$	$SV_{42} \leftarrow EV_{42}+a_{34}*b_{42} = \mathbf{c_{32}}$
8				$SV_{42} \leftarrow EV_{42}+a_{44}*b_{42} = \mathbf{c_{42}}$

PEs de la colonne 3

Etape	PE_{13}	PE_{23}	PE_{33}	PE_{43}
3	$SV_{13} \leftarrow 0+a_{11}*b_{13}$			
4	$SV_{13} \leftarrow 0+a_{21}*b_{13}$	$SV_{23} \leftarrow EV_{23}+a_{12}*b_{23}$		
5	$SV_{13} \leftarrow 0+a_{31}*b_{13}$	$SV_{23} \leftarrow EV_{23}+a_{22}*b_{23}$	$SV_{33} \leftarrow EV_{33}+a_{13}*b_{33}$	
6	$SV_{13} \leftarrow 0+a_{41}*b_{13}$	$SV_{23} \leftarrow EV_{23}+a_{32}*b_{23}$	$SV_{33} \leftarrow EV_{33}+a_{23}*b_{33}$	$SV_{43} \leftarrow EV_{43}+a_{14}*b_{43} = \mathbf{c_{13}}$
7		$SV_{23} \leftarrow EV_{23}+a_{42}*b_{23}$	$SV_{33} \leftarrow EV_{33}+a_{33}*b_{33}$	$SV_{43} \leftarrow EV_{43}+a_{24}*b_{43} = \mathbf{c_{23}}$
8			$SV_{33} \leftarrow EV_{33}+a_{43}*b_{33}$	$SV_{43} \leftarrow EV_{43}+a_{34}*b_{43} = \mathbf{c_{33}}$
9				$SV_{43} \leftarrow EV_{43}+a_{44}*b_{43} = \mathbf{c_{43}}$

PEs de la colonne 4

Etape	PE_{14}	PE_{24}	PE_{34}	PE_{44}
4	$SV_{14} \leftarrow 0+a_{11}*b_{14}$			
5	$SV_{14} \leftarrow 0+a_{21}*b_{14}$	$SV_{24} \leftarrow EV_{24}+a_{12}*b_{24}$		
6	$SV_{14} \leftarrow 0+a_{31}*b_{14}$	$SV_{24} \leftarrow EV_{24}+a_{22}*b_{24}$	$SV_{34} \leftarrow EV_{34}+a_{13}*b_{34}$	
7	$SV_{14} \leftarrow 0+a_{41}*b_{14}$	$SV_{24} \leftarrow EV_{24}+a_{32}*b_{24}$	$SV_{34} \leftarrow EV_{34}+a_{23}*b_{34}$	$SV_{44} \leftarrow EV_{44}+a_{14}*b_{44} = \mathbf{c_{14}}$
8		$SV_{24} \leftarrow EV_{24}+a_{42}*b_{24}$	$SV_{34} \leftarrow EV_{34}+a_{33}*b_{34}$	$SV_{44} \leftarrow EV_{44}+a_{24}*b_{44} = \mathbf{c_{24}}$
9			$SV_{34} \leftarrow EV_{34}+a_{43}*b_{34}$	$SV_{44} \leftarrow EV_{44}+a_{34}*b_{44} = \mathbf{c_{34}}$
10				$SV_{44} \leftarrow EV_{44}+a_{44}*b_{44} = \mathbf{c_{44}}$

Tableaux III.4. Evolution des calculs au niveau de chaque PE du réseau systolique planaire, sans vidage, pour le calcul du produit de deux matrices C = A*B.

9.3.3. 1er réseau systolique linéaire pour le calcul du produit d'une matrice par un vecteur appliqué au calcul de la TFD

Ce réseau peut être déduit directement du 1er réseau systolique planaire pour le calcul du produit de 2 matrices en substituant un vecteur de longueur n à la deuxième matrice. Le même raisonnement de fonctionnement peut alors lui être appliqué. Le produit $y = Ax$ où A est une matrice carrée, $\boldsymbol{x}$ et $\boldsymbol{y}$ sont respectivement les vecteurs d'entrée et de sortie, est schématisé sur la figure III.9.a. L'évolution des calculs au niveau de chaque PE du réseau systolique est présentée sur le tableau III.5. (2n-1) étapes sont nécessaires pour calculer le vecteur y (7 étapes pour le présent cas où n=4)

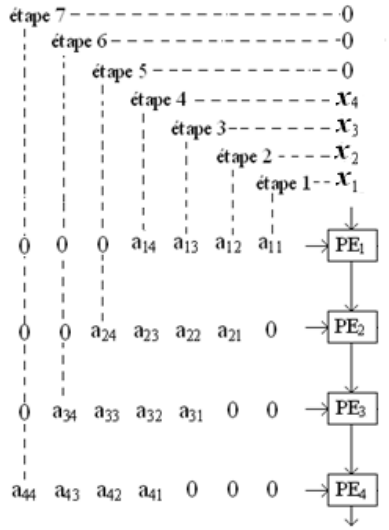
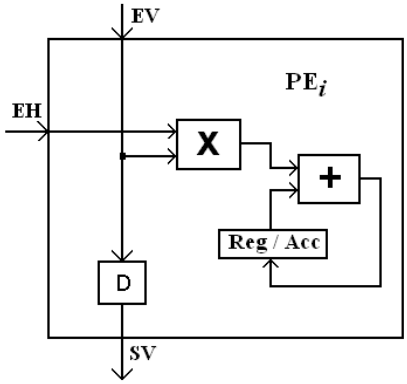


Figure III.9.a. Réseau systolique linéaire, avec vidage, pour le calcul du produit d'une matrice 4x4 par un vecteur 4x1



$$\begin{cases} Reg(t+1) \leftarrow Reg(t) + [EH(t) * EV(t)] \\ SV(t+1) \leftarrow EV(t) \end{cases}$$

Figure III.9.b. Structure de la cellule élémentaire

Etape	PE_1	PE_2	PE_3	PE_4
1	$R_1 \leftarrow R_1 + a_{11} * x_1$			
2	$R_1 \leftarrow R_1 + a_{12} * x_2$	$R_2 \leftarrow R_2 + a_{21} * x_1$		
3	$R_1 \leftarrow R_1 + a_{13} * x_3$	$R_2 \leftarrow R_2 + a_{22} * x_2$	$R_3 \leftarrow R_3 + a_{31} * x_1$	
4	$R_1 \leftarrow R_1 + a_{14} * x_4 = \mathbf{y_1}$	$R_2 \leftarrow R_2 + a_{23} * x_3$	$R_3 \leftarrow R_3 + a_{32} * x_2$	$R_4 \leftarrow R_4 + a_{41} * x_1$
5		$R_2 \leftarrow R_2 + a_{24} * x_{41} = \mathbf{y_2}$	$R_3 \leftarrow R_3 + a_{33} * x_3$	$R_4 \leftarrow R_4 + a_{42} * x_2$
6			$R_3 \leftarrow R_3 + a_{34} * x_4 = \mathbf{y_3}$	$R_4 \leftarrow R_4 + a_{43} * x_3$
7				$R_4 \leftarrow R_4 + a_{44} * x_4 = \mathbf{y_4}$

Tableau III.5. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire, avec vidage, pour le calcul du produit $y = Ax$ d'une matrice 4x4 par un vecteur 4x1.

La TFD étant un produit d'une matrice par un vecteur $X = T_N * x$ où x et X sont respectivement les vecteurs original et transformé, et T_N la matrice carrée (NxN) de transformation, le réseau précédent peut la calculer en substituant la matrice T_N à la matrice A. Pour N=4, le tableau

suivant permet de visualiser les différentes étapes de calcul. Le spectre total est calculé en 7 étapes (2*N*-1 étapes dans le cas général). Une procédure de vidage est nécessaire pour récupérer le spectre au niveau du système hôte.

Etape	PE_1	PE_2	PE_3	PE_4
1	$R_1 \leftarrow R_1 + W_4^0 * x(0)$			
2	$R_1 \leftarrow R_1 + W_4^0 * x(1)$	$R_2 \leftarrow R_2 + W_4^0 * x(0)$		
3	$R_1 \leftarrow R_1 + W_4^0 * x(2)$	$R_2 \leftarrow R_2 + W_4^1 * x(1)$	$R_3 \leftarrow R_3 + W_4^0 * x(0)$	
4	$R_1 \leftarrow R_1 + W_4^0 * x(3) = \boldsymbol{X(0)}$	$R_2 \leftarrow R_2 + W_4^2 * x(2)$	$R_3 \leftarrow R_3 + W_4^2 * x(1)$	$R_4 \leftarrow R_4 + W_4^0 * x(0)$
5		$R_2 \leftarrow R_2 + W_4^3 * x(3) = \boldsymbol{X(1)}$	$R_3 \leftarrow R_3 + W_4^4 * x(2)$	$R_4 \leftarrow R_4 + W_4^3 * x(1)$
6			$R_3 \leftarrow R_3 + W_4^6 * x(3) = \boldsymbol{X(2)}$	$R_4 \leftarrow R_4 + W_4^6 * x(2)$
7				$R_4 \leftarrow R_4 + W_4^9 * x(3) = \boldsymbol{X(3)}$

Tableau III.6. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire de la figure III.9.a. pour le calcul de la TFD pour N = 4.

9.3.4. 2ème réseau systolique linéaire pour le calcul du produit d'une matrice par un vecteur appliqué au calcul de la TFD

Ce réseau peut lui aussi être déduit directement du 2ème réseau systolique planaire pour le calcul du produit de 2 matrices donné dans 9.3.2, en substituant un vecteur de longueur n à la deuxième matrice. Le produit $y = Ax$ est schématisé sur la figure III.10.a.

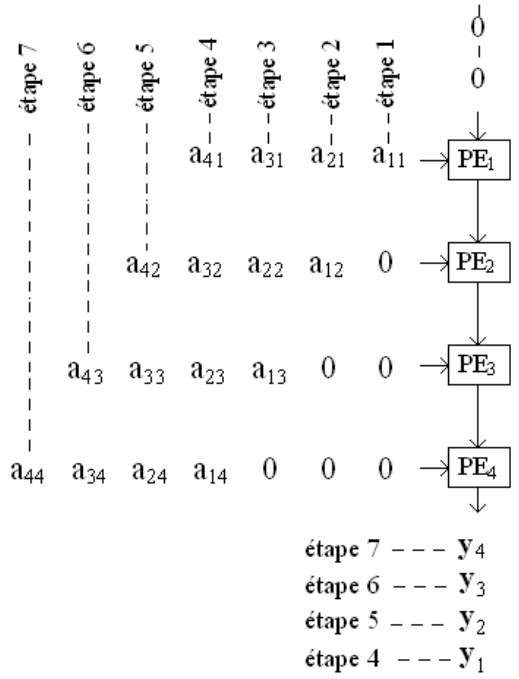


Figure III.10.a. Réseau systolique linéaire, sans vidage, pour le calcul du produit d'une matrice 4x4 par un vecteur 4x1

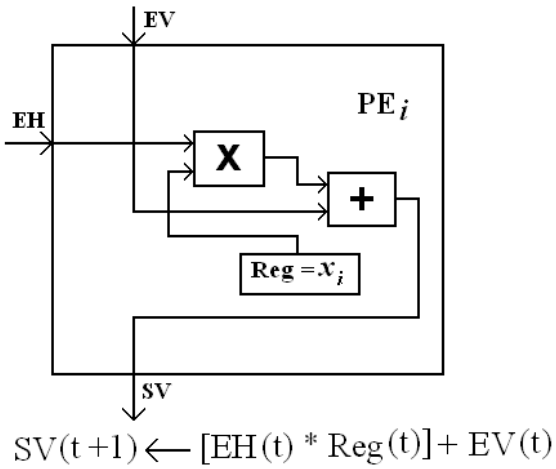


Figure III.10.b. Structure de la cellule élémentaire

L'évolution des calculs au niveau de chaque PE du réseau systolique est présentée sur le tableau III.7. Il faut (2n-1) étapes pour calculer le vecteur *y* (7 étapes pour le cas n=4) sans nécessiter une procédure de vidage étant donné qu'à partir de l'étape n, une composante spectrale est disponible à la sortie de PE_n (PE_4 pour le présent cas) pour être récupérée par le système hôte.

Etape	PE_1	PE_2	PE_3	PE_4
1	$SV_1 \leftarrow 0+a_{11}*x_1$			
2	$SV_1 \leftarrow 0+a_{21}*x_1$	$SV_2 \leftarrow EV_2+ a_{12}*x_2$		
3	$SV_1 \leftarrow 0+a_{31}*x_1$	$SV_2 \leftarrow EV_2+ a_{22}*x_2$	$SV_3 \leftarrow EV_3+ a_{13}*x_3$	
4	$SV_1 \leftarrow 0+a_{41}*x_1$	$SV_2 \leftarrow EV_2+ a_{32}*x_2$	$SV_3 \leftarrow EV_3+ a_{23}*x_3$	$SV_4 \leftarrow EV_4+ a_{14}*x_4 = \mathbf{y_1}$
5		$SV_2 \leftarrow EV_2+ a_{42}*x_2$	$SV_3 \leftarrow EV_3+ a_{33}*x_3$	$SV_4 \leftarrow EV_4+ a_{24}*x_4 = \mathbf{y_2}$
6			$SV_3 \leftarrow EV_3+ a_{43}*x_3$	$SV_4 \leftarrow EV_4+ a_{34}*x_4 = \mathbf{y_3}$
7				$SV_4 \leftarrow EV_4+ a_{44}*x_4 = \mathbf{y_4}$

Tableau III.7. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire, sans vidage, pour le calcul du produit $y = Ax$ d'une matrice 4x4 par un vecteur 4x1.

Ce réseau peut aussi implémenter la TFD en substituant la matrice T_N à la matrice *A*. Pour N=4, le tableau suivant permet de visualiser les différentes étapes de calcul. Le spectre total est calculé en 7 étapes ($2N$-1 étapes dans le cas général) sans nécessiter une procédure de vidage.

Etape	PE_1	PE_2	PE_3	PE_4
1	$SV_1 \leftarrow 0+ W_4^0*x(0)$			
2	$SV_1 \leftarrow 0+ W_4^0*x(0)$	$SV_2 \leftarrow EV_2+ W_4^0*x(1)$		
3	$SV_1 \leftarrow 0+ W_4^0*x(0)$	$SV_2 \leftarrow EV_2+ W_4^1*x(1)$	$SV_3 \leftarrow EV_3+ W_4^0*x(2)$	
4	$SV_1 \leftarrow 0+ W_4^0*x(0)$	$SV_2 \leftarrow EV_2+ W_4^2*x(1)$	$SV_3 \leftarrow EV_3+ W_4^2*x(2)$	$SV_4 \leftarrow EV_4+W_4^0*x(3) = \boldsymbol{X(0)}$
5		$SV_2 \leftarrow EV_2+ W_4^3*x(1)$	$SV_3 \leftarrow EV_3+ W_4^4*x(2)$	$SV_4 \leftarrow EV_4+W_4^3*x(3) = \boldsymbol{X(1)}$
6			$SV_3 \leftarrow EV_3+ W_4^6*x(2)$	$SV_4 \leftarrow EV_4+W_4^6*x(3) = \boldsymbol{X(2)}$
7				$SV_4 \leftarrow EV_4+W_4^9*x(3) = \boldsymbol{X(3)}$

Tableaux III.8. Evolution des calculs au niveau de chaque PE du réseau systolique linéaire de la figure III.10.a. pour le calcul de la TFD pour $N = 4$.

9.4. Réseaux systoliques planaires pour le calcul de la TFD

La TFD à *N* échantillons peut facilement être implémentée sur un réseau systolique planaire qui requerrait au plus (*N* x *N)* PEs, chacun effectuant principalement une opération MAC. Nous proposons dans ce qui suit deux structures systoliques planaires pour le calcul de la TFD. Il faut toutefois noter que plusieurs différentes structures systoliques planaires pour le calcul de la TFD ont été proposées dans la littérature [72, 135, 136, 143, 144]. Si pour ces différentes implémentations le nombre de nœuds, donc de PEs, reste pratiquement le même et

est au plus égal à N^2, leurs différences se situent au niveau de la façon dont sont introduits les différents échantillons *x(n)* et facteurs de transformation *W* dans le réseau. Cela affectera plus ou moins la structure interne de la cellule de base ou plus précisément l'arrangement de son multiplieur et son additionneur complexes ainsi que l'élément de retard. Il faut aussi remarquer que les réseaux systoliques planaires permettent un débit en données beaucoup plus important que leurs homologues linéaires si les calculs sont faits sur une suite continuelle de trames d'échantillons temporels adjacentes l'une à l'autre pour le calcul de plusieurs TFDs.

9.4.1. 1er réseau systolique planaire proposé pour le calcul de la TFD

Le réseau, représenté sur la figure III.11.a. et traitant $N = 4$ échantillons, comprend un seul type de PEs avec l'annexion de quelques éléments de retard dans sa première ligne. La structure interne de ces PEs est donnée sur la figure III.11.b. Les PEs sont indicés par les variables *i* et *j* (*i*, *j* : 0..*N*-1) qui indiquent respectivement les numéros de ligne et de colonne auxquelles ils appartiennent. Le comportement du réseau peut être facilement interprété à partir de la figure III.11.a. Les échantillons *x(n)* et les coefficients de transformation *W* sont injectés au réseau par le biais des cellules limitrophes de gauche et les composantes spectrales ou résultats sont récupérés aux sorties des PEs de la dernière ligne du réseau. A chaque étape de calcul, chaque PE du réseau effectue une opération MAC sur les données présentes sur ses entrées et transmet le résultat et une partie des ses entrées à ses PEs voisins de droite et de dessous. Ces calculs sont explicitement définis par les relations exprimant les entrées et sorties des PEs. Sur la figure III.11.b. les entrées horizontale et verticales ont été dénotées respectivement par EH et EV*k* (*k* :1, 2), et les sorties par SH et SV*k* (*k* :1, 2). Ces entrées et sorties sont indicées avec les mêmes indices de leurs PEs respectifs.

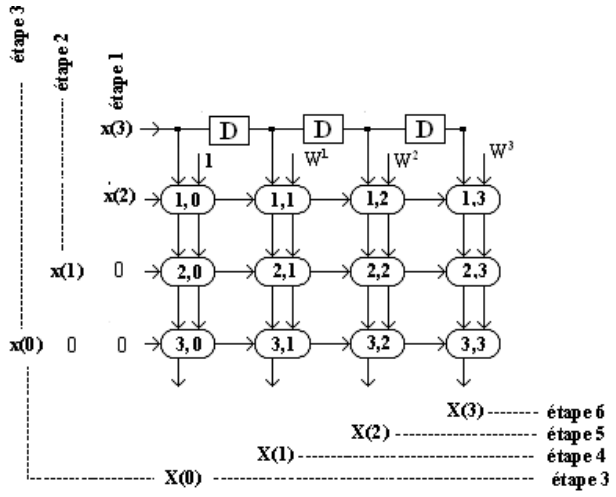


Figure III.11.a. 1er réseau systolique planaire proposé pour le calcul de la TFD, pour $N = 4$.

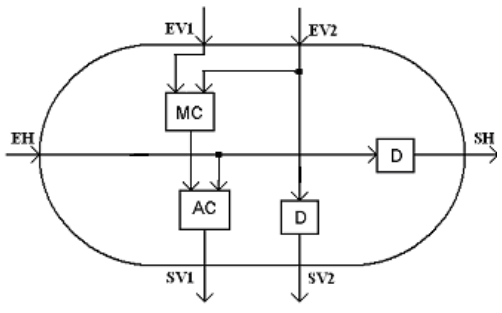


Figure III.11.b. Cellule de base.

Fonctionnement des entrées et sorties des PEs

Entrées des PEs :

Pour tous le PEs de la colonne j=0

$EH_{i\,0}$← données injectées au réseau (séquences selon la figure III.11.a.)

Pour $i = 0$ (ligne 0) Eléments de retard D avec $x(N\text{-}1)$ sur le 1er élément D.

Pour $i = 1$ (ligne 1) - $EH_{1j} \leftarrow SH_{1(j\text{-}1)}$ (j > 0)

- $EV1_{1j} \leftarrow x(N\text{-}1)$ et $EV2_{1\,j} \leftarrow W^{j}$

Pour $i > 1$ (lignes autres que les lignes 0 et 1)

- $EH_{ij} \leftarrow SH_{i(j\text{-}1)}$ (j > 0)
- $EV1_{i\,j} \leftarrow SV1_{(i\text{-}1)\,j}$ et $EV2_{i\,j} \leftarrow SV2_{(i\text{-}1)\,j}$

Sorties des PEs :

- $SH_{i\,j}(t+1) \leftarrow EH_{i\,j}(t)$ (un élément de retard est introduit entre l'entrée et la sortie)
- $SV1_{i\,j}(t+1) \leftarrow EH_{i\,j}(t) + EV1_{i\,j}(t) * EV2_{i\,j}(t)$
- $SV2_{i\,j}(t+1) \leftarrow EV2_{i\,j}(t)$ (un élément de retard est introduit entre l'entrée et la sortie)

Pour $N = 4$, six étapes sont nécessaires pour le calcul de tout le spectre. Ce nombre d'étapes peut être déterminé en raisonnant sur l'évolution de l'échantillon *x(0)*, dernier élément à entrer dans le réseau : il entre dans le réseau à la 3ème étape et traverse le reste des PEs en 3 étapes. Dans le cas général (2N-2) étapes seront nécessaires pour le calcul du spectre. Les six étapes de calcul ont été regroupées sur les quatre tableaux suivants. Chaque tableau donne l'évolution des états des PEs, pour une colonne du réseau.

Colonne 0

Etape	PE_{10}	PE_{20}	PE_{30}
1	$SV1_{10} \leftarrow EV1_{10} * 1 + x(2)$		
2		$SV1_{20} \leftarrow EV1_{20} * 1 + x(1)$	
3			$SV1_{30} \leftarrow EV1_{30} * 1 + x(0) = x(3)+x(2)+x(1)+x(0) = X(0)$

Colonne 1

Etape	PE_{11}	PE_{21}	PE_{31}
2	$SV1_{11} \leftarrow EV1_{11} * W^{1} + x(2)$		
3		$SV1_{21} \leftarrow EV1_{21} * W^{1} + x(1)$	
4			$SV1_{31} \leftarrow EV1_{31} * W^{1} + x(0) = x(3)W^{3}+x(2)W^{2}+x(1)W^{1}+x(0) = X(1)$

Colonne 2

Etape	PE_{12}	PE_{22}	PE_{32}
3	$SV1_{12} \leftarrow EV1_{12} * W^{2} + x(2)$		
4		$SV1_{22} \leftarrow EV1_{22} * W^{2} + x(1)$	
5			$SV1_{32} \leftarrow EV1_{32} * W^{2} + x(0) = x(3)W^{2}+x(2)+x(1)W^{2}+x(0) = X(2)$

Colonne 3

Etape	PE_{13}	PE_{23}	PE_{33}
4	$SV1_{13} \leftarrow EV1_{13} * W^{3} + x(2)$		
5		$SV1_{23} \leftarrow EV1_{23} * W^{3} + x(1)$	
6			$SV1_{33} \leftarrow EV1_{33} * W^{3} + x(0) = x(3)W^{1}+x(2)W^{2}+x(1)W^{3}+x(0) = X(3)$

Tableaux III.9. Evolution des calculs sur chaque PE du 1er réseau systolique planaire proposé.

Le temps d'une étape de calcul est directement lié au temps de réponse des PEs. Chaque PE effectue une opération MAC complexe : une multiplication complexe suivie d'une addition complexe. Ces opérations sont en fait composées d'opérations réelles (Section 2 du chapitre II) : un multiplieur complexe est composé d'une rangée de 4 multiplieurs réels suivie d'une rangée de 2 additionneurs réels, et un additionneur complexe d'une rangée de 2 additionneurs réels. Le temps d'une étape, qu'on note TR, est donc donné par : $TR = t_m + 2.t_a$ (t_m et t_a sont les temps de réponse respectivement d'un multiplieur réel et d'un additionneur réel) .

Le nombre d'étapes pour le calcul de tout le spectre étant de $2N$-2, la latence du réseau est :

$$(2N\text{-}2).TR = (2N\text{-}2).(t_m + 2.t_a)$$

Le nombre d'éléments d'un tel réseau est :

(N-1) éléments de retard (1ère ligne) + $N(N\text{-}1)$ PEs

Le nombre d'opérateurs d'un tel réseau est donc : $N(N\text{-}1)$ additionneurs complexes, $N(N\text{-}1)$ multiplieurs complexes, $2N(N\text{-}1)$ éléments de retard appartenant aux PEs et (N-1) éléments de retard de la 1ère ligne du réseau. Cela donne : $4N(N\text{-}1)$ multiplieurs réels, $4N(N\text{-}1)$ additionneurs réels et $(2N+1)(N\text{-}1)$ éléments de retard.

Les PEs étant en continuelle activité en faisant toujours le même calcul, si le réseau est alimenté par des trames d'échantillons temporels adjacentes l'une à l'autre, on remarque qu'à partir de l'étape 4, des composantes appartenant à différents spectres sont récupérées en même temps aux sorties du réseau et qu'un nouveau spectre complet sera calculé à chaque étape qui suit l'étape 6 (étape ($2N$-2) dans le cas général). Pour montrer cela, trois vecteurs, indicés de 1 à 3, de 4 échantillons chacun sont injectés au réseau suivant l'ordre décrit sur la figure III.12.

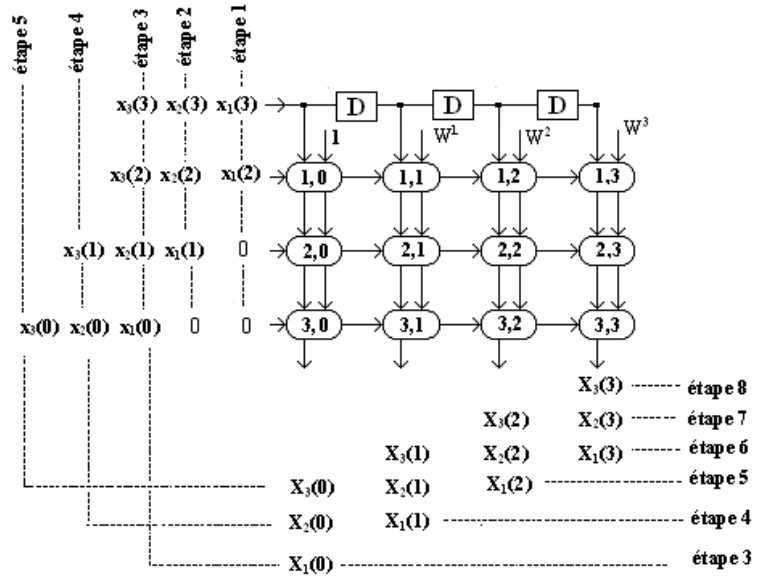


Figure III.12. Comportement du réseau systolique planaire proposé lors d'un traitement d'un flux continuel d'échantillons temporels.

L'évolution séquentielle des calculs est exposée explicitement sur le tableau suivant.

Etape	PE_{10} $SV1_{10}\leftarrow$	PE_{20} $SV1_{20}\leftarrow$	PE_{30} $SV1_{30}\leftarrow$	PE_{11} $SV1_{11}\leftarrow$	PE_{21} $SV1_{21}\leftarrow$	PE_{31} $SV1_{31}\leftarrow$
1	$EV1_{10}$ ***1**+x_1*(2)*					
2	$EV1_{10}$ ***1**+x_2*(2)*	$EV1_{20}$ ***1**+x_1*(1)*		$EV1_{11}$ ***w*^{1}+x_1*(2)*		
3	$EV1_{10}$ ***1**+x_3*(2)*	$EV1_{20}$ ***1**+x_2*(1)*	$EV1_{30}$+x_1*(0)=X_1(0)*	$EV1_{11}$ ***w*^{1}+x_2*(2)*	$EV1_{21}$ ***w*^{1}+x_1*(1)*	
4		$EV1_{20}$ ***1**+x_3*(1)*	$EV1_{30}$+x_2*(0)=X_2(0)*	$EV1_{11}$ ***w*^{1}+x_3*(2)*	$EV1_{21}$ ***w*^{1}+x_2*(1)*	$EV1_{31}$ ***w*^{1}+x_1*(0) =X_1(1)*
5			$EV1_{30}$+x_3*(0)=X_3(0)*		$EV1_{21}$ ***w*^{1}+x_3*(1)*	$EV1_{31}$ ***w*^{1}+x_2*(0) =X_2(1)*
6						$EV1_{31}$ ***w*^{1}+x_3*(0) =X_3(1)*

Etape	PE_{12} $SV1_{12}\leftarrow$	PE_{22} $SV1_{22}\leftarrow$	PE_{32} $SV1_{32}\leftarrow$	PE_{13} $SV1_{13}\leftarrow$	PE_{23} $SV1_{23}\leftarrow$	PE_{33} $SV1_{33}\leftarrow$
3	$EV1_{12}$ ***w*^{2}+x_1*(2)*					
4	$EV1_{12}$ ***w*^{2}+x_2*(2)*	$EV1_{22}$ ***w*^{2}+x_1*(1)*		$EV1_{13}$ ***w*^{3}+x_1*(2)*		
5	$EV1_{12}$ ***w*^{2}+x_3*(2)*	$EV1_{22}$ ***w*^{2}+x_2*(1)*	$EV1_{32}$ ***w*^{2}+x_1*(0)=X_1(2)*	$EV1_{13}$ ***w*^{3}+x_2*(2)*	$EV1_{23}$ ***w*^{3}+x_1*(1)*	
6		$EV1_{22}$ ***w*^{2}+x_3*(1)*	$EV1_{32}$ ***w*^{2}+x_2*(0)=X_2(2)*	$EV1_{13}$ ***w*^{3}+x_3*(2)*	$EV1_{23}$ ***w*^{3}+x_2*(1)*	$EV1_{33}$ ***w*^{3}+x_1*(0 =X_1(3)*
7			$EV1_{32}$ ***w*^{2}+x_3*(0)=X_3(2)*		$EV1_{23}$ ***w*^{3}+x_3*(1)*	$EV1_{33}$ ***w*^{3}+x_2*(0) =X_2(3)*
8						$EV1_{33}$ ***w*^{3}+x_3*(0) =X_3(3)*

Tableau III.10. Les différentes étapes de calcul, au niveau de chaque PE du réseau systolique planaire proposé, pour le calcul successif de 3 TFDs.

9.4.2. 2ème réseau systolique planaire proposé pour le calcul de la TFD

Un deuxième réseau systolique planaire pour le calcul de la TFD est proposé, à titre d'exemple sur la figure III.13.a, pour montrer qu'on peut avoir plusieurs différentes implémentations en changeant l'ordre et l'emplacement des échantillons *x(i)* ainsi que ceux des facteurs de transformation *W* aux entrées du réseau et aussi en modifiant la structure interne des PEs [145]. Pour calculer le spectre pour *N*=5, il a fallu 8 étapes (2*N*-2 : cas général). Les cellules de ce réseau, qui sont de 2 types, sont données avec leurs modes de fonctionnement sur les figures III.13.b et c. L'évolution des calculs est montrée sur les tableaux III.11.

Ici aussi la latence du réseau est donnée par (voir cellule du type 2) :

$(2N\text{-}2).TR = (2N\text{-}2).(t_m + 2.t_a)$

Le nombre d'éléments d'un tel réseau est : (*N*-1) éléments de retard (1ère ligne) + (*N*-1) PEs de type 1 (colonne 0) + (*N*-1)*(*N*-1) PEs de type 2 (autres colonnes)

Le nombre d'opérateurs d'un tel réseau est donc :

N(*N*-1) additionneurs complexes, (*N*-1)(*N*-1) multiplieurs complexes, 2(*N*-1)(*N*-1)+(*N*-1) = (2*N*-1)(*N*-1) éléments de retard appartenant aux PEs et (*N*-1) éléments de retard de la 1ère ligne. Cela donne : 4(*N*-1)(*N*-1) multiplieurs réels, 2(*N*-1)(2*N*-1) additionneurs réels et 2*N*(*N*-1) éléments de retard.

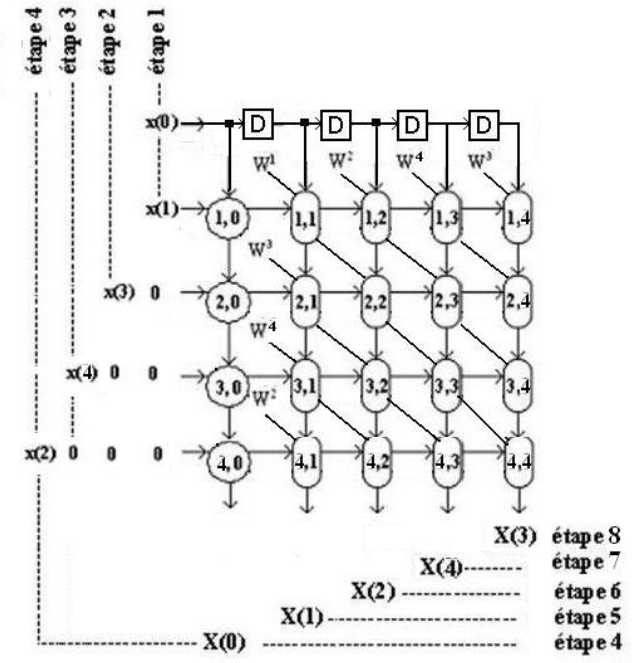


Figure III.13.a. 2ème réseau systolique planaire pour le calcul de la TFD pour *N* = 5.

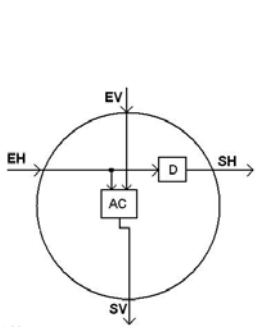


Figure III.13.b. PE de type 1 (colonne 0).

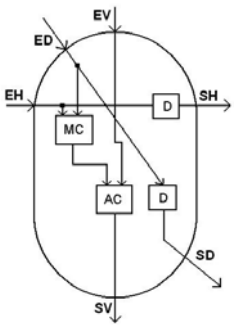


Figure III.13.c. PE de type 2. (restant des colonnes)

Définition des entrées et sorties des différents PEs :

Pour $j = 0$ (colonne 0 : PEs de type 1)

Entrées des PEs :

- $EV_{10} \leftarrow x(0)$
- $EV_{i0} \leftarrow SV_{(i-1)0}$ pour $i > 1$
- $EH_{i0} \leftarrow$ données injectées au réseau (séquences selon la figure III.13.a)

Sorties des PEs :

- $SV_{i0}(t+1) \leftarrow EH_{i0}(t) + EV_{i0}(t)$
- $SH_{i0}(t+1) \leftarrow EH_{i0}(t)$ (un élément de retard entre l'entrée et la sortie)

Pour $j > 0$ (restant des colonnes comportant les cellules de base de type 2)

Entrées des PEs :

- $EV_{1j} \leftarrow x(0)$ et $EV_{ij} \leftarrow SV_{(i-1)j}$ pour $i > 1$
- ED_{1j} et ED_{i1} pour $i > 0 \leftarrow$ les différents W^k selon la figure III.13.a
- $ED_{ij} \leftarrow SH_{(i-1)(j-1)}$ (pour $i, j > 1$)
- $EH_{ij} \leftarrow SH_{i(j-1)}$

Sorties des PEs :

- $SV_{ij}(t+1) = [EH_{ij}(t) * ED_{ij}(t)] + EV_{ij}(t)$
- $SH_{ij}(t+1) = EH_{ij}(t)$ (un élément de retard entre l'entrée et la sortie)
- $SD_{ij}(t+1) = ED_{ij}(t)$ (un élément de retard entre l'entrée et la sortie)

Colonne 0

Etape	PE_{10}	PE_{20}	PE_{30}	PE_{40}
1	$SV_{10} \leftarrow EV_{10} + x(1) = x(0)+x(1)$			
2		$SV_{20} \leftarrow EV_{20} + x(3) = x(0)+x(1)+x(3)$		
3			$SV_{30} \leftarrow EV_{30} + x(4) = x(0)+x(1)+x(3)+x(4)$	
4				$SV_{40} \leftarrow EV_{40} + x(2) = x(0)+x(1)+x(2)+x(3)+x(4) = X(0)$

Colonne 1

Etape	PE_{11}	PE_{21}	PE_{31}	PE_{41}
2	$SV_{11} \leftarrow EV_{11} + x(1)*W^1 = x(0)+x(1)*W^1$			
3		$SV_{21} \leftarrow EV_{21} + x(3)*W^3 = x(0)+x(1)*W^1 + x(3)*W^3$		
4			$SV_{31} \leftarrow EV_{31} + x(4)*W^4 = x(0)+x(1)*W^1 + x(3)*W^3 + x(4)*W^4$	
5				$SV_{41} \leftarrow EV_{41} + x(2)*W^2 = x(0)+x(1)*W^1 + x(2)*W^2 + x(3)*W^3 + x(4)*W^4 = X(1)$

Colonne 2

Etape	PE_{12}	PE_{22}	PE_{32}	PE_{42}
3	$SV_{12} \leftarrow EV_{12} + x(1)*W^2 = x(0) + x(1)*W^2$			
4		$SV_{22} \leftarrow EV_{22} + x(3)*W^1 = x(0) + x(1)*W^2 + x(3)*W^1$		
5			$SV_{32} \leftarrow EV_{32} + x(4)*W^3 = x(0) + x(1)*W^2 + x(3)*W^1 + x(4)*W^3$	
6				$SV_{42} \leftarrow EV_{42} + x(2)*W^4 = x(0) + x(1)*W^2 + x(2)*W^4 + x(3)*W^1 + x(4)*W^3 = X(2)$

Colonne 3

Etape	PE_{13}	PE_{23}	PE_{33}	PE_{43}
4	$SV_{13} \leftarrow EV_{13} + x(1)*W^4 = x(0) + x(1)*W^4$			
5		$SV_{23} \leftarrow EV_{23} + x(3)*W^2 = x(0) + x(1)*W^4 + x(3)*W^2$		
6			$SV_{33} \leftarrow EV_{33} + x(4)*W^1 = x(0) + x(1)*W^4 + x(3)*W^2 + x(4)*W^1$	
7				$SV_{43} \leftarrow EV_{43} + x(2)*W^3 = x(0) + x(1)*W^4 + x(2)*W^3 + x(3)*W^2 + x(4)*W^1 = X(4)$

Colonne 4

Etape	PE_{14}	PE_{24}	PE_{34}	PE_{44}
5	$SV_{14} \leftarrow EV_{14} + x(1)*W^3 = x(0) + x(1)*W^3$			
6		$SV_{24} \leftarrow EV_{24} + x(3)*W^4 = x(0) + x(1)*W^3 + x(3)*W^4$		
7			$SV_{34} \leftarrow EV_{34} + x(4)*W^1 = x(0) + x(1)*W^3 + x(3)*W^4 + x(4)*W^2$	
8				$SV_{44} \leftarrow EV_{44} + x(2)*W^1 = x(0) + x(1)*W^3 + x(2)*W^1 + x(3)*W^4 + x(4)*W^2 = X(3)$

Tableaux III.11. Evolution des calculs sur chaque PE du 2ème réseau systolique planaire proposé pour le calcul de la TFD.

Le tableau suivant donne une comparaison en termes de multiplieurs réels, additionneurs réels et éléments de retard entre les deux réseaux planaires présentés ci-dessus.

On remarque que le 2ème réseau est légèrement meilleur en termes de nombre d'opérateurs réels que le 1er.

	Multiplieurs réels	Additionneurs réels	Eléments de retard D
Réseau 1	$4N(N\text{-}1)$	$4N(N\text{-}1)$	$(2N+1)(N\text{-}1)$
Réseau 2	$4(N\text{-}1)(N\text{-}1)$	$2(N\text{-}1)(2N\text{-}1)$	$2N(N\text{-}1)$

Tableau III.12. Comparaison en termes de nombre d'opérateurs réels et d'éléments de retard entre les deux réseaux planaires proposés.

10. Conclusion

Dans ce chapitre nous avons présenté et étudié les réseaux systoliques qui grâce à leurs très intéressantes propriétés de régularité, localité et modularité ont attiré les chercheurs et concepteurs pour les utiliser dans la réalisation de structures de calcul VLSI. Ils peuvent en effet être adéquatement adaptés pour l'implantation de circuits dédiés à différents algorithmes de traitement de signal comme la transformée de Fourier discrète ou le filtrage numérique. Le pipelining utilisé aussi bien dans la transmission des données que dans le traitement de ces mêmes données permet à ces circuits dédiés d'avoir une très grande puissance de calcul qui sera notamment remarquable lorsque le volume de données à traiter est important.

Les exemples donnés dans ce chapitre, concernant l'évaluation d'un polynôme suivant le schéma de Horner puis le produit matriciel, ont permis de déduire plusieurs réseaux systoliques linéaires et planaires permettant l'implémentation et l'évaluation de la TFD.

L'implémentation des filtres numériques RII sur des réseaux systoliques très performants sera le sujet du prochain chapitre.

Chapitre IV

Implémentation des filtres récursifs RII sur des architectures parallèles très performantes

1. Introduction

Dans le domaine du traitement du signal, on est couramment amené à modifier les composantes spectrales d'un signal donné : c'est le rôle consacré aux filtres. Deux types de filtres peuvent être utilisés : les filtres analogiques et les filtres numériques. Ces derniers, contrairement aux filtres analogiques qui sont réalisés à l'aide d'un agencement de composants physiques (transistors, résistances, condensateurs, inductances , etc.), sont réalisés soit par logiciel sur un ordinateur à usage général, soit par des processeurs programmables (microcontrôleurs, DSPs, FPGAs, etc.), soit par des circuits intégrés dédiés. Cependant, bien que la réalisation d'un traitement sous forme analogique soit plus rapide que par une forme numérique, l'avancée technologique et économique dans la réalisation des circuits numériques a fait que la plupart des filtres utilisés de nos jours sont numériques. En plus de cela et au delà de la réalisation des filtres standards passe-bas, passe-haut, passe-bande et coupe-bande, on peut réaliser numériquement n'importe quel type de filtre multi-bandes.

Donc, comme pour un filtre analogique, un filtre numérique a pour but de modifier les caractéristiques fréquentielles d'un signal donné présent à son entrée. C'est un système qui reçoit une séquence d'entrée $\{x_n\}$ et restitue une séquence de sortie $\{y_n\}$. Ses applications sont diverses et on peut citer entre autres la sélection d'une gamme de fréquences, l'atténuation de certaines fréquences pour extraire une information utile, l'amélioration de la qualité d'un signal par l'élimination du bruit ou l'adaptation d'un signal à une application donnée.

Il y a deux grandes familles de filtres numériques :

- Les filtres à Réponse Impulsionnelle Finie appelés filtres RIF (en anglais filters FIR : pour Finite Impulse Response). Ce type de filtre est dit fini, car sa réponse impulsionnelle se stabilisera ultimement à zéro. Un filtre FIR est non récursif, c'est-à-dire que la sortie dépend uniquement de l'entrée du signal (Il ne possède pas une boucle de contre-réaction).
- Les filtres à Réponse Impulsionnelle Infinie appelés filtres RII (en anglais filters IIR *:* Infinite Impulse Response), possèdent une réponse impulsionnelle qui ne s'annule jamais définitivement ou qui converge éventuellement vers zéro à l'infini. Ce type de filtre est récursif, c'est-à-dire que la sortie du filtre dépend à la fois du signal d'entrée et du signal de sortie (Il possède une boucle de contre-réaction). Les filtres IIR sont principalement la version numérique des filtres analogiques traditionnels : Butterworth, Tchebychev, Bessel, Elliptique. Bien que la complexité d'un filtre RII soit supérieure à celle d'un filtre RIF du même ordre, les filtres RII ont l'avantage d'être plus sélectifs.

Plusieurs travaux ont abordé la synthèse et l'implémentation des filtres sur des réseaux parallèles ou systoliques [34 – 36, 146 – 158]. Dans ce chapitre, nous allons implémenter des filtres RII sur différentes architectures parallèles et systoliques performantes.

2. Définition d'un filtre RII

Un filtre RII d'ordre *N* peut être défini :

- par une équation aux différences définie comme suit [159, 160] :

$$y(n) = \sum_{i=0}^{N} b_i \cdot e(n-i) - \sum_{i=0}^{N} a_i \cdot y(n-i) \qquad \text{(IV.1)}$$

- par sa fonction de transfert :

$$H(z) = \frac{Y(z)}{E(z)} = \frac{\sum_{i=0}^{N} b_i \cdot z^{-i}}{1 + \sum_{i=0}^{N} a_i \cdot z^{-i}} \qquad \text{(IV.2)}$$

- ou par sa représentation d'état sous forme matricielle condensée donnée par :

$$\begin{bmatrix} x(n+1) \\ y(n) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x(n) \\ e(n) \end{bmatrix} \qquad \text{(IV.3)}$$

Avec $x(n) \in \Re^N$; $y(n) \in \Re$; et $e(n) \in \Re$

- A, B, C, et D sont les matrices d'état de dimensions *N*x*N*, *N*x1, 1x*N*, et 1x1 respectivement,
- $x(n)$ est l'état présent du filtre,
- $x(n+1)$ est l'état prochain ou futur du filtre,
- $e(n)$ est le énième échantillon du signal d'entrée,
- $y(n)$ représente la énième réponse du système à l'entrée $e(n)$.

La représentation d'état sous forme matricielle du filtre peut être aisément dérivée à partir de sa fonction de transfert. A cet effet, la fonction **`tf2ss`** (transfer function to space state) de Matlab permet de calculer les matrices A, B, C et D.

3. Implémentation d'un filtre RII 1D sur un réseau systolique conventionnel

Comme mentionné dans l'introduction de ce chapitre, ce qui nous intéresse dans ce travail est l'implémentation d'un filtre RII 1D sur un réseau systolique. D'après la représentation d'état donnée par (IV.3), l'implémentation d'un filtre RII 1D n'est donc autre que l'implémentation du produit de la matrice type $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$ par le vecteur $\begin{bmatrix} x(n) \\ e(n) \end{bmatrix}$

Pour illustrer ce principe, nous allons détailler l'exemple de l'implantation d'un filtre RII 1D d'ordre 3. La représentation d'état sous forme matricielle d'un tel filtre est donnée par :

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \\ x_3(n+1) \\ y(n) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \\ e(n) \end{bmatrix}$$

Avec

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad ; \quad B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad ; \quad C = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \quad ; \quad D = d$$

Le réseau systolique conventionnel de la figure IV.1 permettant le calcul du produit d'une matrice carrée (N+1)x(N+1) par un vecteur colonne (N+1)x1 (avec N=3 : ordre du filtre) réalise ainsi l'opération de filtrage [161 – 162]. Dans ce type de structure, les entrées du réseau sont utilisées pour charger le signal d'entrée et l'état initial du filtre, et les sorties sont utilisées pour produire le signal de sortie et l'état suivant du système. Il faut noter aussi que les cellules élémentaires PE*ij* sont chargées préalablement par les éléments a_{ij}, b_i, c_i, *et* d des matrices A, B, C, et D respectivement selon la représentation de la figure IV.1.

Les figures IV.2.a…g illustrent le principe de fonctionnement du réseau systolique de la figure IV.1. et montrent les différentes étapes de calcul nécessaires pour obtenir un échantillon *y(n)* à la sortie du filtre.

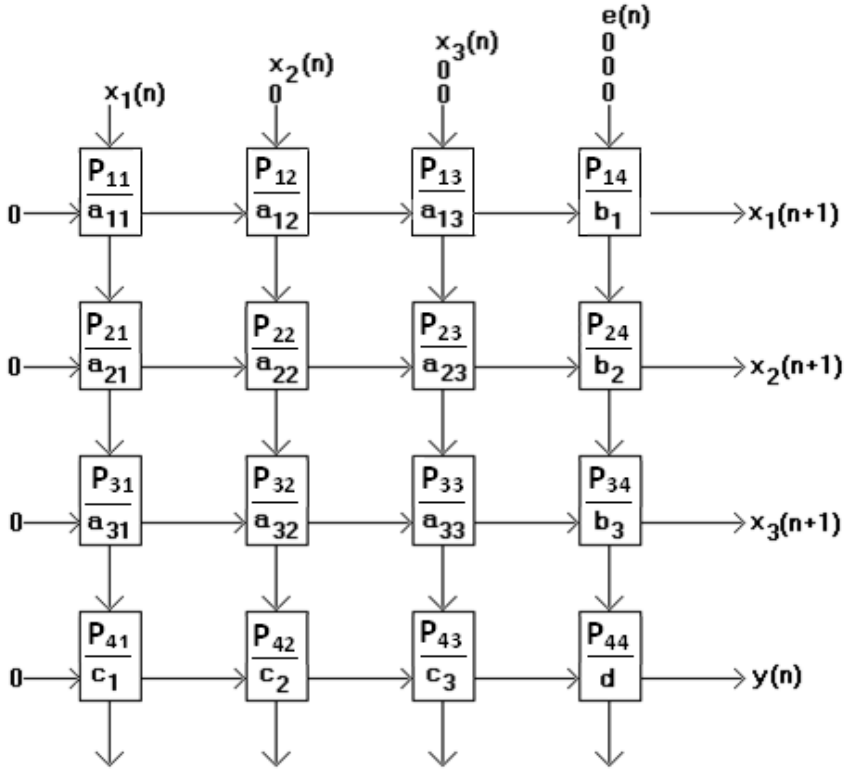


Figure IV.1. Réseau systolique conventionnel d'un filtre RII du 3ème ordre.

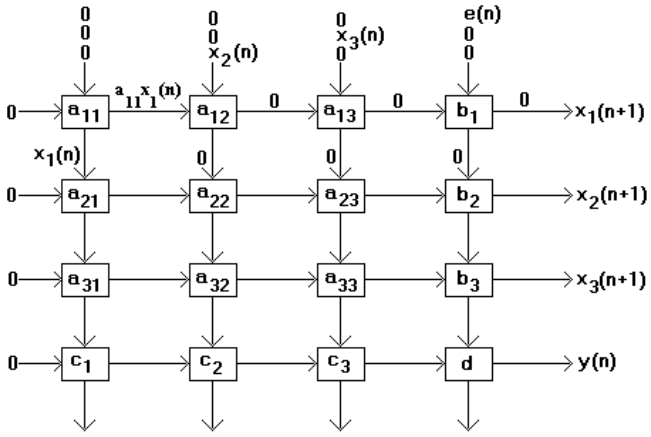


Fig. IV.2.a - Etape 1

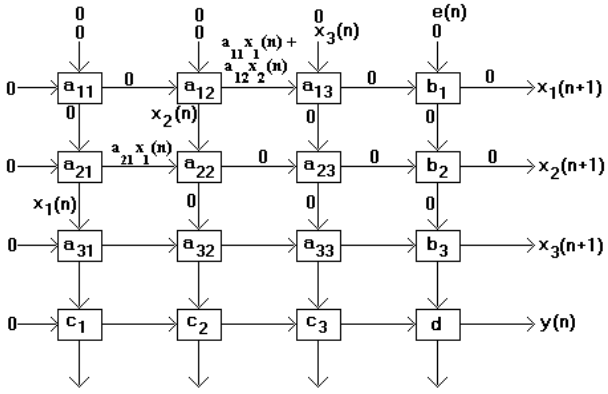


Fig. IV.2.b - Etape 2

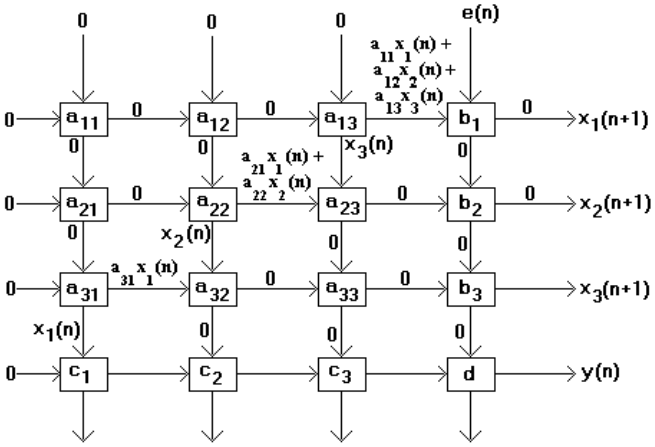


Fig. IV.2.c - Etape 3

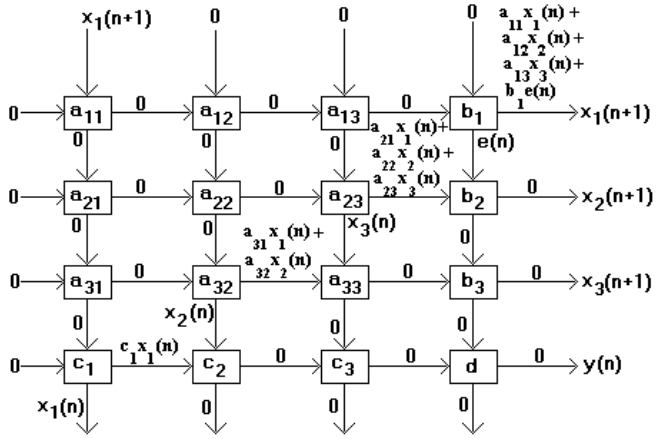


Fig. IV.2.d - Etape 4

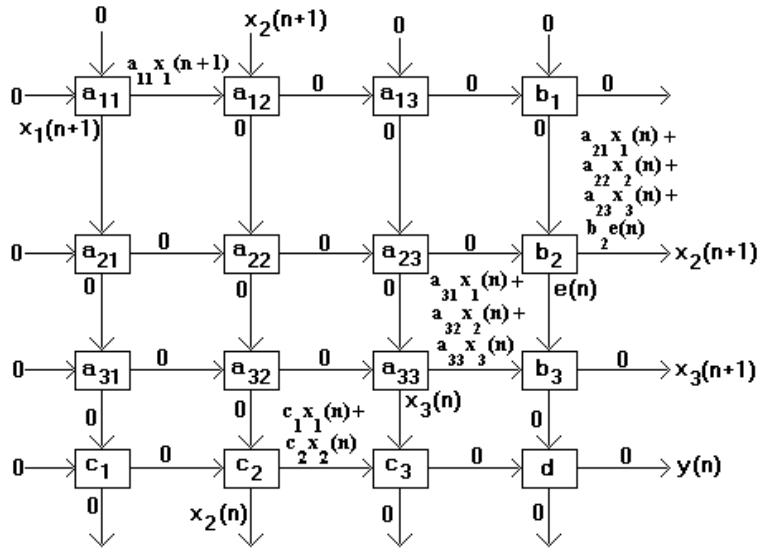


Fig. IV.2.e - Etape 5

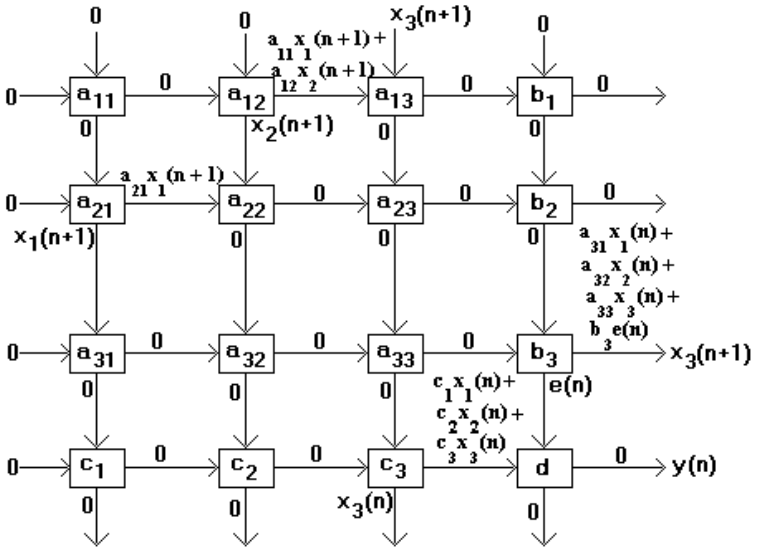


Fig. IV.2.f - Etape 6

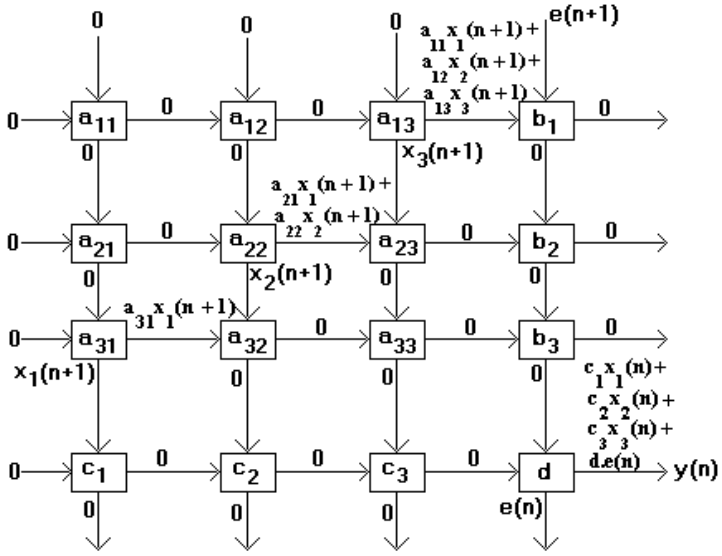


Fig. IV.2.g - Etape 7

Le tableau IV.1 résume clairement l'évolution des calculs sur chaque PEij pendant les 7 premières étapes du réseau systolique conventionnel du filtre de la figure IV.1.

Colonne 1

Etape	PE_{11}	PE_{21}	PE_{31}	PE_{41}
1	$SH_{11}= a_{11}x_1(n)$			
2		$SH_{21}= a_{21}x_1(n)$		
3			$SH_{31}= a_{31}x_1(n)$	
4				$SH_{41}= c_1x_1(n)$
5	$SH_{11}= a_{11}x_1(n+1)$			
6		$SH_{21}= a_{21}x_1(n+1)$		
7			$SH_{31}= a_{31}x_1(n+1)$	

Colonne 2

Etape	PE_{12}	PE_{22}	PE_{32}	PE_{42}
2	$SH_{12}= a_{11}x_1(n)+ a_{12}x_2(n)$			
3		$SH_{22}= a_{21}x_1(n) + a_{22}x_2(n)$		
4			$SH_{32}= a_{31}x_1(n) + a_{32}x_2(n)$	
5				$SH_{42}= c_1x_1(n) + c_2x_2(n)$
6	$SH_{12}= a_{11}x_1(n+1)+ a_{12}x_2(n+1)$			
7		$SH_{22}= a_{21}x_1(n+1) + a_{22}x_2(n+1)$		

Colonne 3

Etape	PE_{13}	PE_{23}	PE_{33}	PE_{43}
3	$SH_{13}= a_{11}x_1(n)+ a_{12}x_2(n) + a_{13}x_3(n)$			
4		$SH_{23}= a_{21}x_1(n) + a_{22}x_2(n) + a_{23}x_3(n)$		
5			$SH_{33}= a_{31}x_1(n) + a_{32}x_2(n) + a_{33}x_3(n)$	
6				$SH_{43}= c_1x_1(n) + c_2x_2(n) + c_3x_3(n)$
7	$SH_{13}= a_{11}x_1(n+1)+ a_{12}x_2(n+1) + a_{13}x_3(n+1)$			

Colonne 4

Eta	PE_{14}	PE_{24}	PE_{34}	PE_{44}
4	$SH_{14}= a_{11}x_1(n)+ a_{12}x_2(n) + a_{13}x_3(n) + b_1e(n)= x_1(n+1)$			
5		$SH_{24}= a_{21}x_1(n) + a_{22}x_2(n) + a_{23}x_3(n) + b_2e(n)= x_2(n+1)$		
6			$SH_{33}= a_{34}x_1(n) + a_{32}x_2(n) + a_{33}x_3(n) + b_3e(n)= x_3(n+1)$	
7				$SH_{44}= c_1x_1(n) + c_2x_2(n) + c_3x_3(n) + de(n)= y(n)$

Tableaux IV.1. Evolution des calculs sur chaque PE du réseau systolique conventionnel d'un filtre RII du 3ème ordre

La cellule ou processeur élémentaire (PE) utilisé dans ce type de réseau n'est autre que celui utilisé pour le produit d'une matrice par un vecteur et est donné sur les figures IV.3.a et b.

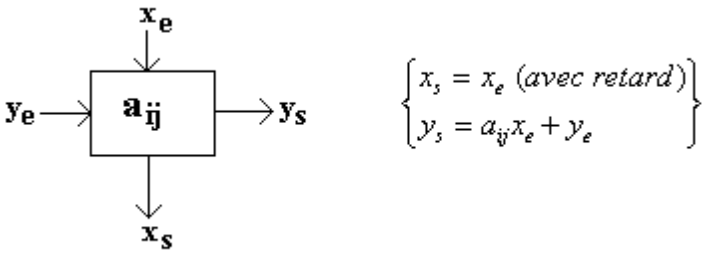


Fig. IV.3.a. PE du réseau systolique conventionnel d'un filtre RII.

Ce PE est constitué d'un multiplieur/additionneur (MAC : Multiplication + Accumulation), d'un registre à décalage et d'un élément de retard. Le multiplieur additionneur assure le calcul de $y_s = a_{ij}x_e + y_e$. Le registre à décalage permet la mémorisation des coefficients *aij*, alors que l'élément de retard permet la transmission de l'entrée x_e verticalement après un retard d'une unité de temps.

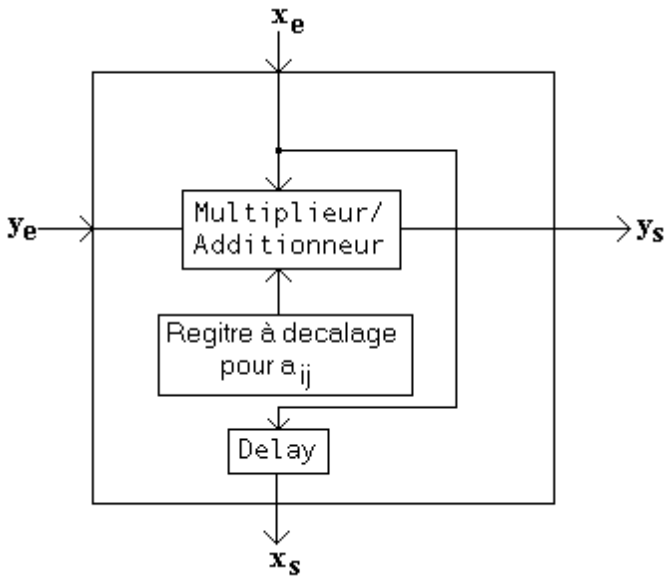


Fig. IV.3.b. Structure interne du PE du réseau systolique conventionnel d'un filtre RII.

Le nombre d'étapes de calcul nécessaires pour obtenir un échantillon *y(n)* à la sortie d'un filtre RII 1D d'ordre *N* sur un réseau systolique conventionnel de dimension (*N*+1)x(*N*+1) est de (2*N*+1). Les sorties *y*(*n*+i) i=1,2,… sont obtenues toutes les *N*+1 étapes.

Le débit en données est estimé à $\frac{1}{(N+1)(m+l)}$ où m et l sont les temps nécessaires pour effectuer une multiplication et une addition respectivement [35, 36, 163].

Il a été montré qu'on peut augmenter le débit en données de ces réseaux systoliques et par conséquent résoudre le problème de la latence introduite par la récursivité de l'algorithme de filtrage RII, en utilisant des structures cylindriques associées avec des matrices d'état multidiagonales [35, 36, 163, 164].

4. Le principe de l'architecture cylindrique

Une architecture cylindrique est définie comme un réseau de PEs placés sur la surface d'un cylindre d'une manière régulière. Pour bien éclaircir ce principe, prenons l'exemple du produit d'une matrice carrée A (2x2) par un vecteur X (2x1), avec :

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad et \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Les schémas résumant les fonctionnements d'une cellule élémentaire ainsi que celui du réseau cylindrique sont donnés sur la figure IV.4.

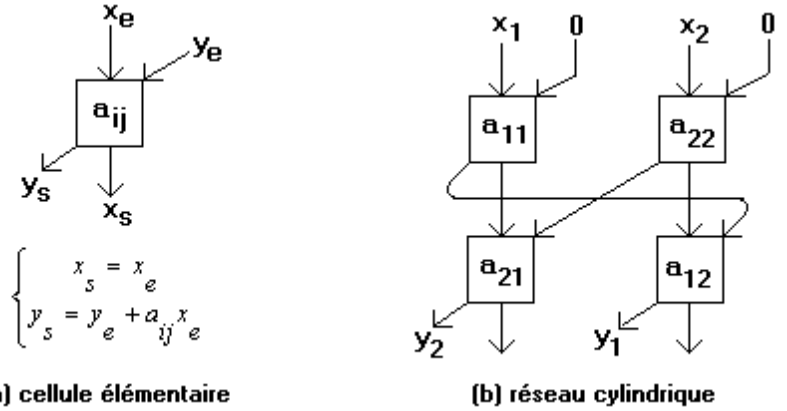


Fig. IV.4. Structure cylindrique du produit d'une matrice (2x2) par un vecteur (2x1)

Chaque cellule élémentaire ou PE possède 2 entrées et 2 sorties, verticales et transversales (Fig. IV.4.a).

Les éléments de la matrice A sont enregistrés dans les mémoires des PEs et les éléments de du vecteur X sont transmis verticalement au réseau en même temps. Chaque PE multiplie l'entrée verticale avec l'élément a_{ij} stocké dans son registre interne, ajoute ce produit à l'entrée transversale, et le résultat ainsi obtenu est transmis transversalement.

Les sorties y_1 et y_2 sont obtenues après 2 étapes de calcul. En général le nombre d'étapes de calcul nécessaires pour effectuer la multiplication d'une matrice (NxN) par un

vecteur (Nx1) sur un réseau de type cylindrique (*N*x*N*) est de *N* étapes [165]. Ce type de structure est plus rapide que celui adopté pour un réseau systolique conventionnel de même dimension (*N*x*N*) qui nécessite lui 2*N*-1 étapes.

5. Implémentation des filtres RII 1D sur les réseaux cylindriques

Comme vu plus haut, un filtre récursif 1D (d'ordre N) représenté par la fonction matricielle d'état donnée par (IV.3), est réalisé par le produit d'une matrice (*N*+1,*N*+1) par un vecteur (*N*+1,1). On pourra l'adapter pour l'implémenter sur un réseau systolique cylindrique. Pour illustrer cela, reprenons l'exemple d'un filtre RII 1D d'ordre 3. On aura alors besoin de 3 types de cellules [35, 36]: des cellules d'entrée, des cellules internes et des cellules de sortie. Ces cellules avec leurs modes de fonctionnements et leurs structures internes sont donnés sur les figures IV.5.a…c.

5.1. Les cellules d'entrée

Chaque cellule d'entrée du réseau est constituée d'un commutateur qui permet de commuter entre l'état initial (n=0) du réseau c'est à dire à la première récursion de calcul et l'état à l'instant n ≠0, c'est à dire au début de la deuxième récursion après avoir obtenu le *y(n)* et les *xi(n+1).* Elle est constituée aussi d'un registre à décalage pour la mémorisation des coefficients αij, d'un multiplieur/additionneur pour le calcul de $y_s = \alpha_{ij}(x_e + y_e)$, et d'un élément de retard qui permet la transmission de l'entrée x_e ou y_e verticalement après un retard d'une unité de temps.

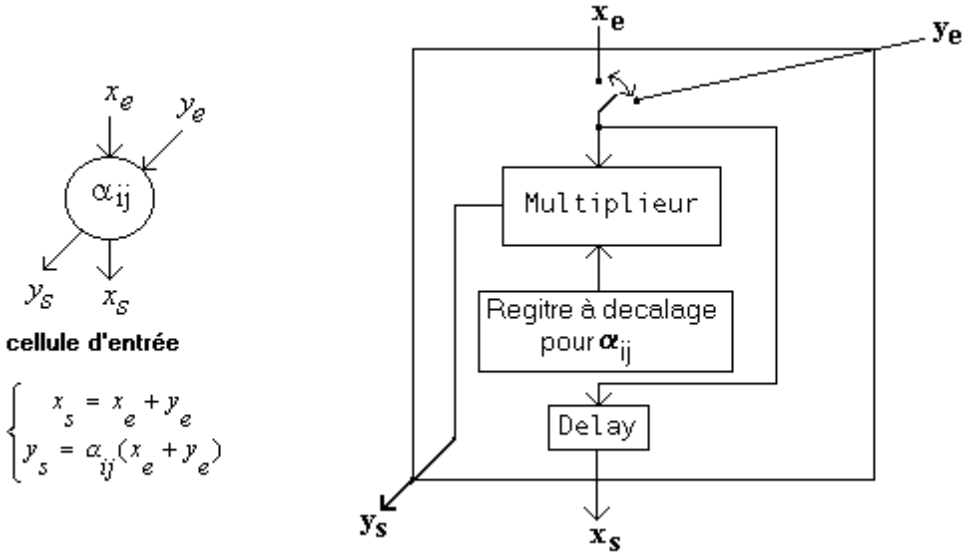


Fig. IV.5.a. Fonctionnement et structure interne de la cellule d'entrée pour le réseau cylindrique de processeurs pour les filtres RII 1D

5.2. Les cellules internes

Chaque cellule interne du réseau est constituée d'un registre à décalage pour la mémorisation des coefficients αij, d'un multiplieur/additionneur pour le calcul de $y_s = \alpha_{ij} x_e + y_e$, et d'un élément de retard pour la transmission de l'entrée x_e verticalement avec un retard d'une unité de temps.

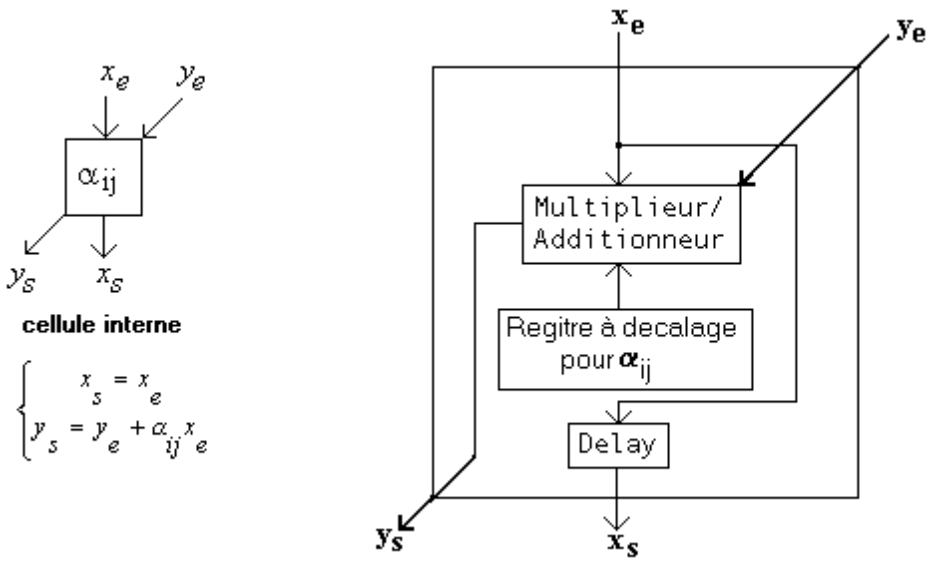


Fig. IV.5.b. Structure interne de la cellule interne pour le réseau cylindrique de processeurs pour les
filtres RII 1D

5.3. Les cellules de sortie

Chaque cellule de sortie du réseau est constituée d'un registre à décalage pour la mémorisation des coefficients αij, et d'un multiplieur/additionneur pour le calcul de $y_s = \alpha_{ij} x_e + y_e$.

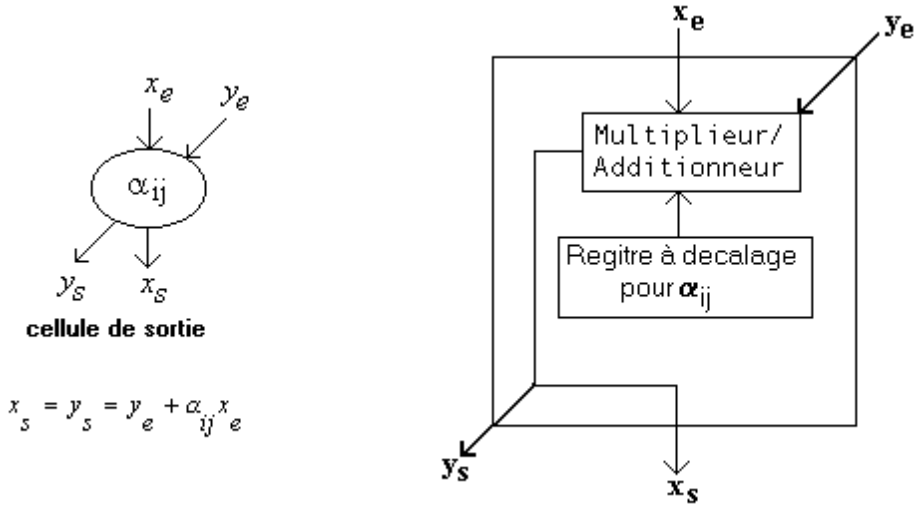


Fig. IV.5.c. Structure interne de la cellule de sortie pour le réseau cylindrique de processeurs pour les
filtres RII 1D

La figure IV.6 donne le schéma du réseau systolique cylindrique pour l'implémentation d'un filtre RII 1D d'ordre 3. Le fonctionnement de ce réseau est supposé synchrone. Les éléments des matrices A, B, C et D sont chargés initialement dans les mémoires des PEs suivant la disposition spécifiée sur la figure IV.6.

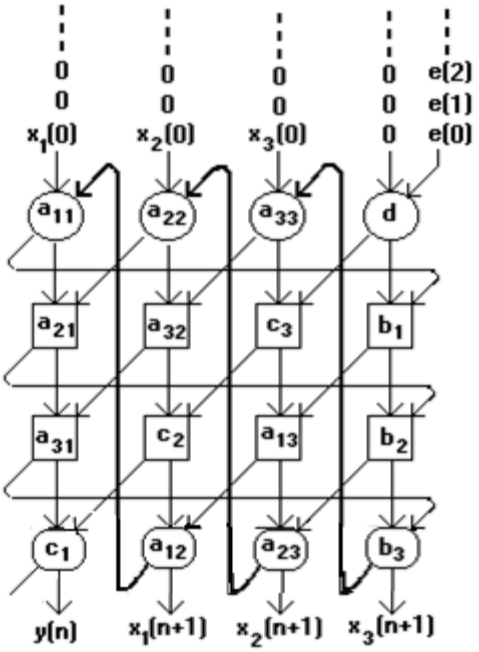


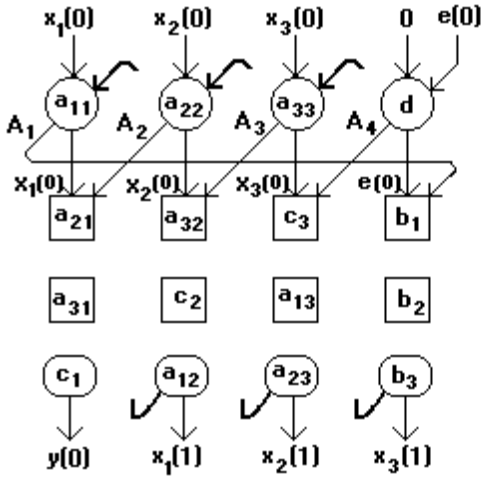
Fig. IV.6. Réseau systolique cylindrique d'un filtre RII 1D du 3ème ordre.

- Les cellules d'entrée permettent, dans un premier temps le chargement des états initiaux $x_i(0)$ du filtre et l'entrée $e(0)$, et ensuite les états suivants $x_i\,(n+1)$ et les entrées $e(n+1)$. Ces états initiaux et suivants seront transmis verticalement, de haut en bas, aux cellules inférieures après chaque étape de calcul (Fig. IV.7.a…d). Les cellules d'entrée permettent aussi de calculer les 1^{ers} termes de $x_i\,(n+1)$ et de $y(n)$ suivant les formules données sur la figure IV.5.a, et de les transmettre transversalement aux cellules internes placées juste au-dessous d'elles.

- Les cellules internes permettent le calcul des termes suivants de $x_i\,(n+1)$ et de $y(n)$ suivant les formules données sur la figure IV.5.b. et leur transmission suivant la même procédure vers les cellules suivantes.

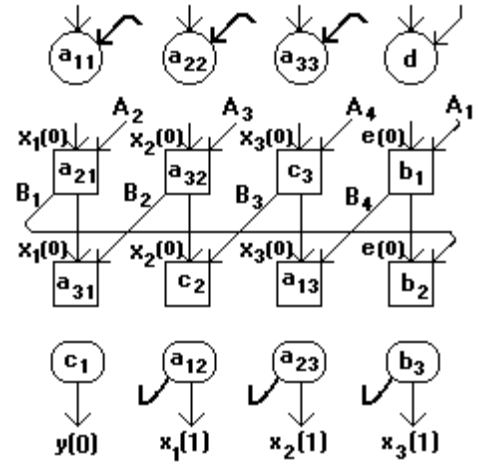
- Les cellules de sortie calculent les derniers termes de $x_i\,(n+1)$ et de $y(n)$ suivant les formules données sur la figure IV.5.c. La sortie $y(n)$ est donc disponible sur la cellule de gauche et les états suivants $x_i\,(n+1)$ seront transmis transversalement vers les cellules d'entrée pour les prochaines étapes de calcul qui détermineront l'échantillon suivant $y(n+1)$.

Les étapes de calcul pour l'implémentation d'un filtre RII 1D d'ordre 3 en utilisant le réseau cylindrique donné par la figure IV.6 sont schématisées sur les figures IV.7.a…d.



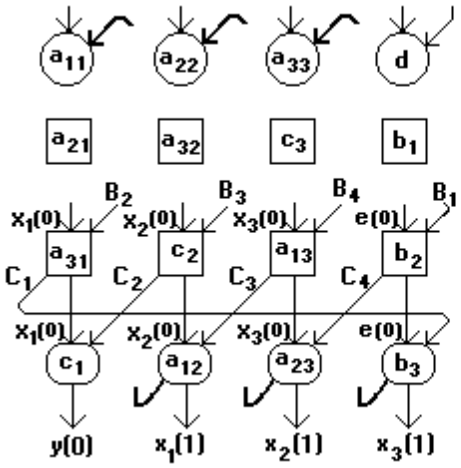
$$A_1 = a_{11}x_1(0)$$
$$A_2 = a_{22}x_2(0)$$
$$A_3 = a_{33}x_3(0)$$
$$A_4 = de(0)$$

Fig. IV.7.a. Etape 1



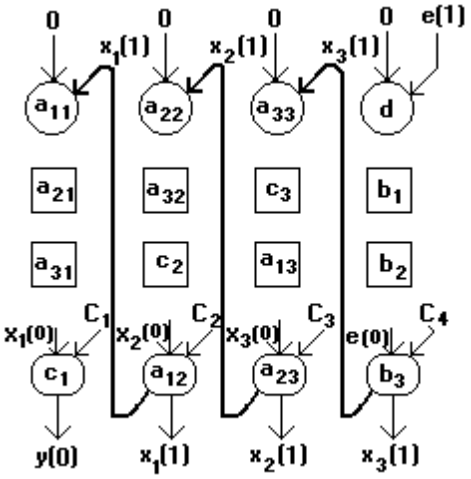
$$B_1 = a_{21}x_1(0) + a_{22}x_2(0)$$
$$B_2 = a_{32}x_2(0) + a_{33}x_3(0)$$
$$B_3 = c_3x_3(0) + de(0)$$
$$B_4 = a_{11}x_1(0) + b_1e(0)$$

Fig. IV.7.b. Etape 2



$$C_1 = a_{31}x_1(0) + a_{32}x_2(0) + a_{33}x_3(0)$$
$$C_2 = c_2x_2(0) + c_3x_3(0) + de(0)$$
$$C_3 = a_{11}x_1(0) + a_{13}x_3(0) + b_1e(0)$$
$$C_4 = a_{21}x_1(0) + a_{22}x_2(0) + b_2e(0)$$

Fig. IV.7.c. Etape 3



$$\mathbf{y(0)} = c_1 x_1(0) + c_2 x_2(0) + c_3 x_3(0) + d\,e(0)$$
$$x_1(1) = a_{11} x_1(0) + a_{12} x_2(0) + a_{13} x_3(0) + b_1 e(0)$$
$$x_2(1) = a_{21} x_1(0) + a_{22} x_2(0) + a_{23} x_3(0) + b_2 e(0)$$
$$x_3(1) = a_{31} x_1(0) + a_{32} x_2(0) + a_{33} x_3(0) + b_3 e(0)$$

Fig. IV.7.d. Etape 4

La première sortie *y(0)* et le nouvel état du système { $x_1(1)$, $x_2(1)$, $x_3(1)$ } ne sont délivrés qu'après 4 étapes de calcul. En général et pour un filtre d'ordre *N*, il faudra (*N*+1) étapes de calcul pour obtenir une sortie *y(n)* sur le réseau cylindrique (*N*x*N*) associé. La vitesse de calcul du réseau est donc fonction de l'ordre du filtre. Le débit en données est estimé à $\frac{\mathbf{1}}{(N+\mathbf{1})(m+l)}$ où *m* et *l* sont les temps nécessaires pour effectuer une multiplication et une addition respectivement. Comme mentionné dans le chapitre précédent, le débit en données peut être augmenté en utilisant des structures cylindriques associées avec des matrices d'état multidiagonales [35, 36].

5.4. Implémentation d'un filtre RII 1D sur un réseau cylindrique représenté par une matrice creuse (ou multidiagonale)

L'implémentation des filtres RII 1D sur un réseau systolique cylindrique représenté par une matrice multidiagonale permet d'augmenter le débit en données qui sera égal à $\frac{\mathbf{1}}{n(m+l)}$ (où *n* est le nombre de diagonales non nulles dans la matrice), et cela quelque soit l'ordre du filtre [35, 36]. Elle permet aussi de réduire le nombre de PEs dans le réseau, et par conséquent d'augmenter la vitesse de calcul et de diminuer la complexité de mise en œuvre.

Le principe et les étapes de calcul resteront les mêmes que ceux expliqués précédemment. Le fait nouveau est que les matrices A, B, C et D seront, dans le cas général, de la forme:

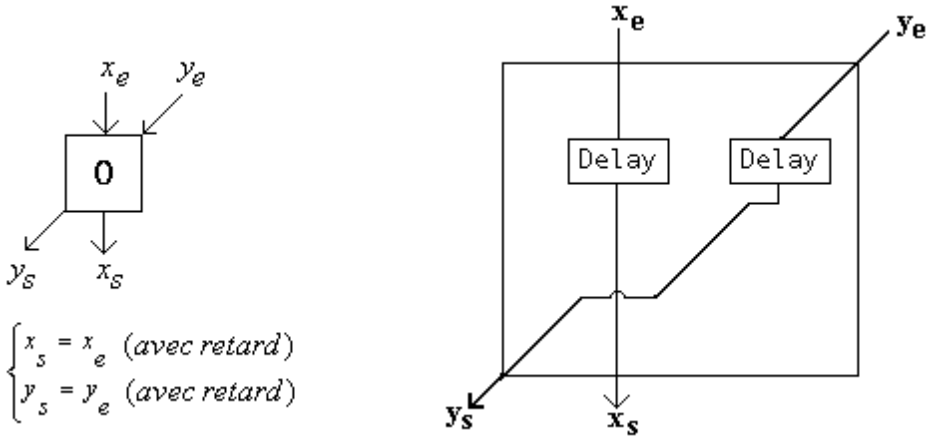
$$A=\begin{bmatrix} n_0 & p_1 & \mathbf{0} & \mathbf{0} & . & . & \mathbf{0} \\ m_1 & n_1 & p_2 & \mathbf{0} & \mathbf{0} & . & . \\ \mathbf{0} & m_2 & n_2 & p_3 & \mathbf{0} & \mathbf{0} & . \\ \mathbf{0} & \mathbf{0} & m_3 & n_3 & . & \mathbf{0} & \mathbf{0} \\ . & . & \mathbf{0} & . & . & . & \mathbf{0} \\ \mathbf{0} & . & . & \mathbf{0} & . & . & p_{N-1} \\ \mathbf{0} & \mathbf{0} & . & . & \mathbf{0} & m_{N-1} & n_{N-1} \end{bmatrix} \quad ; \quad B=\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ . \\ . \\ . \\ \mathbf{0} \\ b \end{bmatrix} \quad ; \quad C=\begin{bmatrix} \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & c \end{bmatrix} \quad ; \quad D=d$$

Et dans le cas de l'exemple introduit précédemment (filtre RII 1D d'ordre 3), les matrices seront de la forme:

$$A=\begin{bmatrix} n_0 & \mathbf{0} & \mathbf{0} \\ m_1 & n_1 & \mathbf{0} \\ \mathbf{0} & m_2 & n_2 \end{bmatrix} \quad ; \quad B=\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ b \end{bmatrix} \quad ; \quad C=\begin{bmatrix} \mathbf{0} & \mathbf{0} & c \end{bmatrix} \quad ; \quad D=d$$

Cela permet de remplacer un bon nombre de cellules de la structure précédente par des cellules de type 0.

Fig. IV.8. Structure interne de la cellule de type 0



6. Application de la technique CTP à l'algorithme de filtrage RII 1D

Il a été montré dans [35] que la matrice d'état multidiagonale n'est pas toujours facile à obtenir et qu'elle dépend entièrement des conditions et des algorithmes d'optimisation utilisés. Une solution plus générale est donc proposée au problème de la latence en faisant appel à la technique CTP (Concurrent triple product) [166]. Cette technique permet d'obtenir des architectures cylindriques plus rapides. Elle consiste à transformer le produit d'une matrice par un vecteur $v = H.u$ où

$$v=\begin{bmatrix} x(n+1) \\ y(n) \end{bmatrix} \qquad H=\begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad u=\begin{bmatrix} x(n) \\ e(n) \end{bmatrix}$$

en un produit de trois matrices V=L.U.R où H=LoR (produit tensoriel). U et V sont des matrices obtenues à partir des vecteurs respectifs u et v [35, 167].

Si on considère le filtre RII 1D du 3ème ordre (*N*-1=3), donc N=4=2x2=pq; (p=q=2),

et :

$$H=\begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ c_1 & c_2 & c_3 & d \end{bmatrix} \quad ; \quad v=\begin{bmatrix} x_1(n+\mathbf{1}) \\ x_2(n+\mathbf{1}) \\ x_3(n+\mathbf{1}) \\ y(n) \end{bmatrix} \quad ; \quad u=\begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \\ e(n) \end{bmatrix}$$

La décomposition à terme unique de H (H=LoR) est définie par les deux matrices (2x2) L et R, telles que:

$$L=\begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix} \qquad R=\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$$

Les matrices U et V sont obtenues par un réarrangement des termes des vecteurs u et v respectivement, telles que:

$$U=\begin{bmatrix} x_1(n) & x_3(n) \\ x_2(n) & e(n) \end{bmatrix} \qquad V=\begin{bmatrix} x_1(n+1) & x_3(n+1) \\ x_2(n+1) & y(n) \end{bmatrix} \qquad avec\ V=L.U.R$$

Le calcul se fait en deux phases (figures IV.9.a…e):

- Durant la 1ère phase s'effectue le calcul des coefficients du produit matriciel LU de la même façon que celui dans les structures cylindriques.
- Durant la deuxième phase les cellules du réseau commutent c'est à dire changent de fonction et cela pour le calcul du produit de la matrice LU par la matrice R.

Les éléments de la matrice L sont stockés dans les mémoires locales (représentées par des triangles sur les figures IV.9.a…e), et celles de la matrice U sont transmis de haut en bas.

Pendant la 1ère phase chaque nœud ou cellule effectue la multiplication de l'entrée verticale par le scalaire stocké. Le produit ainsi obtenu est additionné à l'entrée arrivant transversalement, et le tout est retransmis par le chemin transversal. La séquence verticale est transmise sans modification. Au bout de l'étape p (dans notre cas p=2) la première colonne de la matrice LU est calculée par les nœuds du bas.

Les colonnes de la matrice LU sont alors rebouclées sur les chemins transversaux de bas en haut, et les lignes de la matrice R suivent les colonnes de la matrice U sur les chemins verticaux. Avec ces nouvelles entrées, les cellules changent de fonction ou commutent. C'est la $2^{ème}$ phase de calcul. Chaque nœud retransmet les deux séquences d'entrée sans modification après avoir calculé leur produit qui sera mémorisé dans la mémoire de la cellule. La commutation dans le fonctionnement des nœuds se propage de haut en bas avec l'arrivée des éléments des matrices LU et R. Au bout de l'étape (p+q+1) toutes les composantes de la matrice V=LUR sont localisées dans les mémoires des nœuds.

Les différentes étapes de calcul sont schématisées sur les figures IV.9.a…e.

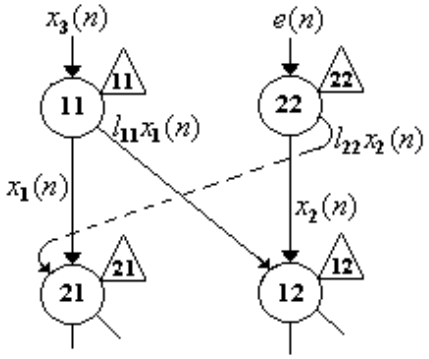
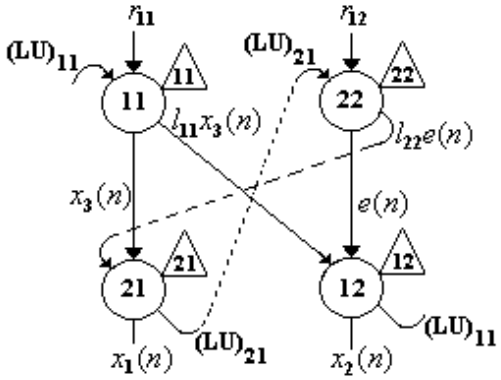
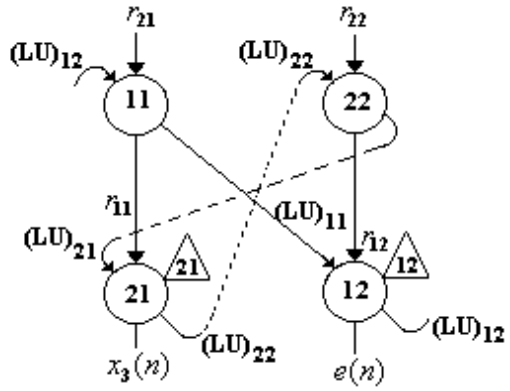


Fig. IV.9.a. Etape 1



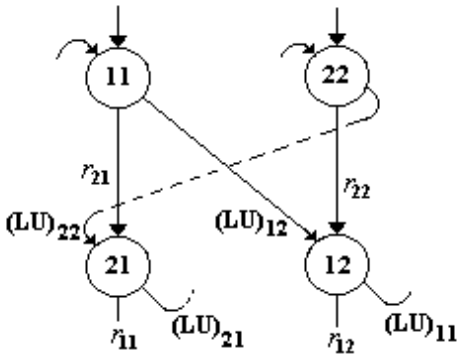
$$(LU)_{21} = l_{21}x_1(n) + l_{22}x_2(n)$$
$$(LU)_{11} = l_{12}x_2(n) + l_{11}x_1(n)$$

Fig. IV.9.b. Etape 2



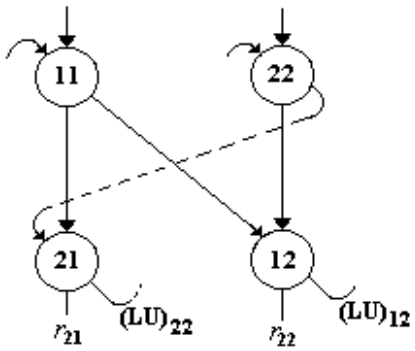
$$(LU)_{22} = l_{21}x_3(n) + l_{22}e(n)$$
$$(LU)_{12} = l_{12}e(n) + l_{11}x_3(n)$$

Fig. IV.9.c. Etape 3



$$V_{11} = (LU)_{11}r_{11} + (LU)_{12}r_{21}$$
$$V_{22} = (LU)_{21}r_{12} + (LU)_{22}r_{22}$$

Fig. IV.9.d. Etape 4



$$V_{21} = (LU)_{21} r_{11} + (LU)_{22} r_{21}$$
$$V_{12} = (LU)_{11} r_{12} + (LU)_{12} r_{22}$$

Fig. IV.9.e. Etape 5

Fig. IV.9. Principe de fonctionnement du réseau cylindrique dynamiquement reconfigurable d'un filtre RII 1D du 3ème ordre.

Le débit en données du réseau systolique obtenu par la technique CTP est estimé par [35]:

$$\frac{1}{(p+q+1)(m+l)}$$

7. Conclusion

Dans ce chapitre, nous avons proposé quelques implémentations très performantes des filtres RII 1D. L'implémentation directe des filtres RII 1D, représentés dans l'espace d'état, sur des réseaux systoliques conventionnels permet un débit en données appréciable. Une amélioration du point de vue vitesse de calcul et débit en données est apportée par l'utilisation des réseaux systoliques du type cylindrique sans pour autant augmenter la complexité de mise en œuvre des PEs. Une amélioration plus accentuée du débit en données est obtenue avec la représentation des filtres par des matrices creuses multidiagonales qui permettent aussi de réduire la complexité matérielle en diminuant le nombre de PEs.

Comme la matrice d'état multidiagonale n'est pas toujours facile à obtenir et qu'elle dépend entièrement des algorithmes d'optimisation utilisés, nous avons fait appel à la technique CTP qui permet d'obtenir des architectures cylindriques encore plus rapides avec une meilleure complexité de mise en œuvre.

Conclusion générale

Conclusion générale

L'objectif, que nous nous sommes fixé au départ, consistait à développer des architectures parallèles aussi performantes que possible de circuits dédiés au calcul de la transformée de Fourier discrète et au calcul des filtres récursifs IIR, deux outils incontournables dans le domaine du traitement numérique du signal.

Pour cela, trois algorithmes FFT, l'un applicable à une base arbitraire B et les deux autres aux bases 2 et 4, ont été tout d'abord développés. Les bases 2 et 4 ont été choisies parce quelles permettent une complexité relativement réduite au niveau des processeurs élémentaires (PEs) par rapport aux autres bases. Il nous a été permis par la suite de déduire trois types d'architectures que nous avons présentés dans le chapitre 2. La première architecture, conséquence directe des diagrammes de fluence qui reflètent graphiquement la structure même de l'algorithme, s'étale sur $\log_B(N)$ (B = 2 ou 4 pour les cas étudiés) rangées, chacune composée de respectivement (N/B) PEs. Elle permet le meilleur débit en données en présentant une latence de $\log_B(N)$ étapes pour le premier spectre et une latence d'une étape pour chacun des spectres suivants. Si le nombre d'échantillons à traiter s'avère très élevé, les contraintes technologiques actuelles pourraient ne pas permettre la réalisation d'une telle architecture. Dans un souci de réduire le nombre de PEs, deux alternatives ont été suggérées. La première consiste à ne laisser qu'une rangée de (N/B) PEs de la structure précédente et de reboucler convenablement les sorties de ceux-ci sur leurs entrées en ajoutant un système de multiplexeurs et démultiplexeurs. Quant à la deuxième alternative, elle permet de réduire drastiquement le nombre de PEs. En effet une seule ligne de $\log_B(N)$ PEs est utilisée. Un nombre de registres plus ou moins élevé sera en amont de chaque PE, suivant sa position dans la chaîne, pour introduire les retards nécessaires à une lecture adéquate des échantillons et des spectres intermédiaires par chacun des PEs. La réduction du nombre de PEs est réalisée au détriment de la latence qui est augmentée pour les deux dernières alternatives.

Il a été constaté qu'au sein d'un même type d'architectures le nombre d'étapes de calculs ainsi que celui des PEs par rangée diminuaient avec l'augmentation de la base B utilisée (la base 4 est meilleure que la base 2), mais qu'en contre partie le nombre d'opérateurs (multiplieurs et additionneurs) composant les PEs augmentait, engendrant ainsi une complexité plus poussée au niveau de ces derniers. Une diminution du nombre d'étapes et

de PEs joue en faveur de l'utilisation de bases supérieures à 4. Dans ce contexte, les perspectives que nous nous fixons sont l'utilisation des bases 8 et 16 pour développer des algorithmes qui aboutiraient à des architectures qui pourraient être encore plus performantes. Toutefois, bien que les architectures FFT qu'on a développées soient performantes, leur implantation VLSI pourrait poser problème vis-à-vis du facteur localité. Ce problème est résolu par l'utilisation des architectures systoliques. Nous avons alors présenté différents réseaux systoliques pour le calcul de la TFD. Bien que composés d'un nombre élevé de PEs, ces réseaux utilisent un parallélisme intensif dans les calculs, permettent un débit en données très élevé et disposent de tous les facteurs requis à une implantation VLSI.

Dans le dernier chapitre, réservé à l'implémentation des filtres RII 1D, nous avons commencé par une implémentation directe des filtres RII 1D, représentés dans l'espace d'état, sur les réseaux systoliques du type conventionnel puis nous avons présenté une implémentation sur des réseaux systoliques du type cylindrique dynamiquement reconfigurables. Les réseaux systoliques conventionnels souffrent d'une latence considérable, par contre les réseaux systoliques cylindriques dynamiquement reconfigurables, utilisant la technique de décomposition CTP, permettent de réduire ce problème de latence. Cela permet d'améliorer la vitesse de calcul, d'augmenter le flux de données en entrée (débit) et de réduire la complexité de mise en œuvre. Nous avons aussi étudié et proposé des structures internes pour les PEs utilisés par les différents réseaux processeurs qui pourront faire l'objet d'une implémentation hardware en VHDL avant de passer à leur réalisation sur silicium.

Bibliographie

Bibliographie

[1] J. G. Proakis and D. M. Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications," Prentice-Hall, Englewood Cliffs, New Jersey, 1996.

[2] Sen M. Kuo, Bob H. Lee, Wenshun Tian, "Real–Time Digital Signal Processing: Fundamentals, Implementations and Applications," Wiley , 2013.

[3] A. V. Oppenheim and R. W. Schafer, "Discrete-Time Signal Processing," Pearson New International Edition, 2013.

[4] William Hartmann, "Acoustic Signal Processing," Springer Handbook of Acoustics, pp. 503-530, 2007.

[5] F. Le Chevalier, "Principes de Traitement des Signaux Radar et Sonar," Masson, 1989.

[6] You He, Xin Guan, "Radar Data Processing With Applications," Wiley, 2016.

[7] Udo Zölzer, "Digital Audio Signal Processing," Wiley, 2008.

[8] Tony F. Chan and Jianhong J. Shen, "Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods," SIAM, 2005.

[9] Chi-Un Lei, Chung-Man Cheung, Ngai Wong, "Efficient 2D Linear-Phase IIR Filter Design and Application in Image Processing," In Advances in Communication Systems and Electrical Engineering. Lecture Notes in Electrical Engineering, vol 4, pp 411-424, Springer, Boston, MA, 2008.

[10] Andreas Leven, Noriaki Kaneda, Stephen Corteselli, "Real-Time Implementation of Digital Signal Processing for Coherent Optical Digital Communication Systems," IEEE Journal of Selected Topics in Quantum Electronics, vol 16, Issue 5, pp 1227 – 1234, 2010.

[11] D. Chikouche, N. Amardjia, N. Haloui, R. E. Bekka, "Application de la Méthode d'Analyse Spectrale AR et de la Méthode de Prony à la Détection des Défauts d'Engrenages : Comparaison de Performance," International Conference on Image and Signal Processing, Icisp'2003, pp. 729-736, Agadir - Morocco, June 25-27, 2003.

[12] E.S. Gopi, "Digital Signal Processing for Medical Imaging Using Matlab," Springer, 2013.

[13] F. Mayer-Lindenberg, "Dedicated Digital Processors: Methods in Hardware/Software Co-design," Wiley-Blackwell; 1 edition, 2003.

[14] Chin-Teng Lin, Yuan-Chu Yu, Lan-Da Van, "Cost-Effective Triple-Mode Reconfigurable Pipeline FFT/IFFT/2-D DCT Processor," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol 16, Issue 8, pp 1058 – 1071, 2008.

[15] Azad Fakhari, Mahmood Fathy, "A Two Level Architecture for High Throughput DCT-Processor and Implementing on FPGA," International Conference on Reconfigurable Computing and FPGAs, pp 115 – 120, 2010.

[16] Wei-Lung Yang, Hsi-Pin Ma, "A configurable wavelet processor for biomedical applications," VLSI Design, Automation and Test (VLSI-DAT), pp 1 – 4, 2015.

[17] Azadeh Safari, Niras C V, Yinan Kong, "VLSI architecture of multiplier-less DWT image processor," IEEE 2013 Tencon - Spring, pp 280 – 284, 2013.

[18] Muhsen Aljada, Kamal Alameh, "Reconfigurable multi-passband optical filter using Opto-VLSI processor," OECC/ACOFT 2008 - Joint Conference of the Opto-Electronics and Communications Conference and the Australian Conference on Optical Fibre Technology, pp 1 – 2, 2008.

[19] Katsutoshi Seki, Tomoyoshi Kobori, James Okello, Masao Ikekawa, "A Cordic-Based Reconfigrable Systolic Array Processor for MIMO-OFDM Wireless Communications," IEEE Workshop on Signal Processing Systems, pp 639 – 644, 2007.

[20] Jun-Feng Tang, Xiao-Jin Li, Gang Zhang, Zong-Sheng Lai, "Design of high-throughput mixed-radix MDF FFT processor for IEEE 802.11.3c," IEEE 11th International Conference on Solid-State and Integrated Circuit Technology, pp 1 – 3, 2012.

[21] Chen Yang, Yi-zhuang Xie, Liang Chen, He Chen, Yi Deng, "Design of a configurable fixed-point FFT processor," IET International Radar Conference, pp 1 – 4, 2015.

[22] Md. Zakir Hussain, Kazi Nikhat Parvin, Zeba Fatima Mir Ilyas Ali, "Performance efficient FFT processor design," International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), pp 286 – 290, 2016.

[23] Bhawna Kalra, J. B. Sharma, "Bit parallel multiplication based 64 Point Pipeline FFT Processor for OFDM application," International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp 1 – 5, 2016.

[24] Johan Löfgren; Peter Nilsson, "On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems," NORCHIP, pp 1 – 4, 2011.

[25] Fahad Qureshi; Muazam Ali; Jarmo Takala, " Multiplierless reconfigurable processing element for mixed radix-2/3/4/5 FFTs, " IEEE International Workshop on Signal Processing Systems (SiPS), pp 1 - 6, 2017.

[26] Xin-Yu Shih; Hong-Ru Chou; Yue-Qu Liu, " Reconfigurable VLSI design of processing kernel for multiple-radix single-path delay feedback FFT systems," IEEE 5th Global Conference on Consumer Electronics , pp 1 - 2, 2016.

[27] Sun-Yuan Kung, "On supercomputing with systolic/wavefront array processors," Proceedings of the IEEE, Vol 72, Issue 7, pp.867 – 884, 1984.

[28] O. Ibarra, M. Palis, "VLSI algorithms for solving recurrence equations and applications," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, Issue 7, pp. 1046-1064, Jul. 1987.

[29] P. Quinton, Y. Robert, "Algorithmes et Architectures Systoliques," Etudes et Recherches en Informatique, Editions Masson, 1989.

[30] Earl E. Jr. Swartzlander, "Systolic FFT Processors: Past, Present and Future, " IEEE 17th International Conference on Application-specific Systems, Architectures and Processors (ASAP'06) , pp.153 – 158, 2006.

[31] Javier Mora, Andrés Otero, Eduardo de la Torre, Teresa Riesgo, "Fast and compact evolvable systolic arrays on dynamically reconfigurable FPGAs," 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), pp.1 - 7, 2015.

[32] H. T. Kung, C. E. Leiserson, "Systolic arrays for (VLSI)," Introduction to VLSI Systems, C. Mead, L. Conway, eds., Addison-Wesley Ltd., Reading, MA, 1980.

[33] D. J. Evans, C. R. Wan, "Massive parallel processing for matrix multiplications: a systolic approach, " Highly Parallel Computations: Algorithms and Applications, M. P. Bekakos, ed., WITpress, Southampton, Boston, pp. 139-173, 2001.

[34] D. Chikouche, R.E. Bekka, A. Khalef, F. Djahli, F. Belilita, "Processeurs parallèles dédiés au traitement du signal en temps réel," Proceedings Conférence DAT'2000, pp. 243-247, Alger, 22-24 Mai 2000.

[35] D. Chikouche, "Contribution à l'implémentation des filtres numériques récursifs unidimensionnels et bidimensionnels sur des réseaux processeurs concurrents : Réalisation de

hauts débits en données par l'usage de la technique CTP," Thèse de Doctorat d'Etat en Electronique, Université de Sétif, Institut d'Electronique, 1999.

[36] D. Chikouche, F. Belilita, N. Amardjia, R.E. Bekka, A. Khellaf, "Structures architecturales parallèles des réseaux processeurs dédiés aux filtrage RII 1D," 3ème Conférence sur le Génie Electrique, CGE 2003, Bordj El Bahri, 15-16 Février 2004.

[37] A. Madanayake, Len T. Bruton, "A real-time systolic array processor implementation of two-dimensional IIR filters for radio-frequency smart antenna applications," IEEE International Symposium on Circuits and Systems, pp. 1252 – 1255, 2008.

[38] Kevin S. H. Ong, Suhaib A. Fahmy, Keck-Voon Ling, "A scalable and compact systolic architecture for linear solvers," 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors, pp. 186 - 187, 2014.

[39] James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, vol.19, pp. 297–301, 1965.

[40] E. Oran Brigham, "The Fast Fourier Transfom and its Applications," Englewood Cliffs, New Jersey, Prentice-Hall, 1988.

[41] L. Auslander, E. Feig, and S. Winograde, "New Algorithms for the Multidimensional Discret Fourier Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 31, Issue 2, pp. 388-403, Apr. 1989.

[42] R. W. Cox and R. Tong, "Two-and three-dimensional image rotation using the FFT," IEEE Transactions on Image Processing, vol. 8, pp. 1297-1299, Sept. 1999.

[43] W. Philips, "On computing the FFT of digital images in quadradtree format," IEEE Transactions on Image Processing, vol. 47, pp. 2059-2060, July 1999.

[44] A. Celantano and V.D. Lecce, "A FFT based technique for image signature generation," Proceedings of SPIE (Society of Photo-Optical Instrumentation Engineers): storage and retrieval for image and video databases, vol. 3022, pp. 457-466, Feb. 1997.

[45] Z. H. Cho et al, "Fourier transform nuclear magnetic resonance tomographic imaging", Proceedings of the IEEE, vol. 70, no. 10, pp 1152-1173, Oct. 1982.

[46] A. C. Kak et M. Slaney, "Principles of Computerized Tomographic Imaging," IEEE Press, New York, NY, USA, 1987.

[47] D. Chikouche, N. Amardjia, R. E. Bekka, "An efficient Radix-Two Algorithm to compute the 2D Fourier Transform," WSEAS Transactions on Signal Processing, vol. 1, Issue 3, pp. 312-315, Dec. 2005.

[48] D. Chikouche, N. Amardjia, N. Khenfer, R. E. Bekka, F. Belilita, "A Fast Algorithm for the Computation of Radix-4 Two-Dimensional Fourier Transform and its Parallel Implementation," Asian Journal of Information Technology, vol. 5, pp. 476-479, May. 2006.

[49] Fairouz Belilita, Nourredine Amardjia, Djamel Chikouche, "An Enhanced Arbitrary Radix FFT Algorithm," International Journal of Applied Engineering Research (IJAER), Vol.12, N° 11, pp. 2692-2701, 2017.

[50] C. S. Burrus and T. W. Parks, "DFT/FFT and Convolution Algorithms," New York, NY, John Wiley, 1985.

[51] H. Guo, G. A. Sitton, and C. S. Burrus, "The Quick Discrete Fourier Transform," Proceedings of IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP) 3, 445–448, 1994.

[52] W. M. Gentleman and G. Sande, "Fast Fourier Transforms - For Fun and Profit," Proceedings of AFIPS (American Federation of Information Processing Societies), vol. 29, pp. 563-578, 1966.

[53] Duhamel, P., and Vetterli, M., "Fast Fourier transforms: A tutorial review and a state of the art," Signal Processing 19(4), pp. 259–299, 1990.

[54] Jiang, L.-X., Liu, C.-Y., and Zhang, P.,"A novel overall in-place in-order prime factor FFT algorithm," 5th International Congress on Image and Signal Processing (CISP), pp. 1500–1503, 2012.

[55] Harris, D.B., and McClellan, J.H., "Vector-radix fast Fourier transform, "Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing, Hartford, CT, pp. 548–551, 1977.

[56] Duhamel, P., "Implementing of split-radix FFT algorithms for complex, real, and real-symmetric data," IEEE Transactions on Acoustics, Speech and Signal Processing 34(2), pp. 285–295, 1986.

[57] Pei, S.C., and Chen, W.Y.,"Split vector-radix-2/8 2-D fast Fourier transform," IEEE Signal Processing Letters 11(5), pp. 459–462, 2004.

[58] Winograd, S., "On computing the discrete Fourier transform", Math. Comp. 32, pp.175–199, 1978.

[59] G. Jayasumana, C. Loeffler, "Searching for the best Cooley-Tukey FFT algorithms," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 12, pp. 2408 - 2411, Apr. 1987.

[60] A. Saidi, "Decimation-in-time-frequency FFT algorithm," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, pp. 453-456, Apr. 1994.

[61] M. Frigo and S. G. Johnson, "The Design and Implementation of FFTW3," Proceedings of the IEEE, vol.93, Issue 2, pp. 216–231, 2005.

[62] S. Bouguezal, "Contribution au développement et à la parallélisation des algorithmes FFT multidimensionnelle," Thèse de Magister, Université de Batna, Institut d'Electronique, 1997.

[63] Larry Carter and Kang Su Gatlin, "Towards an optimal bit-reversal permutation program," Proc. 39th Annual Symposium on Foundation of Computer Science (FOCS), pp. 544–553, 1998.

[64] M. Rubio, P. Gómez, and K. Drouiche, "A new superfast bit reversal algorithm," International Journal of Adaptive Control and Signal Processing, vol. 16, pp. 703-707, 2002.

[65] D. Sundararajan, M. O. Ahmad, M. N. S. Swamy, "A fast FFT bit-reversal algorithm," IEEE Transactions on Circuits and Systems-II, vol. 41, issue 10, pp. 701-703, Oct. 1994.

[66] A. Ramakrishna, N. Balaji, P. Srihari, "An efficient and enhanced memory based FFT processor using radix 16 booth with carry skip adder, " International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 1608 – 1612, 2016.

[67] Md. Zakir Hussain, Kazi Nikhat Parvin, Zeba Fatima Mir Ilyas Ali, "Performance efficient FFT processor design, " International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 286 – 290, 2016.

[68] Bhawna Kalra, J. B. Sharma, "Bit parallel multiplication based 64 Point Pipeline FFT Processor for OFDM application, " International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 1 – 5, 2016.

[69] Rajdeep Kaur, Tarandip Singh, "Design of 32-point mixed radix FFT processor using CSD multiplier, " Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), 538 – 543, 2016.

[70] Chen Yang, Yi-zhuang Xie, Liang Chen, He Chen, Yi Deng, "Design of a configurable fixed-point FFT processor," IET International Radar Conference, 1 – 4, 2015.

[71] B. Gold and T. Bially, "Parallelism in fast Fourier transform hardware," IEEE Transactions on Audio Electro acoustics, vol. 21, pp. 5-16, 1973.

[72] C. D. Thompson, "Fourier transforms in VLSI," IEEE Transactions on Computers, vol. C-32, no. 11, pp. 1047-1057, Nov. 1983.

[73] Pramod Kumar Meher, Basant Kumar Mohanty, Sujit Kumar Patel, Soumya Ganguly, Thambipillai Srikanthan, "Efficient VLSI Architecture for Decimation-in-Time Fast Fourier Transform of Real-Valued Data," IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 62, Issue 12, 2836 – 2845, 2015.

[74] Hao Xiao, An Pan, Yun Chen, Xiaoyang Zeng, "Low-cost reconfigurable VLSI architecture for fast fourier transform," IEEE Transactions on Consumer Electronics, Vol. 54, Issue 4, 1617 – 1622, 2008.

[75] Chao Cheng, Keshab K. Parhi, "Low-Cost Fast VLSI Algorithm for Discrete Fourier Transform," IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 54, Issue 4, 791 – 806, 2007.

[76] C. Fenouillet-Beranger, P. Batude, L. Brunet, V. Mazzocchi, C-M. V. Lu, F. Deprat, J. Micout, M-P. Samson, B. Previtali, P. Besombes, N. Rambal, F. Andrieu, O. Billoint, M. Brocard, S. Thuries, G. Cibrario, M. Vinet, "Recent advances in 3D VLSI integration," International Conference on IC Design and Technology (ICICDT), 1 – 4, 2016.

[77]S. Bouguezel, M. O. Ahmad, M. N. S. Swamy, "An improved radix-16 FFT algorithm," Canadian Conference on Electrical and Computer Engineering 2004, Vol. 2, 1089 – 1092, 2004.

[78] J.-M. Muller ; D. Trystram , "Architectures pour le calcul de la transformée de Fourier discrète,"Traitement du Signal, , Vol. 5, N° 6-NS, pp. 405-420, 1988.

[79] H. Stones, "Parallel processing with the perfect shuffle," IEEE Transactions on Computers, pp.156-161, Feb. 1971.

[80] C. Wu and T. Feng, "Universality of the shuffle-exchange network," IEEE Transactions on Computers, vol. C-30, no. 5, pp. 324-331, May 1981.

[81] S. He and M. Torkelson, "Design and implementation of a 1024 point pipeline FFT processor," IEEE Conference on Custom Integrated Circuits, pp.131-134, May 1998.

[82] Yun-Nan Chang, K.K. Parhi, "An efficient pipelined FFT architecture," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 50, Issue 6, pp. 322-325, June 2003.

[83] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," IEEE Transactions on Computers, vol. C-33, Issue 5, pp. 414-426, May 1984.

[84] Hung-Yu Wang, Jhong-Jhou Wu, Chun-Wei Chiu, Yu-Hsuan Lai, A Modified Pipeline FFT Architecture," International Conference on Electrical and Control Engineering, 4611 – 4614, 2010.

[85] N. Amardjia, D. Chikouche, F. Belilita, S. Bouguezal, "Implémentation de la FFT sur des architectures parallèles VLSI très performantes," 4th International Conference on Computer Integrated Manufacturing CIP'2007.

[86] Shousheng He; M. Torkelson, "A new approach to pipeline FFT processor," Proceedings of International Conference on Parallel Processing, pp. 766 – 770, 1996.

[87] Yun-Nan Chang, K.K. Parhi, "Efficient FFT implementation using digit-serial arithmetic," IEEE Workshop on Signal Processing Systems, pp. 645-653, Oct. 1999.

[88] C. C. W. Hui, T. J. Ding, J. V. McCanny, and R. F. Woods, "A 64 point Fourier transform chip for video motion compensation using phase correlation," IEEE Journal of Solid-state Circuits, vol. 31, pp. 1751-1761, Nov. 1996.

[89] T. Kurtis Johnson, A.R. Hurson, Behrooz Shirazi, "General-Purpose Systolic Arrays, " Journal Computer, Vol. 26 Issue 11, pp. 20-31, Nov. 1993.

[90] M. J. Flynn, "Very high-speed computing systems," Proceedings, IEEE Computer, vol. 54, pp. 1901-1909, 1966.

[91] N. Amardjia, "Développement d'architectures systoliques très performantes pour l'implémentation d'un algorithme récent de calcul rapide de la DFT bidimentionnelle, " Thèse de doctorat d'état, Université Farhet Abbas de Sétif, avril 2007.

[92] R. Hughey, D.P. Lopresti, "Architecture of a programmable systolic array," Proceedings of the International Conference on Systolic Arrays, pp. 41-49, May 1988.

[93] M. Gokhale et al., "Building and using a highly parallel programmable logic array," IEEE Computer, pp. 81-89, Jan. 1991.

[94] G. Causapruno et al., "Reconfigurable Systolic Array: From Architecture to Physical Design for NML," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 11, pp. 3208-3217, Nov. 2016.

[95] M. C. Chen and C. A. Mead, "Concurrent algorithms as space-time recursion equations," VLSI and Modern Signal Processing, S. Y. Kung, H. J. Whitehouse, and T. Kailath, eds., ch. 13, pp. 224-240, Englewood Cliffs, NJ, Prentice-Hall, 1985.

[96] K. M. Chandy and J. Misra, "Systolic algorithms as programs," Distributed Computing, vol. 1, pp. 177-183, 1986.

[97] P. Lee, Z. M. Kedem, "Synthesizing linear array algorithms from nested loop algorithms," IEEE Transactions on Computers, vol. 37, Issue 12, pp. 1578-1598, 1988.

[98] G. Udgirkar and G. Indumathi, "VLSI global routing algorithms: A survey," 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2528 - 2533, 2016.

[99] Doru Florin Chiper, "An efficient algorithm for a memory-based systolic array VLSI implementation of type IV DCT," International Symposium on Signals, Circuits and Systems (ISSCS), pp. 1 – 4, 2015.

[100] M. A. Hasan, A. G. Wassal, "VLSI algorithms, architectures, and implementation of a versatile GF(2m) processor," IEEE Transactions on Computers, vol. 49, no. 10, pp. 1064 – 1073, 2000.

[101] Gang Wu, Chris Chu, "Detailed Placement Algorithm for VLSI Design With Double-Row Height Standard Cells," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 35, Issue 9, pp. 1569 – 1573, 2016.

[102] Rommel M Anacan, Josephine L. Bagay, "Logical Effort Analysis of various VLSI design algorithms," IEEE International Conference on Control System, Computing and Engineering (ICCSCE), pp.19 - 23, 2015.

[103] Dhiraj Sangwan, Seema Verma, Rajesh Kumar, "An Efficient Approach to VLSI Circuit Partitioning Using Evolutionary Algorithms," International Conference on Computational Intelligence and Communication Networks, pp.925 – 929, 2014.

[104] Wei Jin, Chang N. Zhang, Hua Li, "Mapping multiple algorithms into a reconfigurable systolic array," Canadian Conference on Electrical and Computer Engineering, pp. 1187 – 1192, 2008.

[105] H. T. Kung, "Why Systolic Architectures", IEEE Computer, vol. 15, Issue 1, pp. 37-46, Jan. 1982.

[106] H.T. Kung, "Some System and Implementation Issues in Systolic Algorithm Designs," EURASIP International Conference on Digital Signal Processing, Florence, Italy, 1984.

[107] S.Y. Kung, "VLSI Array Processors," Englewood Cliffs, NJ, Prentice Hall, 1988.

[108] J.H. Weston, C.N. Zhang, Li Hua, "Some space considerations of VLSI systolic array mappings," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 48, Issue 4, 419-424, Apr. 2001.

[109] R. Resmi, B. Bala Tripura Sundari, "Allocation of optimal reconfigurable array using graph merging technique," International Conference on Embedded Systems (ICES), pp. 49 – 54, 2014.

[110] C. R. Wan, D. J. Evans, "Nineteen ways of systolic matrix multiplication," Inter. J. Comput. Math., vol. 68, 39-69, 1998.

[111] L. Melkemi, M. Tchuente, "Complexity of matrix product on a class of orthogonally connected systolic arrays," IEEE Transactions on Computers, vol. 36, Issue 5, pp. 615-619, 1987.

[112] Feifei Dong, Sihan Zhang and Cheng Chen, "Improved Design and Analyse of Parallel Matrix Multiplication on Systolic Array Matrix," IEEE International Conference on Computational Intelligence and Software Engineering, pp.1 – 4, 2009.

[113] D. Moldovan, "ADVIS: A Software Package for the Design of Systolic Arrays," IEEE Transactions on Computer-Aided Design, vol. 6, Issue 1, pp. 33-40, Jan. 1987.

[114] G.H. Li, B.W. Wah, "The design of optimal systolic arrays", IEEE Transactions on Computers, vol. 34, Issue 1, pp. 66-77, 1985.

[115] V. V. Dongen, "Quasi-regular arrays: definition and design methodology," Systolic Array Processors, J. McCanny, J. McWhirter, and J. Earl Swartzlander, eds., pp. 126-135, Englewood Cliffs, NJ, Prentice-Hall, 1989.

[116] S.V. Rajopadhye, R. M. Fujimoto, "Automating the Design of Systolic Arrays," The VLSI Journal, vol. 9, pp. 225-242, 1990.

[117] L. Thiele, "Compiler techniques for massive parallel architectures", COMPEURO'90: State of the art in Computer Science, 1990.

[118] F. Raimbault, D. Lavenier. "Relacs for Systolic Programming," IEEE International Conference on Application-Specific Array Processors, pp. 132-135, Venice, Italy, 1993.

[119] F. Raimbault, P. Quinton, D. Lavenier. "Architectures Systoliques et Parallelisme de données; l'environnement de programmation ReLaCS," TSI (Traitement du Signal et des Images), vol. 12, pp. 597-620, 1993.

[120] D. Lavenier, C. Wagner, "Conception d'un réseau systolique à partir de C-stolic. Application à la biologie moléculaire," 4ème Symposium sur les Architectures Nouvelles de Machines, Rennes,1996.

[121]. Sun-Yuan Kung ; Arun ; Gal-Ezer ; Bhaskar Rao, "Wavefront Array Processor: Language, Architecture, and Applications, " IEEE Transactions on Computers, vol. 31 Issue No. 11, pp.1054 – 1066, 1982.

[122] J.H. Kim, S.M. Reddy, "On the Design of Fault-Tolerant Two-Dimensional Systolic Arrays for Yield Enhancement," IEEE Transactions on Computers, vol. 38, Issue 4, pp. 515-525, Apr. 1989.

[123] M.O. Esonu, A.J. Al-Khalili, S. Hariri, "On the design of optimal fault-tolerant systolic array architectures," Proceedings, Fifth International Symposium on Parallel Processing, pp. 352-357, 30 Apr - 2 May 1991.

[124] M.O. Esonu, A.J. Al-Khalili, S. Hariri, D.Al-Khalili, "Fault-tolerant design methodology for systolic array architectures," IEE Proceedings on Computers and Digital Techniques, vol. 141, Issue 1, pp. 17-28, Jan. 1994.

[125] J.H. Kim, "On the design of easily testable and reconfigurable systolic arrays," Proceedings of the International Conference on Systolic Arrays, pp. 505-514, May 1988.

[126] Toshiyuki Ishimura, Akinori Kanasugi, "A Design and Simulation for Dynamically Reconfigurable Systolic Array," Third International Conference on Convergence and Hybrid Information Technology, Vol. 2, pp. 172 - 175, 2008.

[127] Hao Xiao, An Pan, Yun Chen, Xiaoyang Zeng, "Low-cost reconfigurable VLSI architecture for fast fourier transform," IEEE Transactions on Consumer Electronics, Vol. 54, Issue 4, pp. 1617 - 1622, 2008.

[128] D.E. KNUTH, "The art of computer programming", vol. 2, Addison-Wesley Ltd., 1981.

[129] W.P. Burleson, "Polynomial evaluation in VLSI using distributed arithmetic," IEEE Transactions on Circuits and Systems, Vol. 37, Issue: 10, pp. 1299 – 1304, Oct 1990.

[130] Linda Null,Julia Lobur, "Essentials of Computer Organization and Architecture, " Jones & Bartlett Learning, 2006

[131] Gianluca Forte; John M. Espinosa-Duran; Jaime Velasco-Medina, "Systolic architectures to evaluate polynomials of degree n using the Horner's rule," IEEE Fourth Latin American Symposium on Circuits and Systems (LASCAS), 2013.

[132] L.W. Chang and M.Y Chen, "A new systolic array for Discrete Fourier Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, Issue: 10, pp. 1665-1666, Oct. 1988.

[133] J. A. Beraldin, T. Aboulnasr, and W. Steenaart, "Efficient one dimensional systolic array realization of the discrete Fourier transform," IEEE Transactions on Circuits and Systems, vol. 36, pp. 95–100, 1989.

[134] S. Sarkar, A. K.Majumdar, "Fast Fourier transform using linear tagged systolic array," IEEE Conference on Computer and Communication Systems, vol.1, pp. 289-293, Sept. 1990.

[135] Chi-Min Liu and Chein-Wei Jen, "A new systolic array algorithm for discrete Fourier transform," IEEE International Symposium on Circuits and Systems, vol.4, pp. 2212-2215, June 1991.

[136] M. H. Lee, "High speed multidimensional systolic arrays for discrete Fourier transform," IEEE Transactions on Circuits and Systems, vol. 39, Issue: 12, pp. 876 – 879, Dec. 1992.

[137] H. V. Jagadish, T. Kailath, "A family of new efficient arrays for matrix multiplication," IEEE Transactions on Computers, vol. 38, Issue 1, pp. 149-155, 1989.

[138] I. Z. Milentijevic, I. Z. Milovanovic, E. I. Milovanovic, M. K. Stojcev, "The design of optimal planar systolic arrays for matrix multiplication," Comput. Math. Appl., vol. 33, Issue 6, pp. 17-35, 1997.

[139] E. I. Milovanovic, M. B. Tosic, I. Z. Milovanovic, I. Z. Milentijevic, "Designing processor-time optimal systolic arrays for matrix-vector multiplication," J. Electrotechn. Math., vol. 1, pp. 7-19, 1998.

[140] Parastoo Kamranfar, S. Ali Shahabi, Ghazaleh Vazhbakht, Zainalabedin Navabi, "Configurable Systolic Matrix Multiplication," 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems, pp. 336 – 341, 2014.

[141] Ke Chen, Fabrizio Lombardi, Jie Han, "Matrix multiplication by an inexact systolic array," Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures , pp. 151 – 156, 2015.

[142] M. K. Stojčev, E. I. Milovanovič, S. R. Markovič, I. Ž. Milovanovič, "Synthesis of orthogonal systolic arrays for fault-tolerant matrix multiplication, " 27th International Conference on Microelectronics Proceedings, pp. 327 – 334, 2010.

[143] Hyesook Lim ; E.E. Swartzlander, "Multidimensional Systolic Arrays for the Implementation of Discrete Fourier Transforms," IEEE Transactions on Signal Processing, Vol. 47, Issue: 5, May 1999.

[144] Shousheng He; M. Torkelson, "A New Expandable 2D Systolic Array for DFT Computation Based on Symbiosis of 1D Arrays," Proceedings 1st International Conference on Algorithms and Architectures for Parallel Processing, vol.1, pp. 12 – 19, 1995.

[145] Chi-Min Liu ; Chein-Wei Jen, "A new systolic array Algorithm for discret Fourier transform, " IEEE International Sympoisum on Circuits and Systems, 1991.

[146] C. -L. Wang, C. -H. Wei, S. -H. Chen, "Efficient bit-level systolic array implementation of FIR and IIR digital filters," IEEE Journal on Selected Areas in Communications, Vol. 6, Issue 3, pp. 484 – 483, 1988.

[147] M. Lapointe, P. Fortier, H. T. Huynh, "A new faster and simpler systolic structure for IIR filters," IEEE International Symposium on Circuits and Systems, vol. 2, pp. 1227 – 1230, 1990.

[148] S. -M. Lei, K. Yao, "Efficient systolic array implementations of digital filtering, "IEEE International Symposium on Circuits and Systems, 1183 - 1186 vol.2, 1989.

[149] J. Zhang, W. Steenaart, "VLSI realizable high performance structures for real-time state-space filtering," IEEE Transactions on Circuits and Systems, Vol. 36, Issue 4, pp. 631 – 637, 1989.

[150] S. C. Knowles, R. F. Woods, J. G. McWhirter, J. V. McCanny, "A high performance systolic IIR filter architecture," IEE Colloquium on Digital Signal Processing for VLSI, pp. 2/1 - 2/4, 1988.

[151] R. F. Woods, S. C. Knowles, J. V. McCanny, J. G. McWhirter, "Systolic IIR filters with bit level pipelining," ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing, vol.4, pp. 2072 – 2075, 1988.

[152] Shih-Chieh Wen, Chi-Min Liu, Chein-Wei Jen, "The designs of two-level pipelined systolic arrays for recursive digital filters with maximum throughput rate," International Symposium on VLSI Technology, Systems, and Applications - Proceedings of Technical Papers, pp. 317 – 321, 1991.

[153] Rimesh M. Joshi, Arjuna Madanayake, Jithra Adikari, Len T. Bruton, "Synthesis and Array Processor Realization of a 2-D IIR Beam Filter for Wireless Applications, "IEEE

Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, Issue 12, pp. 2241 – 2254, 2012.

[154] H. L. P. Arjuna Madanayake, Len T. Bruton, "A Systolic-Array Architecture for First-Order 3-D IIR Frequency-Planar Filters," IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 55, Issue 6, pp. 1546 – 1559, 2008.

[155] L. A. Sousa, "Bidirectional systolic arrays for digital recursive filters," IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology, vol. 3, pp. 499 - 502, 1998.

[156] M. K. Ibrahim, D. Ait-Boudaoud, "On the implementation of systolic IIR/FIR filters, " IEEE International Sympoisum on Circuits and Systems, vol.1, pp. 492 – 495, 1991.

[157] B. K. Mohanty, P. K. Meher, "High throughput and low-latency implementation of bit-level systolic architecture for 1D and 2D digital filters, " IEE Proceedings - Computers and Digital Techniques , Vol. 146, Issue 2, pp. 91 – 99, 1999.

[158] S. C. Knowles, R. F. Woods, J. G. McWhirter; J. V. McCanny, "Bit-level systolic arrays for IIR filtering," Proceedings. International Conference on Systolic Arrays, pp. 653 - 663, 1988.

[159] Steve Winder, "Analog and Digital Filter Design," second edition, Elsevier, 2002.

[160] Arthur Williams, Fred Taylor, "Electronic Filter Design Handbook," Fourth Edition, McGraw-Hill Education, 2010.

[161] R. B. Urquhart, D. Wood, "Systolic matrix and vector multiplication methods for signal processing Communications, Radar and Signal Processing," IEE Proceedings Vol. 131, Issue 6, pp. 623 – 631, 1984.

[162] S. -M. Lei; K. Yao, "Efficient systolic array implementations of IIR digital filtering," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing Vol. 39, Issue 8, pp. 581 – 584, 1992.

[163] Dr D. Chikouche, Dr R.E. Bekka, "Architectures rapides dynamiquement reconfigurables des fitres numériques récursifs 1D et 2D," Revue traitement du signal, Vol. 16, N°2, 1999, pp. 145-156.

[164] D. Chikouche, R. E. Bekka, "Cylindrical architectures for 1-D recursive digital filters: A state space approach," IEE Proceedings - Computers and Digital Techniques, Vol. 145, Issue 5, 327 – 332, 1998.

[165] W. A. Porter, J. L. Arevena, "Cylindrical arrays for matrix multiplication," Proc.of the 24 allerton conference on communications, control, and computing, Allerton Park, IL., 1986.

[166] J. L. Aravena, "Triple matrix product architectures for fast signal processing," IEEE Transactions on Circuits and Systems, Vol. 35, Issue 1, pp. 119 – 122, 1988.

[167] D. Chikouche, R. E. Bekka, A. Boucenna, "Recursive filters using systolic architectures and switched capacitor techniques," 9th International Conference on Electronics, Circuits and Systems, vol. 2, pp. 595 – 598, 2002.

Résumé : De nombreuses applications du large domaine du traitement numérique du signal (TNS), qui travaillent sur un volume important de données et exécutent un nombre considérable de calculs, dévoilent deux exigences fondamentales : l'exécution des calculs en temps réel et la localisation de l'unité de traitement à leur niveau. Ceci est pratiquement satisfait par l'emploi de processeurs spécialisés et dédiés à ces applications. De plus, l'exigence d'une exécution en temps réel est grandement favorisée par l'utilisation d'architectures parallèles dans la construction de ces processeurs car cela permet d'atteindre des débits en données élevés. Il faut noter que grâce à l'avancée de la technologie microélectronique d'intégration à très grande échelle (VLSI) et à l'utilisation des techniques de conception assistée par ordinateur (CAO), la construction de tels processeurs s'est grandement simplifiée. Dans ce contexte, l'objectif de ce travail est de proposer des processeurs dédiés à base d'architectures parallèles concurrentes et constituées de réseaux ordonnés de cellules ou processeurs élémentaires (PEs). Plusieurs architectures parallèles seront proposées et une étude comparative entre celles-ci sera faite. La comparaison portera sur plusieurs facteurs dont la régularité et la modularité des réseaux de PEs, l'homogénéité et la localité des réseaux d'interconnexion, le débit en données et la surface silicium à implanter. Comme exemples d'application, ces réseaux de processeurs seront adaptés au calcul de la transformée de Fourier et aux filtres numériques récursifs qui sont deux outils de calcul incontournables dans le domaine du TNS.

Mots clés : Architectures parallèles; Filtres numériques récursifs; Réseaux de processeurs élémentaires; Transformée de Fourier Discrète; Transformée de Fourier Rapide.

Abstract: Many applications of the wide domain of digital signal processing (DSP), which work on a large volume of data and perform a considerable number of calculations, reveal two essential requirements: the execution of calculations in real time and the localization of the processing unit at their level. This is practically satisfied by the use of specialized processors dedicated to these applications. In addition, the necessity of a real-time execution is greatly supported by the use of parallel architectures in the construction of these processors because it allows achieving high data rates. It should be noted that thanks to the advancement of microelectronic integration technology on a very large scale (VLSI) and the use of computer-aided design (CAD) techniques, the construction of such processors has been greatly simplified. In this context, the objective of this work is to propose dedicated processors based on concurrent parallel architectures and constituted by ordered networks of elementary cells or processors (EPs). Several parallel architectures will be proposed and a comparative study between them will be made. The comparison will cover several factors including the regularity and modularity of the EPs networks, the homogeneity and locality of the interconnection networks, the data rate and the silicon surface to be implanted. As application examples, these processor networks will be adapted to the calculation of Fourier transform and recursive digital filters, which are two compelling tools in the field of DSP.

Key words : Parallel architectures; Recursive digital filters; Elementary processor arrays; Discrete Fourier transform; Fast Fourier transform.

ملخص: عدة تطبيقات في المجال الواسع لمعالجة الإشارات الرقمية، التي تعمل على كمية كبيرة من البيانات وتؤدي عددًا كبيرًا من الحسابات ،تكشف عن اثنين من المتطلبات الأساسية: تنفيذ الحسابات في الوقت الفعلي وتوطين وحدة المعالجة على مستواها. هذا ما يتحقق عمليا عن طريق استخدام المعالجات المتخصصة المكرسة لهذه التطبيقات. بالإضافة إلى ذلك، يتم تحقيق متطلبات التنفيذ في الوقت الحقيقي إلى حد كبير باستخدام معماريات موازية في بناء هذه المعالجات لأنها تسمح بتحقيق تدفقا عاليا للبيانات. وتجدر الإشارة إلى أنه بفضل تطوير تكنولوجيا التكامل الإلكتروني الدقيقة على نطاق واسع جداً (VLSI) واستخدام تقنيات التصميم بمساعدة الكمبيوتر(CAD)، فإن بناء هذه المعالجات تبسط كثيرا. في هذا السياق، الهدف من هذا العمل هو اقتراح معالجات مخصصة تقوم على معماريات متوازية و متنافسة تشكلها شبكات مرتبة من خلايا أو معالجات أولية(EPs). سيتم اقتراح عدة معماريات موازية وسيتم إجراء دراسة مقارنة بينهما. ستشمل المقارنة العديد من العوامل بما في ذلك انتظامية وشكلية شبكات المعالجات الأولية ، تجانس ومحلية شبكات التوصيل، تدفق البيانات ،وسطح السيلكون المراد غرسه. كأمثلة للتطبيق، سيتم تكييف صفائف المعالج هذه إلى حساب تحويل فوريه والمرشحات الرقمية المتكررة التي تعتبر نوعان من الأدوات الحسابية المهمة في مجال معالجة الإشارات الرقمية.

الكلمات الرئيسية: معماريات موازية، مرشحات رقمية متكررة، شبكات المعالجات الأولية، تحويل فوريه المتقطع، تحويل فوريه السريع.