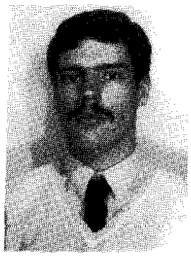
Une architecture optimisée de processeur de filtrage numérique intégré; évaluation graphique de son rendement

An optimized architecture of a digital filtering integrated procesor and efficiency graphical valuation



Eric MARTIN

Institut d'Électronique Fondamentale, Faculté d'Orsay, Paris-XI, 91405 ORSAY CEDEX

Agrégation de génie électrique, ENSET, Cachan. Assistant normalien à l'Université de Paris-XI depuis octobre 1985, il met actuellement la dernière main à sa thèse nouvelle formule sur l'optimisation des architectures dédiées au traitement du signal.

Roger REYNAUD

Institut d'Électronique Fondamentale, Faculté d'Orsay, Paris-XI, 91405 ORSAY CEDEX.

Agrégation de physique, ENSET, Cachan. Thèse de troisième cycle au laboratoire des signaux et systèmes sur le sujet « Filtrage de Kalman appliqué à la déconvolution de signaux monodimensionels : (1) Étude d'un nouvel algorithme de type Kalman rapide. (2) Implantation dans un processeur spécialisé opérant en arthmétique entière ». A rejoint l'Institut d'Électronique Fondamentale dans le groupe micro-électronique en tant que maître assistant en janvier 1985. Dans cette équipe, il anime l'action « Achitectures dédiées au traitement du signal ».

Plilippe ELLEAUME

Thomson-CSF, Division Système Défense et Contrôle, Département DTP/TES, 1, rue des Mathurins, 92222 BAGNEUX CEDEX

Ingénieur ISEN, ENSEAE. A travaillé plusieurs années dans le domaine des études théoriques sur le traitement du signal numérique et la réalisation d'opérateurs à forte puissance de calcul. Il dirige maintenant un laboratoire s'occupant de la conception et la réalisation de processeurs de traitement du signal.

Claude-Alain WAMBERGUE

Thomson-CSF, Division Système Défense et Contrôle, Département DTP/TES, 1, rue des Mathurins, 92222 BAGNEUX CEDEX

Ingénieur ISEN, M. Sc, Univ. of Colorado. A travaillé plusieurs années dans le domaine des études théoriques sur le traitement du signal. Depuis un an, a rejoint un laboratoire de développement d'applications de traitement du signal radar, où il étudie plus particulièrement l'architecture de systèmes et réalise des processeurs spécialisés.

Francis DEVOS

Institut d'Électronique Fondamentale, Faculté d'Orsay, Paris-XI, 91405 ORSAY CEDEX

Professeur d'Université à l'Institut d'Électronique Fondamentale. Depuis 1980, il anime une équipe d'architecture et de conception de machines dédiées. Ses sujets d'intérêt sont centrés sur les architectures dédiées à parallélisme massif.

RÉSUMÉ

Cette étude s'inscrit dans le cadre de la conception d'un processeur dédié au filtrage numérique et présente l'optimisation de son architecture. Nous définissons la notion de rendement temporel d'utilisation de la surface de silicium et le mesurons par une méthode graphique. A partir du choix d'une jauge d'algorithme, le papillon de la Transformée de Fourier Rapide, nous évaluons l'influence de tel choix d'architecture sur les performances et le rendement. Nous pouvons alors mesurer de façon objective l'apport des diverses modifications. Le plan suit la conception descendante proposée par Bowen et Brown [5], choix et analyse d'un algorithme (*cf.* 2), analyse de la structure à implanter (*cf.* 3 et 4). A la différence de [5], nous essayons de valider cette structure indépendamment de la technologie (*cf.* 5).

MOTS CLÉS

Traitement du signal, architecture de systèmes, optimisation graphique.

SUMMARY

This paper deals with the optimization of a digital signal processor's architecture. We define the notion of silicon area temporal utilization efficiency and calculate it with a graphic method. Using benchmark algorithm (Fast Fourier Transform butterfly) we estimate the influence of an architecture choice on its efficiency. In this manner we can objectively measure the contribution of the multifarious modifications.

KEY WORDS

Signal processing, computer architecture, graphic optimization.

TABLE DES MATIÈRES

1. Introduction

La transformée de Fourier est un algorithme très utilisé en traitement du signal [1, 2, 3] qui nécessite une quantité importante de calculs. S'il est possible de programmer cet algorithme sur les nombreux processeurs de traitement du signal [4, 7, 8, 11-13, 15-18], ceux-ci sont relativement lents pour cette application.

Nous proposons d'étudier ici l'architecture d'une unité arithmétique dans une optique processeur-en-tranche, ou intégrée. Cette unité arithmétique devra être optimisée suivant les deux critères contradictoires suivants :

– réduction du temps de calcul de la transformée de Fourier. Il faut pour celà réduire non seulement le temps de cycle (par « pipelinage »; *fig.* 4, ou mise en parallèle, *fig.* 3, des opérateurs arithmétiques), mais aussi minimiser le nombre global de cycles nécessaires au calcul;

– minimisation de la surface de silicium de l'unité de calcul.

L'évaluation de la surface de silicium dépendant aussi bien du nombre de portes de l'architecture que des liaisons internes, nous remplacerons cette donnée par la complexité en nombre de portes de l'architecture.

Pour pouvoir évaluer le compromis entre les deux critères précédents, nous introduisons la définition d'un rendement. Le rendement UTS (Utilisation

Temporelle de la Surface de silicium) est le rapport de deux termes : le produit de la surface de silicium utilisée à chaque instant par son temps partiel d'utilisation sur le produit de la surface totale par le temps global du calcul.

La conception de l'unité arithmétique se fera en cherchant à obtenir à chaque étape de l'étude le meilleur rendement UTS . Les diverses solutions envisagées seront comparées à l'aide d'un graphique permettant de calculer le rendement UTS sur l'ensemble de l'architecture. Il va sans dire que le rendement est fonction de l'algorithme que l'on cherche à implanter, et qu'une structure fort efficace pour certains algorithmes peut se retrouver moins adaptée pour d'autres [14].

Une analyse préliminaire de la TFR va permettre de définir les opérateurs qui doivent figurer dans l'unité arithmétique.

2. Analyse des calculs de l'algorithme

Nous avons retenu l'algorithme de transformation de Fourier à base 2 à entrelacement temporel. Cet algorithme, représenté schématiquement sur la figure 1, se compose d'étapes successives comprenant

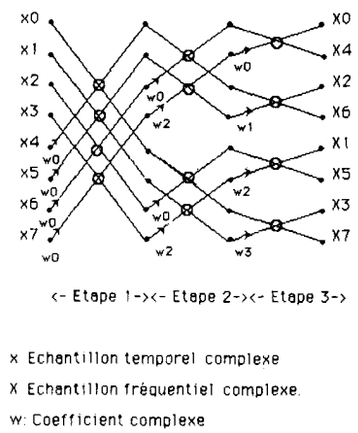


Fig. 1. — Représentation d'une TFR sur huit points.

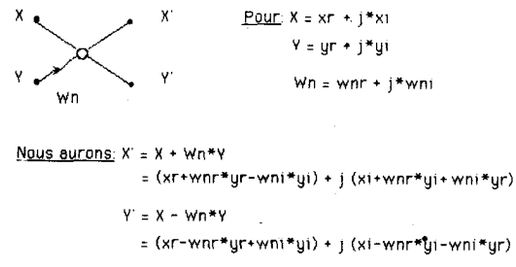


Fig. 2. — Calculs d'un papillon TFR.

chacune une suite de papillons à calculer. Le papillon TFR (*fig.* 2) est donc le motif de base qu'il faut optimiser.

Nous réalisons les calculs du papillon en trois séquences :

— multiplication des parties réelles et imaginaires de Y et W;

— addition et soustraction des produits (yr*wr, yi*wi...);

— addition de nombres complexes (X +/− (W*Y)).

Il nous faut définir plusieurs unités de temps qui interviennent ulterieurement : le temps de calcul propre par opérateur, le temps de cycle (synchronisation des opérateurs), le temps de calcul d'un papillon représentant plusieurs temps de cycle.

L'analyse du mécanisme de réalisation du papillon montre la nécessité de deux opérateurs matériels : un multiplieur et un additionneur-soustracteur.

Nous allons maintenant voir comment disposer ces opérateurs dans l'unité arithmétique.

3. Structure de l'unité arithmétique

3.1. Choix des opérateurs

Le souci de minimiser la surface de silicium de l'unité arithmétique nous entraîne à rentabiliser le fonctionnement des opérateurs. Pour cela, nous devons prendre en compte deux contraintes matérielles qui leurs sont liées :

— le temps de calcul par opérateur;

— la surface de silicium (ou nombre de portes) par opérateur.

Prenons l'exemple d'une technologie bipolaire rapide.

Un multiplieur 16 par 16 vers 32 bits nécessite approximativement une surface Smul de 2 500 portes, pour un temps de calcul Tmul de 70 ns (TRW MPY 016 K) [11]. Un additionneur-soustracteur sur 32 bits nécessite approximativement une surface Sadd de 800 portes pour un temps de calcul Tadd de 50 ns (74F181). Ces chiffres montrent le rôle critique du multiplieur sur le rendement UTS de l'unité arithmétique.

Il faut compter quatre multiplications et huit additions ou soustractions par papillon (*fig.* 2). En tenant compte du rôle prépondérant du multiplieur, il faut réaliser les calculs d'un papillon en quatre cycles (une multiplication par cycle), et imbriquer les calculs du papillon suivant sans attente. Dans cette optique, devons-nous choisir d'associer un ou deux additionneurs à notre multiplieur?

— Un seul additionneur devra effectuer deux opérations en un temps de cycle Tcyc1, imposant une durée de celui-ci de 100 ns. Cette configuration architecturale conduit à un rendement UTS de 77 %.

$$(Smul*Tmul + Sadd*2*Tadd)/(Smul*Tcyc1 + Sadd*Tcyc1) = 77\ \%$$

– L'utilisation de deux additionneurs travaillant en même temps que le multiplieur, imposera un temps de cycle Tcyc2 de 70 ns. Le rendement UTS des opérateurs sera alors de 89 %.

$$(Smul*Tmul+2*Sadd*Tadd)/(Smul*Tcyc2+2*Sadd*Tcyc2)=89\ \%$$

Nous optons pour la structure à deux additionneurs. Ce choix découle de la comparaison des rendements UTS obtenus. Remarquons qu'il existe encore d'autres solutions pouvant améliorer le rendement UTS, comme par exemple l'utilisation d'un multiplieur « pipeliné ».

3.2. Infrastructure de l'unité arithmétique

Après avoir défini les opérateurs en type et nombre, il faut étudier les interconnections des opérateurs, qui peuvent limiter les possibilités offertes par la structure de l'unité arithmétique [13]. Si le multiplieur et les additionneurs sont placés en parallèle, ils sont reliés par des bus connectés à chaque opérateur (*fig.* 3). Ces bus, que nous nommons bus généraux puisqu'ils ont plusieurs sources, permettent une grande flexibilité d'emploi. Cependant ils coûtent plus cher en surface de silicium que des bus particuliers, nommés ainsi car ils ne possèdent qu'une source : longueur propre plus importante et présence de multiplexeurs pour les choix de direction.

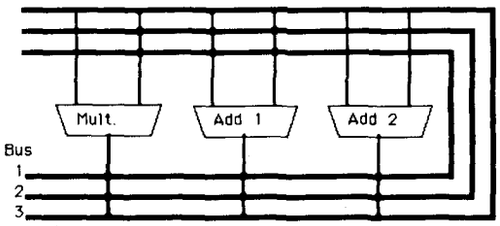


Fig. 3. – Structure parallèle.

En conséquence les opérateurs seront « pipelinés » pour réduire le coût des interconnexions. Les trois phases du calcul du papillon permettent de savoir qu'il faut placer le multiplieur puis les additionneurs, il n'en serait pas de même si nous avions choisi le calcul d'une TFR à entrelacement fréquentiel. Nous voyons apparaître les rôles spécifiques des opérateurs, où le multiplieur et l'additionneur n° 1 servent à faire une multiplication complexe, tandis que l'additionneur n° 2 permet une addition complexe.

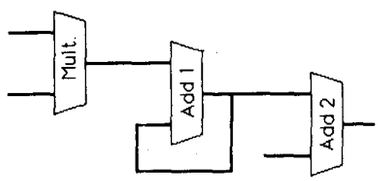


Fig. 4. – Structure pipelinée.

3.3. Implantation physique

Nous comptons utiliser le schéma 4 pour réaliser un placement initial des opérateurs. En effet, l'évaluation semi-graphique d'une distance moyenne des échanges entre opérateurs doit aussi nous donner un critère très précieux de préplacement de ces opérateurs pour une CAO de placement routage itérative. Ce point est en cours d'approfondissement.

4. Transferts de données entre UA et mémoires

Le transfert des données s'effectue sur des bus qui représentent une surface non négligeable. S'il n'est pas possible ici de calculer cette surface dépendant du placement des opérateurs sur la puce, cette surface croît avec le nombre de bus à implanter. S'occuper de minimiser le nombre de bus reviendra à augmenter le rendement UTS .

Nous considérons qu'il existe deux types de variables transitant dans l'unité arithmétique :

– les variables locales à cette unité, transférées d'un opérateur à un autre. Elles ont leurs chemins déjà tracés par les liaisons entre les opérateurs puisque nous avons construit l'architecture « pipelinée » en fonction de ces variables;

– les variables globales, provenant ou partant vers une mémoire externe. Elles définissent les liaisons entre l'unité arithmétique et l'environnement de celle-ci (mémoires, ports, etc.).

La meilleure utilisation des bus de communication entre l'unité arithmétique et son environnement se fera par une occupation quasi permanente de ceux-ci. Pour obtenir cette optimisation, nous sommes passés par les étapes suivantes :

(*a*) pour une structure de l'unité de calcul donnée, nous pouvons calculer le nombre de transferts de variables globales nécessaires par temps de cycle pour « alimenter » correctement les opérateurs;

(*b*) nous modifions alors la structure afin de réduire ce nombre par une mémorisation locale de certaines variables utilisées à peu d'intervalles;

(*c*) les contraintes temporelles associées au matériel utilisé définissent le multiplexage temporel du bus. Dans notre exemple durant le temps de multiplication, le bus peut avoir un cycle de lecture puis un cycle d'écriture;

(*d*) le bilan des trois premiers points définit alors le nombre minimal de bus qu'il faut connecter à la structure pour optimiser son rendement.

Nous précisons les considérations sur l'étude du papillon.

(*a*) Le décompte des flots de données des équations présentées sur la figure 2 fait apparaître 20 lectures et 4 écritures pour effectuer le papillon en 4 temps de cycles. Cela donne 5 transferts par cycle en lecture et 1 en écriture. Il s'agit d'un nombre moyenné de transferts entre des sources et des destinations diverses, donc nécessitant l'utilisation de bus généraux. Dans ce cas le rendement temporel d'un bus général est supérieur au rendement d'un grand nombre de bus particuliers.

(*b*) Pour diminuer le nombre de transferts, nous devons rendre locales toutes les variables arrivant dans l'unité arithmétique et devant être réutilisées à court terme. Le stockage des valeurs réelles et imaginaires à l'entrée des opérateurs permettra de limiter les transferts à six lectures (X, Y, W) et quatre écritures par papillon. Pour un calcul réalisé en quatre cycles, nous devrons prévoir deux transferts par cycle en lecture, et un en écriture.

(*c*) Pour une mémoire à temps d'accès de 45 ns et temps d'écriture de 15 ns (adresse positionnée), compte tenu d'un temps de cycle de 125 ns (comprenant le temps de calcul du multiplieur et de transfert des registres), deux transferts par cycle sont réalisables sur un même bus : un en lecture et un en écriture.

(*d*) Nous devons donc concevoir deux bus généraux pour notre architecture, les bus 1 et 2 de la figure 5. Remarquons que l'un d'entre eux peut se réduire à n'effectuer qu'un transfert entre la mémoire des coefficients et l'entrée du multiplieur.

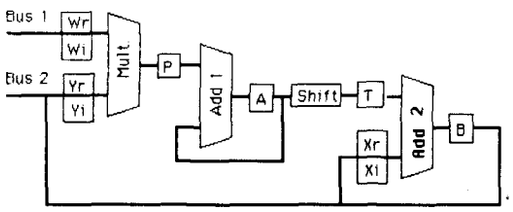


Fig. 5. — Architecture de l'unité arithmétique à deux additionneurs pour un rendement UTS optimal.

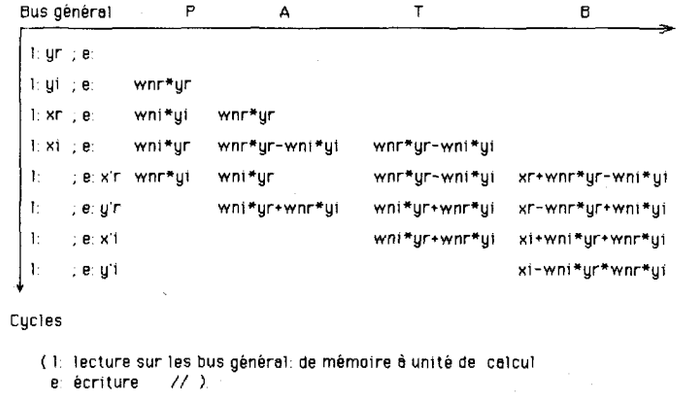


Fig. 6. — Occupation des bus et des registres de l'unité arithmétique à deux additionneurs lors du calcul du papillon.

Nous avons représenté sur la figure 6 le fonctionnement du calcul d'un papillon en visualisant le contenu des registres et du bus 2 de la figure 5. Il permet de vérifier la bonne marche du calcul en quatre cycles, et la possibilité d'imbriquer les calculs successifs de deux papillons.

La définition de l'architecture de l'unité arithmétique ne sera complète qu'après avoir évalué pour l'ensemble son rendement UTS . Cette étude complémentaire permet de vérifier si un opérateur ou un bus doit être redéfini dans le cas de sous- ou sur-utilisation vis-à-vis des autres fonctions.

5. Validation de l'architecture

5.1. Évaluation graphique du rendement : exemple du TMS 32010

Nous venons de définir une structure de l'unité arithmétique pour la TFR avec le souci d'augmenter le rendement UTS . Nous avons tenu compte de cet impératif lors de l'analyse des deux thèmes principaux de l'étude : choix et mise en place des opérateurs,

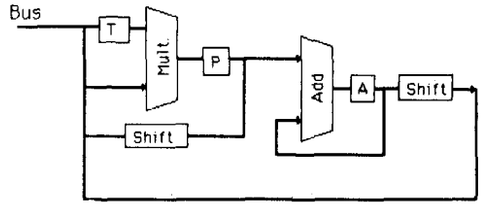


Fig. 7. — Architecture de l'unité arithmétique du TMS 32010.

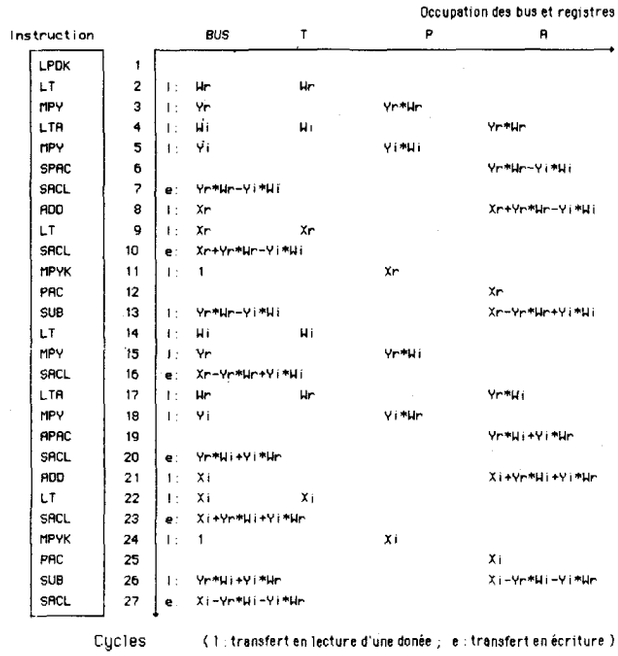


Fig. 8. — Programmation d'un papillon sur le TMS 32010 et occupation des registres.

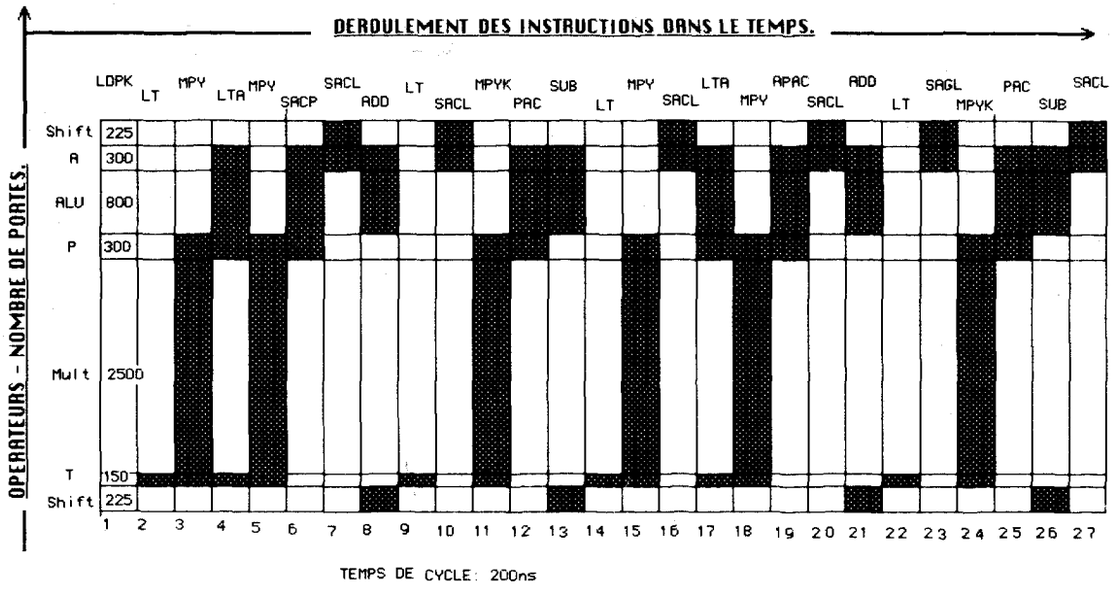


Fig. 9. — Évaluation du rendement UTS du TMS 32010 : rendement global de 29 %.

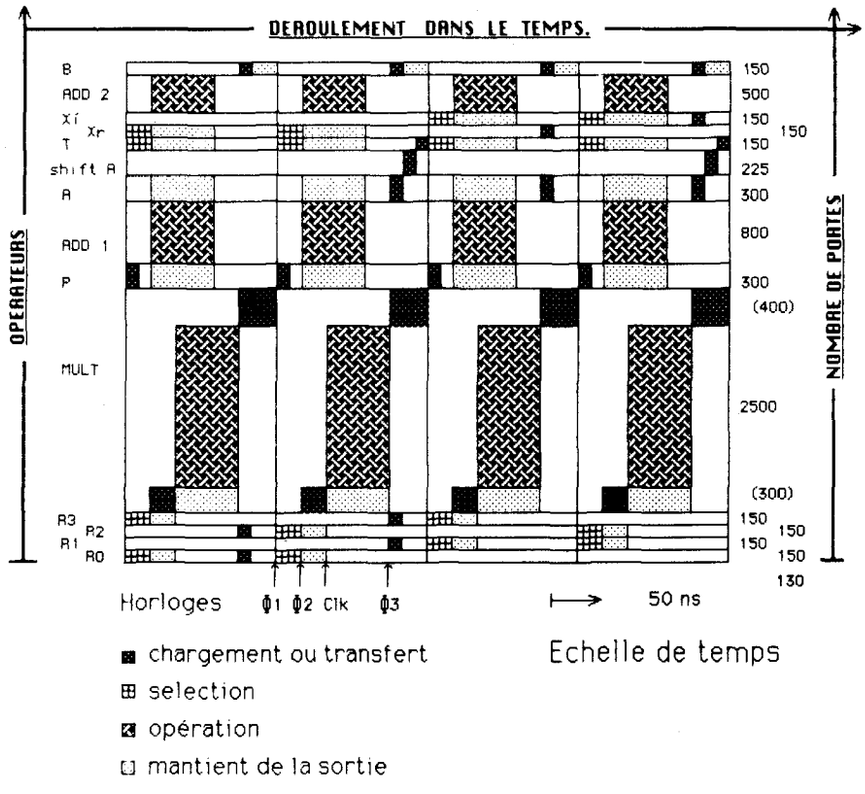


Fig. 10. — Évaluation du rendement UTS de l'architecture à deux additionneurs : rendement exact de 38 %.

transfert des données. Nous nous proposons maintenant de visualiser ou d'estimer graphiquement les rendements obtenus.

Pour cela, nous représentons sur un chronogramme l'utilisation des opérateurs et des registres de l'unité arithmétique. L'échelle verticale est proportionnelle au nombre de portes que représente l'opérateur. L'échelle horizontale est proportionnelle au temps. Une surface sur le graphe est le produit d'un nombre de portes d'un opérateur par sa durée d'utilisation; le rendement UTS exact est alors le rapport des surfaces hachurées du graphe sur la surface totale.

A titre d'exemple, essayons d'évaluer le rendement UTS d'un produit du commerce, le TMS 32010 de

Texas Instrument, pour la même jauge. La figure 8 décrit l'occupation des registres du TMS 32010 au cours des cycles d'un papillon [8]. Le calcul s'effectue en 27 cycles et occupe six fois le multiplieur puisque les stokages temporaires à l'intérieur de l'unité de traitement sont limités.

La figure 9 représente le graphe surface fonction du temps pour le TMS 32010, qui donne un rendement UTS de 29 %. Ce rendement ne correspond pas exactement à la définition qui a été faite du rendement UTS, dans la mesure où nous n'avons pas tenu compte des temps exacts de calcul ou de traversée des registres (la valeur calculée par ce rendement est supérieure à celle calculée par le rendement UTS). Nous définissons le *rendement UTS global* en considérant un opérateur utilisé durant un cycle entier des lors qu'il est utilisé en cours de cycle.

5.2. Comparaison du choix des opérateurs

La figure 10 représente le graphe d'utilisation de la structure à deux additionneurs (*fig.* 5) pour un séquencement synchronisé à quatre horloges.

A fin de comparaison, nous avons représenté sur la figure 11 l'architecture calculant les papillons radix 2 dans le cas où le concepteur s'impose de n'intégrer qu'un additionneur, et sur la figure 12 le séquencement correspondant, en visualisant le contenu des registres. Le calcul du papillon s'effectuera en huit cycles. Nous avons représenté sur la figure 13 le graphe du rendement de l'architecture de la figure 11. La figure 13 fait apparaître le goulot d'étranglement formé par l'additionneur unique qui est le seul opérateur utilisé à chaque cycle dans cette structure. Cela montre la nécessité de l'architecture retenue dès le paragraphe 3.

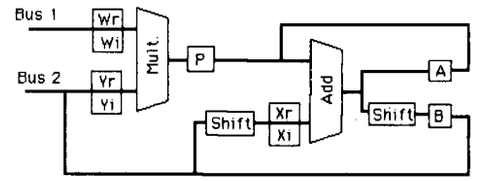


Fig. 11. – Architecture de l'unité arithmétique à un additionneur, optimisée grâce au rendement UTS.

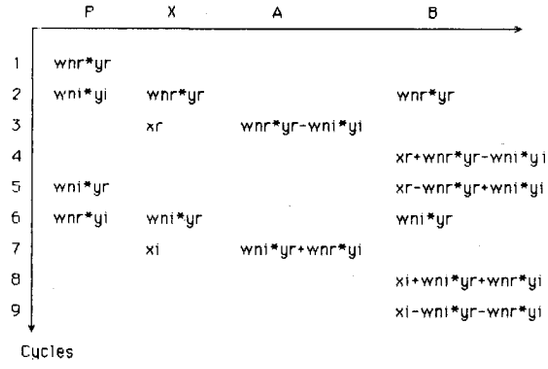


Fig. 12. – Occupation des registres de l'unité arithmétique à un additionneur lors du calcul du papillon.

Cette structure nécessite environ 5 700 portes pour l'unité arithmétique, alors que celle comportant deux additionneurs en nécessite 5 850. Le résultat est que pour une augmentation de moins de 3 % du nombre de portes, nous divisons par deux le temps de calcul du papillon. Le rendement UTS est de 38 % pour l'architecture à deux additionneurs, alors qu'il n'est que de 22 % pour une architecture à un additionneur. La chute des rendements observés par rapport à ceux calculés au paragraphe 3.1 provient de l'introduction des divers registres (*fig.* 5) qui augmentent le temps de cycle.

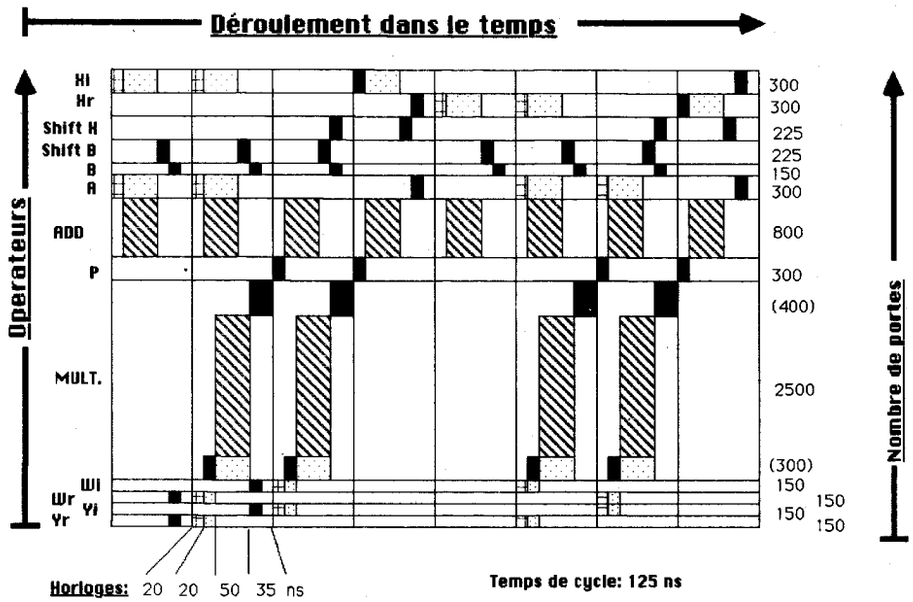


Fig. 13. – Évaluation du rendement UTS de l'architecture à un seul additionneur : rendement exact de 22 %.

5.3. Comparaison des possibilités de transfert de données

Nous comparons maintenant deux structures possibles de l'unité de calcul, celle du TMS 32010 (*fig.* 7) et celle d'une unité optimisée comportant aussi un multiplieur et un seul additionneur (*fig.* 11). La différence se joue sur les possibilités de transfert des données et sur les possibilités « soft » de gérer ces transferts.

Nous avons calculé un rendement global du TMS 32010 de 29 %. Pour pouvoir comparer ce rendement à celui de l'architecture à un additionneur, nous recalculons son rendement global : celui-ci est de 59 %. Le lecteur retrouvera aisément cette valeur à l'aide de la figure 13 et de la définition du rendement global.

L'évaluation graphique permet non seulement une comparaison des rendements UTS en fonction de la définition des opérateurs, mais aussi en fonction de l'étude des transferts des données. S'il est difficile *a priori* de chiffrer le rendement UTS au niveau des bus, l'étude de ceux-ci permet cependant une nette amélioration du rendement UTS de la partie opérative de l'unité de calcul.

6. Conclusion

Le souci de rentabiliser la surface de silicium d'un circuit intégré est très important pour un concepteur [6, 9, 10]. En effet, il conduira non seulement à minimiser la surface totale en supprimant des fonctions peu ou mal utilisées, mais aussi à augmenter la rapidité de l'architecture. C'est avec cet objectif que nous avons tout d'abord conçu une structure de l'unité de calcul pour l'algorithme de transformée de Fourier rapide. A chaque point de la conception, une évaluation locale du rendement UTS permet d'éliminer les solutions peu rentables.

A titre d'exemple, nous avons résumé dans le tableau les rendements exacts (prise en compte du temps exact de calcul par opérateur) et les rendements globaux (prise en compte du temps de cycle par opérateur) des différentes architectures rencontrées.

Tableau IV

Rendements UTS exacts et globaux des différentes architectures

	TMS 320	1 add.	2 add.
Rendement exact.	X	22 %	38 %
Rendement global.	29 %	59 %	94 %
Nombre de cycles.	27	8	4
Temps de calcul.	5,4 ms	1 ms	0,5 ms

Le rendement global permet une approche immédiate de la comparaison. Il a pour vocation d'éliminer une architecture à trop faible rendement, mais il reste insuffisant pour une étude précise de l'utilisation des opérateurs. Seul le rendement exact permet une approche détaillée du fonctionnement au niveau du choix et du placement des opérateurs, de leurs liaisons et de leurs séquencements.

Outre une meilleure conception, le rendement UTS permet une validation de l'ensemble de l'architecture. Cette vérification mène à la détection de sous- ou sur-utilisation de fonctions. Le lecteur poura vérifier en comparant les figures 10 et 13 que l'additionneur unique est une fonction sur-utilisée pour cette solution.

L'évaluation graphique du rendement exact permet de conduire une étude détaillée du séquencement de l'architecture. Cette étude, que nous n'avons pas rapportée ici, consiste à éliminer les « temps morts » des opérateurs durant un cycle machine en ajoutant des horloges adéquates.

En conclusion, nous pouvons résumer les trois impacts de l'étude du rendement temporel du silicium :

– assister la conception d'une architecture dans le choix des opérateurs à implanter;

– vérifier l'utilisation véritable des opérateurs implantés;

– améliorer alors le séquencement interne par redéfinition des horloges.

L'augmentation du rendement entraîne une meilleure utilisation du silicium, et mène à une

– augmentation des performances temporelles de l'architecture étudiée.

Et/ou une

– diminution de la surface totale du silicium facilitant l'intégration de l'architecture.

Manuscrit reçu le 3 *décembre* 1985.

BIBLIOGRAPHIE

[1] L. R. Rabiner et B. Gold, *Theory and application of digital signal processing*, Prentice Hall, 1975.

[2] A. Oppenheim, *Application of digital signal processing*, Prentice Hall, 1978.

[3] M. Bellanger, *Traitement numérique du signal;* Masson, 1980.

[4] T. Nishitani, R. Maruta, Y. Kawakami et H. Goto, A single-chip digital signal processor for telecommunication application, *IEEE trans.*, SC-16, n° 4, août 1981.

[5] Bowen et Brown, *VLSI systems design for digital Signal processing*, Prentice Hall, 1982.

[6] D. Karlin, VLSI Building Blocks for Digital Signal Processing, *Proc. IEEE Int. Conf. ASSP*, 1982.

[7] G. D. Covert, A 32 point monolithic FFT processor chip, *Proc. IEEE Int. Conf. ASSP*, 1982.

[8] S. Magar, R. Hester et R. Simpson, Signal-processing mC builds FFT-based spectrum analyzer, *Electronic Design*, août 1982.

[9] B. NEW et D. BRAIN, Bipolar VLSI Facilitates Fourier Transformation, *Proc. IEEE Int. Conf. ASSP*, 1982.

[10] R. E. OWEN, VLSI architecture for digital signal Processing, *VLSI Design*, juin 1984.

[11] M. GINDRE et D. HOSTE, Processeur FFT temps réel utilisant la famille AMD 29500 : Principes généraux et composants, *Mini et Micro*, n° 218, octobre 1984.

[12] M. GINDRE et D. HOSTE, Processeur FFT temps réel utilisant la famille AMD 29500 : Exemple et réalisation, *Mini et Micro*, n° 223, décembre 1984.

[13] M. GINDRE et D. HOSTE, Processeur FFT temps réel utilisant la famille AMD 29500 : Microprogrammation, *Mini et Micro*, n° 225, janvier 1985.

[14] M. MEHALIC, P. RUSTAN et G. ROUTE, Effects of architecture Implementation on DFT Algorithm Performance, *IEEE Trans.*, ASSP-33, n° 3, juin 1985.

[15] L. GUNDEL, A noval high-speed Fourier-vector processor, *Signal Processing*, 9, 1985, p. 107-120, North-Holland.

[16] Data book TRW, *TRW lsi production*, 1984.

[17] NEC, *Workshop digital signal processor mPD* 7720, 1984.

[18] Texas Instrument, *TMS* 320 *digital signal processor family*, 1984.