Implémentation de la FFT sur des architectures parallèles VLSI très performantes

N. Amardjia*, D. Chikouche**, F. Belilita***, S. Bouguezal*
Laboratoire LIS, Département d'Electronique, Faculté des Sciences de l'Ingénieur, Université Ferhat Abbas de Sétif.
*Département d'Electronique, Faculté des Sciences de l'Ingénieur, Université Ferhat Abbas de Sétif.
**Département d'Electronique, Faculté des Sciences et Sciences de l'Ingénieur, Université Mohamed Boudiaf de M'sila.
***Département de Physique, Faculté des Sciences, Université Ferhat Abbas de Sétif.
E-mail : amardjia_nour@yahoo.fr dj_chikou@yahoo.fr

***Résumé* - Cet article présente trois types d'architectures VLSI très performantes de processeurs dédiés au calcul de la transformée de Fourier rapide. Elles s'appuient dans leur développement sur la forme récursive de l'algorithme par décimation temporelle en base 2, reformulé de façon à permettre une implémentation matérielle plus aisée. Ces architectures à structures parallèles sont formées d'un ensemble ordonné de processeurs élémentaires identiques sur lesquels les calculs sont distribués d'une manière efficace et appropriée. Une étude comparative de performance, en termes de débit en données et de nombre de processeurs élémentaires, donc en surface Silicium implantable, sera faite entre ces trois types d'architectures dénotées suivant l'ordonnancement de leurs processeurs par étendues, rebouclées et linéaires.**

I. INTRODUCTION

Le calcul de la transformée de Fourier rapide (FFT) est d'une grande importance dans les applications du traitement du signal telles que le filtrage numérique, l'acoustique, la sismographie, le radar, le sonar, les systèmes numériques de communications, etc. [1-4]. Dans les applications où une exécution en temps réel est nécessaire, l'utilisation de processeurs dédiés à ce calcul serait la mieux adaptée [5, 6]. Grâce à l'avancée de la technologie microélectronique d'intégration à très grande échelle (VLSI) et à l'apparition des techniques de conception par ordinateur (CAO), la construction de tels processeurs est devenue possible. L'utilisation appropriée d'un réseau ordonné de processeurs élémentaires (PE) dans la construction de ces processeurs dédiés permet d'aboutir à des architectures parallèles hautement concurrentes et admettant des débits en données élevés. Le développement des architectures proposées dans cet article se base sur l'algorithme FFT de Cooley/Tukey [7], reformulé pour permettre plus facilement un tel développement. En effet l'algorithme de Cooley/Tukey permet de décomposer une transformée de Fourier discrète (TFD) à N échantillons en (N/2) transformées minimales, à 2 échantillons chacune, répétées sur $log_2(N)$ étapes. Il faut noter que chaque calcul élémentaire d'une étape de la FFT dépend des données fournies par l'étape précédente et seulement de celles-là. Ce qui entraîne qu'une transformée minimale est indépendante de ses semblables de même rang, et qu'une FFT peut donc être parallélisée et calculée sur place lors de son implémentation. Différentes architectures parallèles seront développées, dans cet article, pour l'implémentation de la FFT. Les performances de ces structures seront analysées en termes de débit en données et de nombre de PEs ou surface Silicium implantable.

II. REFORMULATION DE LA FFT PAR DECIMATION TEMPORELLE EN BASE 2.

La transformée de Fourier discrète $X(k)$ d'une fonction $x(n)$ est donnée par la relation (1) :

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

- n et k représentent respectivement les variables discrètes et normalisées de l'espace original et de l'espace transformé.
- N représente le nombre des valeurs discrètes et successives des variables n et k
- $W_N^{kn} = e^{-2j\pi kn/N}$ sont les coefficients de transformation.

Si N est élevé, son calcul requiert un très grand nombre d'opérations complexes : N^2 multiplications et $N(N-1)$ additions. Ce nombre d'opérations est drastiquement réduit par l'utilisation d'algorithmes FFT. L'algorithme FFT par décimation temporelle en base 2, utilisant les séries arithmétiques dans son développement, permet une approche matérielle plus aisée lors de son implémentation sur des architectures parallèles. L'idée de départ consiste à considérer N une puissance de 2 ($N=2^p$, donc $p = log_2(N)$) et de réécrire différemment les indices des composantes $x(n)$ et $X(k)$ dans l'équation de la TFD donnée par (1) :

$$\begin{aligned} k &= 2^{p-1}.k_{p-1} + 2^{p-2}.k_{p-2} + \cdots + 2.k_1 + k_0 \\ n &= 2^{p-1}.n_{p-1} + 2^{p-2}.n_{p-2} + \cdots + 2.n_1 + n_0 \end{aligned} \qquad (2)$$

La factorisation successive des termes suivant les n_i en commençant par le sous indice n_{p-1} permettra, après des changements de variables adéquats et quelques artifices de calcul [8, 9], de décomposer récursivement, en p étapes (variation de i), la transformée globale en $(N/2)$ transformées minimales par étape (variation de r), chacune sur 2 valeurs. Ces transformées minimales sont données par la relation (3) :

$$\begin{pmatrix} X_i(r) \\ X_i(r+2^{p-i}) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & W_N^{C_i(r)} \end{pmatrix} \times \begin{pmatrix} X_{i-1}(r) \\ X_{i-1}(r+2^{p-i}) \end{pmatrix} \qquad (3)$$

avec

$$\boldsymbol{C_i(r)} = \left[\sum_{q=0}^{i-2} 2^q . \boldsymbol{r}_{p-1-q} \right] . 2^{p-i} \qquad (4)$$

L'ensemble des transformées minimales est déterminé en appliquant l'algorithme suivant à la relation (3):

Algorithme :

Pour i : 1 … p

Pour r : 0 … N-1 (sauf les valeurs où $r_{p-i} \neq 0$)

Calcul de (3)

Chaque transformée minimale requiert une multiplication et deux additions complexes (sauf à l'étape 1 où seulement deux additions complexes sont nécessaires). Le nombre global des transformées minimales étant de *(N/2).log₂(N)*, le nombre total d'opérations est donc de *(N/2).log₂(N/2)* multiplications complexes, et de *N.log₂(N)* additions complexes. La FFT en base 2 a donc un ordre de complexité O(*N.log₂(N)*). En comparant cette complexité avec celle du calcul direct de la TFD qui est de O(N^2) le rapport est de $N/log_2(N)$. Pour une TFD de N =1024 points par exemple, et en raisonnant en termes de nombre d'opérations seulement, si le calcul se fait par la FFT, le gain en temps d'exécution est supérieur à 100 par rapport au calcul direct.

III. SCHEMATISATION DES CALCULS

Chaque cellule de base représentée par *(3)* met en œuvre les mêmes calculs pour $X_i(r)$ et $X_i(r+2^{p-i})$ avec seulement un changement de signe dans la somme. Le diagramme de la figure 1, baptisé papillon, schématise les calculs mis en œuvre. Le noeud plus (+) représente une addition entre l'entrée horizontale et l'entrée transversale, et le noeud moins (-) une soustraction, avec l'entrée horizontale à soustraire de l'entrée transversale.

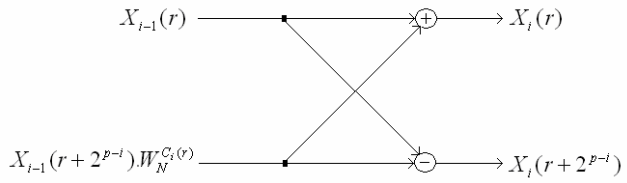

Figure 1. Papillon de la FTT en base 2.

L'évolution de l'ensemble des calculs, incluant les différentes étapes, peut donc être mieux visualisée par l'utilisation des graphes de fluence qui font apparaître tous les papillons intervenant dans une FFT. Les graphes de fluence seront d'ailleurs notre point de départ pour l'implémentation de la FFT sur des architectures parallèles. La figure 2 donne à titre d'exemple le diagramme de fluence de la FFT pour N=8. Ce diagramme de fluence peut être généralisé pour tout N puissance de 2.

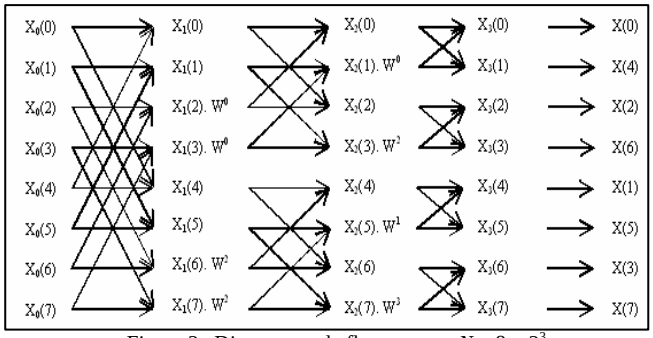

Figure 2. Diagramme de fluence pour $N = 8 = 2^3$.

IV. IMPLEMENTATION DE LA FFT SUR DES ARCHITECTURES PARALLELES PERFORMANTES

A. *Le processeur élémentaire*

Chaque cellule de base ou PE, réalisant les calculs de la relation (3), effectue 3 opérations complexes : une multiplication et deux additions. La structure interne d'un tel processeur est donnée sur la figure 3. Cette dernière présente un multiplieur et deux additionneurs complexes. Le papillon, présent sur les graphes de fluence de la FFT, est représenté par le bloc de droite englobant les deux additionneurs complexes.

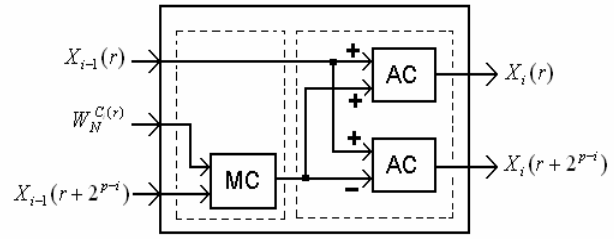

Figure 3. Structure interne du PE à base d'opérateurs complexes.

Les opérations complexes sont en fait réalisées à partir d'opérateurs réels. La structure interne du PE en fonction de ces derniers permettra alors d'avoir une vue plus concrète de sa complexité de mise en œuvre. Cette structure est représentée sur la figure 4. Elle présente quatre multiplieurs et six additionneurs réels

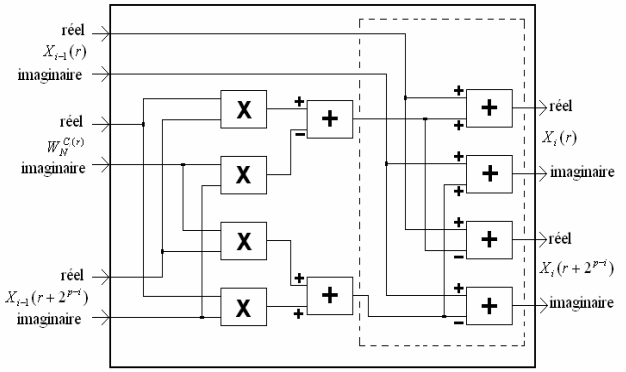

Figure 4. Structure interne du PE à base d'opérateurs réels.

Un facteur important à mettre en évidence est le temps de réponse du processeur. En posant t_m et t_a les temps de réponse respectifs du multiplieur et de l'additionneur réels, et en ne prenant en compte que ces temps pour simplifier le raisonnement, le temps de réponse du PE est de :

$$TR\_PE = t_m + 2.t_a \quad (5)$$

B. *Implémentation de la FFT sur des architectures parallèles étendues*

La première architecture de la FFT proposée présente une structure en pipeline. Elle est directement déduite du graphe de fluence de celle-ci, avec une réorganisation dans les réseaux d'interconnexion pour faire ressortir le facteur modularité au niveau des différents étages. En prenant l'exemple de $N = 8$ échantillons temporels, la FFT pourrait être implémentée selon la structure donnée sur la figure 5. Cette structure est étendue sur trois rangées, chacune ayant quatre PEs. Pour le cas général, elle aura (N/2) PEs par rangée et sera étendue sur $log_2(N)$ rangées. Les échantillons arrivent en parallèle et en même temps sur les entrées des PEs de la rangée limitrophe de gauche et les composantes spectrales sont récupérées simultanément aux sorties des PEs de la rangée frontalière de droite.

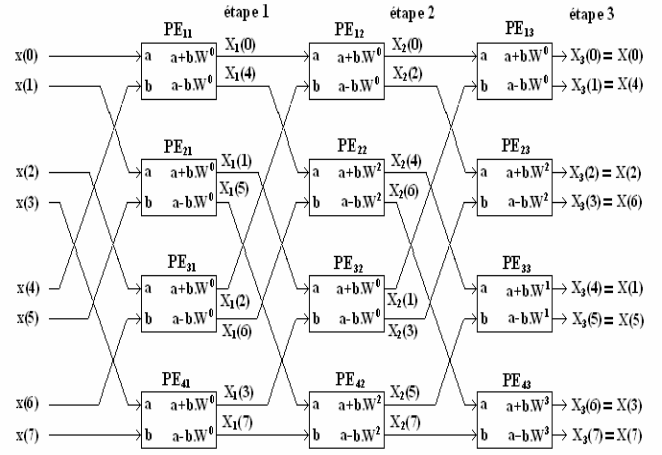

Figure 5. Architecture parallèle étendue de la FFT (N = 8).

Pour un nombre d'échantillons N, le nombre total d'opérateurs réels est de $(2.N).log_2(N)$ multiplieurs et $(3.N).log_2(N)$ additionneurs. Pour un débit de données continu, c'est-à-dire si plusieurs FFT sont à calculer sur des trames de données adjacentes, chaque trame ayant $N=2^p$ données, le premier spectre relatif à la première trame est obtenu en $log_2(N)$ étapes. Il sera suivi d'un spectre par étape pour chaque nouvelle trame de données.

C. *Implémentation de la FFT sur des architectures parallèles rebouclées*

Quand le nombre d'échantillons N devient très important, le nombre de PEs devient excessif pour une implémentation étendue. Ceci pourrait poser problème lors de l'implantation sur la même plaque de silicium. Une alternative permettant de remédier à cet inconvénient serait de n'utiliser qu'une ou quelques rangées de PEs et de reboucler la structure sur elle même, et ceci afin de réutiliser les résultats intermédiaires présents sur ses sorties. Un système de contrôle est alors ajouté pour pourvoir 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 rediriger les sorties du réseau (états intermédiaires) vers ses entrées. Des bascules doivent être prévues 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 6 fournit un schéma bloc explicatif d'une telle structure.

La figure 7 donne la structure d'une architecture parallèle rebouclée pour le calcul de la FFT pour 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, avaient pour mission de rediriger les résultats intermédiaires vers l'entrée du réseau de processeurs.

Les coefficients $W_8^{C_i(r)}$, variables d'une étape à l'autre, ont été notés d'une manière générale W, sans exposant.

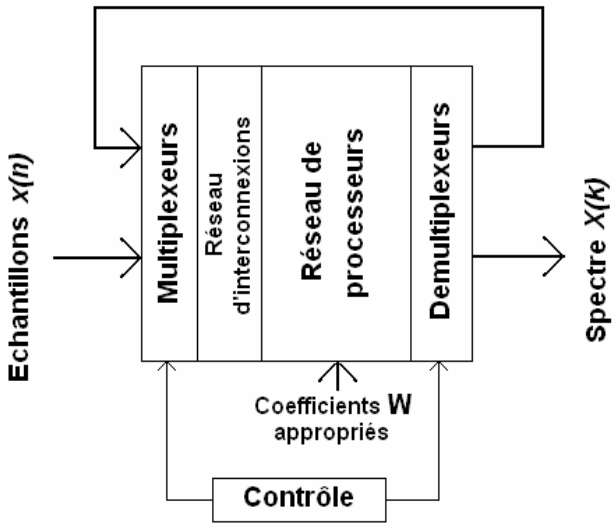

Figure 6. Structure générale d'une architecture parallèle rebouclée.

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. Pour $N = 1024 = 2^{10}$, 5120 PEs sont nécessaires pour une architecture étendue, alors que seulement 512 PEs le sont pour une architecture rebouclée. Cela fait un gain de 90%, et sur le nombre de PEs et sur le nombre de réseaux d'interconnexion, et donc sur la surface silicium totale à utiliser.

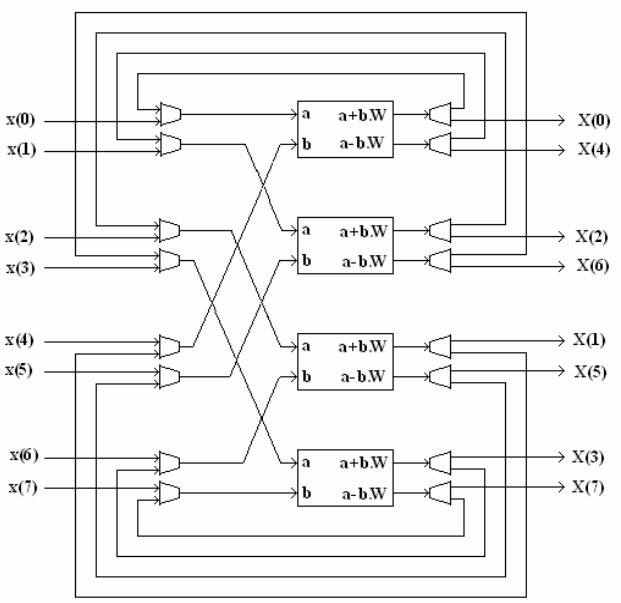

Figure 7. Architecture parallèle rebouclée de la FFT (N = 8).

D. *Implantation de la FFT sur des architectures linéaires*

L'architecture de la FFT rebouclée permet de réduire considérablement le nombre de PEs (d'un rapport de $log_2(N)$). Les architectures linéaires permettent de réduire encore plus ce nombre, et c'est pour cela qu'ils ont été le sujet de plusieurs travaux de recherche [10-13]. L'architecture linéaire proposée ici, ne permet de garder qu'une ligne de PEs, composée de $log_2(N)$ processeurs. La figure 8 fournit un schéma bloc explicatif d'une telle structure. Le nombre total de PEs utilisés dans une architecture étendue a donc été divisé cette fois ci par (N/B). Ce qui représente un gain en nombre de PEs encore beaucoup plus intéressant que celui de l'architecture rebouclée. Si on reprend l'exemple de $N = 1024 = 2^{10}$, 10 PEs seulement seront nécessaires pour une architecture FFT linéaire. Cela fait un gain en surface silicium de 99.8% par rapport à une architecture étendue, et de 98% par rapport à une architecture rebouclée. Ce gain ne prend en compte que les PEs, et est donc un petit peu exagéré, car des éléments supplémentaires, en l'occurrence des éléments mémoire et des commutateurs sont additionnés à la structure pour veiller à son bon fonctionnement.

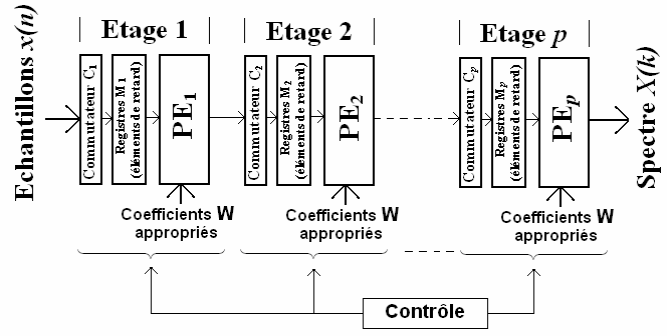

Figure 8. Structure générale d'une architecture parallèle linéaire.

Afin d'avoir une idée claire sur le fonctionnement d'une telle structure, un exemple d'architecture linéaire de la FFT, avec N = 8, est exposé sur la figure 9. La figure 10 représente les modes de fonctionnement des commutateurs C_1, C_2 et C_3. L'évolution des calculs est donnée sur le tableau 1.

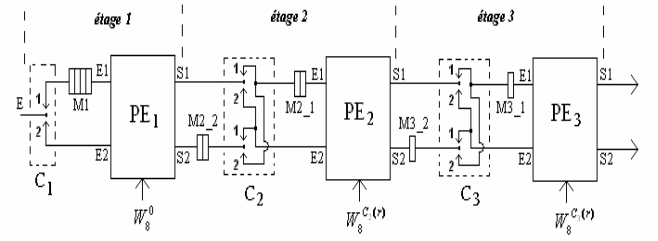

Figure 9. Architecture parallèle linéaire de la FFT (N = 8).

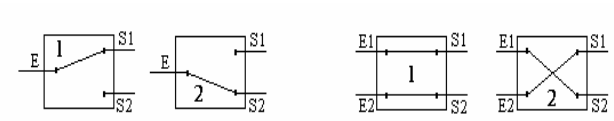

Figure 10. Etats possibles des commutateurs C_1, C_2 et C_3.

Tableau 1.
Sorties des PEs et contenus des différents registres M*i* pendant les différentes étapes de calcul pour l'architecture linéaire de la FFT (N = 8).

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}
											$X_3(6)$
											$X_3(7)$

Une remarque importante est à soulever : si plusieurs FFT sont à calculer, et que le réseau doit donc travailler sur des trames de données adjacentes, on pourra voir que les PEs ne sont actifs que pendant la moitié du temps. Ceci implique qu'entre 2 calculs consécutifs de FFT, rien n'est récupéré à la sortie du réseau pendant (N/2) étapes. Si maintenant on considère qu'un débit en données plus important est présent à

l'entrée du réseau et qu'un système de mémoires peut être prévu pour retenir les données et les fournir à ce dernier, à la cadence adéquate et dans l'ordre donné sur la figure 11 (C_1 et M1 ont été éliminés), les PEs seront alors tout le temps actifs, et deux composantes spectrales sont obtenues à la sortie du réseau à chaque étape. Dans le cas général, où la structure est composée de $log_2(N)$ étages, le spectre de la première trame est obtenu à l'étape $(N-1)$. Un spectre correspondant à une nouvelle trame d'échantillons est obtenu toutes les (N/2) étapes suivantes.

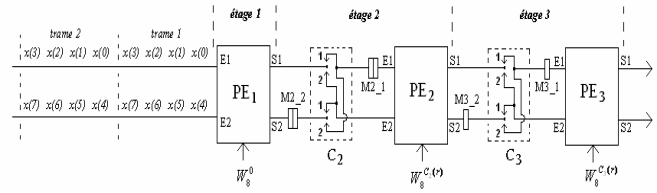

Figure 11. Architecture linéaire de la FFT à débit amélioré (N = 8)

V. COMPARAISON ENTRE LES DIFFERENTES ARCHITECTURES FFT PRESENTEES

Les tableaux 2 et 3 récapitulent les caractéristiques importantes des différentes architectures proposées, respectivement en nombre de processeurs et opérateurs réels et en nombre d'étapes de calcul.

Tableau 2.
Caractéristiques en nombre de processeurs et opérateurs réels des différentes architectures FFT proposées.

	Nombre de processeurs	Nombre de multiplieurs réels	Nombre d'additionneurs réels
FFT étendue	$(N/2)log_2(N)$	$(2N).log_2(N)$	$(3N).log_2(N)$
FFT rebouclée	$N/2$	$2N$	$3N$
FFT linéaire (sans C_1 ni M1)	$log_2(N)$	$4.log_2(N)$	$6.log_2(N)$

Tableau 3.
Caractéristiques en nombre d'étapes de calcul des différentes architectures FFT proposées.

	Nombre d'étapes pour le calcul du 1er spectre	Nombre d'étapes pour la sortie du spectre suivant
FFT étendue	$log_2(N)$	1
FFT rebouclée	$log_2(N)$	$log_2(N)$
FFT linéaire (sans C_1 ni M1)	N-1	$N/2$

VI. CONCLUSION

Trois différentes architectures permettant l'implémentation de la FFT ont été présentées. Si un choix devait se faire entre ces différentes architectures, plusieurs considérations devraient être prises en compte : la faisabilité, le coût, les contraintes auxquelles est soumise l'application à laquelle est destinée l'implantation, etc. En s'aidant des tableaux 2 et 3, on peut conclure que l'architecture étendue 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 mais son temps de réponse est allongé 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 architectures précitées. Cela est bien sûr fait au détriment du temps de réponse qui est encore rallongé par rapport à ceux des deux premières architectures citées.

REFERENCES

[1] A.E. Cetin, O.N. Gerek, and Y.Yardimci, "Equiripple FIR filter design by the FFT algorithm," *IEEE Signal Processing Magazine*, vol.14, pp.60-64, March 1997.

[2] Catherine CATZ, "Analyseurs de Fourier, Traité Mesures et Contrôle," *Techniques de l'Ingénieur*, 2003.

[3] J. G. Proakis, "Digital Communications," Mc Graw Hill, 4th edition, 2000.

[4] F. Le Chevalier, "Principes de Traitement des Signaux Radar et Sonar," Masson, 1989.

[5] 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.

[6] 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.

[7] 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.

[8] S. Bouguezal, D. Chikouche and A. Khellaf, "An efficient algorithm for the computation of the multidimensional discrete Fourier transform," *Multidimensional Systems and Signal processing*, vol. 10, pp. 275-304, July 1999.

[9] 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.

[10] 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.

[11] A.M. El-Khashab, E.E. Swartzlander, "A modular pipelined implementation of large fast Fourier transforms," *IEEE Transactions on Signals, Systems and Computers*, vol. 2, pp. 995 – 999, Nov. 2002.

[12] T. Lenart and V. Owall, "A 2048 complex point FFT processor using a novel data scaling approach," *IEEE Proceedings of International Symposium on Circuits and Systems*, vol. 4, pp. 45 – 48, May 2003.

[13] 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.