Adéquation d'un algorithme à une architecture, application à la transformée de Fourier

Adequacy of an Algorithm to an Architecture, an Application to the Fast Fourier Transform

par Jean-Luc PHILIPPE, Olivier SENTIEYS*, Eric MARTIN**, Hélène DUBOIS***

*LASTI-ENSSAT, Université de Rennes, 6, rue de Kérampont F-22300 Lannion
**LESTER – UBS – 10, rue Jean Zay, F-56100 Lorient

résumé et mots clés

La prise en compte des caractéristiques de l'architecture cible est souvent effectuée tardivement lors du développement d'un algorithme de TDSI. De plus, la nature de l'architecture ainsi que la nature des contraintes liées à l'application pouvant varier au cours du temps, la difficulté de la maîtrise de cette interaction entre architecture et algorithme s'en trouve largement accrue. Nous proposons dans cet article, une formalisation de la démarche de conception qui permet de manière interactive, par le calcul de métriques, de guider les transformations à opérer sur un algorithme afin d'optimiser son adéquation à une architecture. La démarche sera exposée pour un algorithme orienté traitement, et une architecture comprenant des processeurs programmables auxquels seront associés, selon les besoins, des circuits dédiés.

Architecture hétérogène, Cycle de conception, Efficacité, FFT, multiprocesseur, Synthèse architecturale, Transformations.

abstract and key words

The characteristics of the target architecture are very often taken into account late, at the time of the implementation step of a signal processing application design flow. Furthermore, the target architecture and the constraints may be time evolving, making the adequacy between architecture and algorithm, more and more difficult. In this paper, we propose a methodology that aims to guide algorithmic transformations by computing some metrics, thus improving the adequacy. The approach will be explained on a compute bound algorithm. The results are based on MIMD architecture, and heterogeneous architecture by the use of Asic.

Heterogeneous architecture, Design cycle, Efficiency, FFT, Multiprocessor, High level synthesis, Transformations.

1. introduction : CAO d'architectures et traitement du signal

Pour effectuer une classification de la conception des systèmes électroniques, le diagramme en Y [1] est une référence. Sa principale idée est que tout élément d'un système peut être représenté à l'aide de trois domaines : comportemental, structurel et physique (figure 1). Dans le premier domaine, le comportement du système est décrit sans référence à l'architecture cible. Dans le second, le système est représenté par une hiérarchie d'éléments fonctionnels et leurs interconnexions. Les transitions dans ce diagramme permettent de définir aisément les différentes tâches de conception.

La mise en œuvre matérielle de la spécification d'un système de traitement du signal, conduit à réaliser un certain nombre de transformations, aussi bien dans le domaine comportemental que structurel. La synthèse est le processus qui permet de changer de domaine tout en restant au même niveau d'abstraction, ou d'affiner le niveau d'abstraction, tout en restant dans le même domaine. L'analyse quant à elle, est le cycle qui permet, à niveau d'abstraction constant, de changer de domaine, ou d'abstraire le niveau de représentation, tout en restant dans le même domaine [2].

Ainsi, tout en restant dans le domaine comportemental, la première transformation consistera à décrire les spécifications systèmes à un niveau d'abstraction inférieur sous la forme d'une association d'algorithmes. Sur ceux-ci, un certain nombre

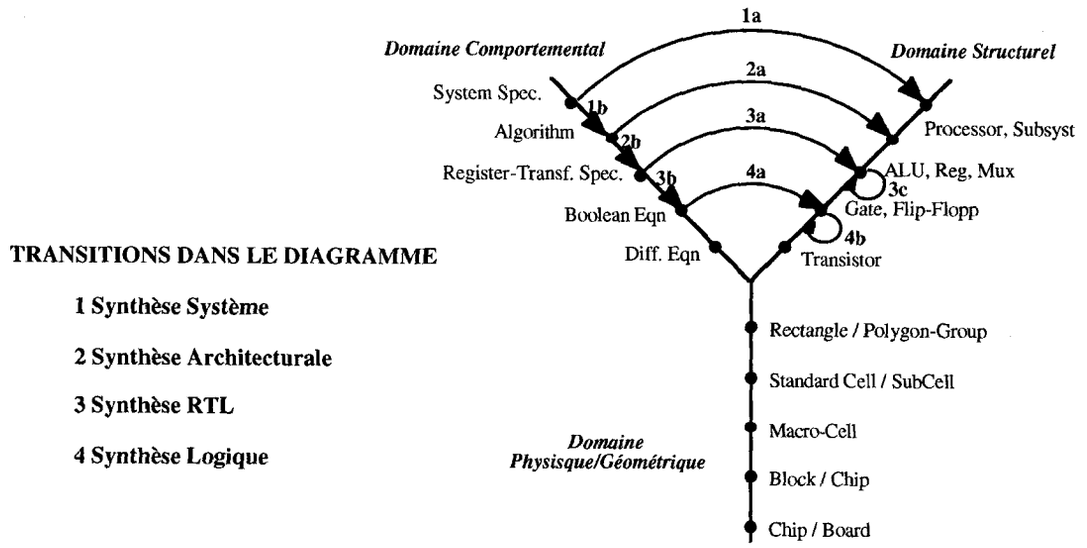


Figure 1. – Cycle de conception, diagramme en Y.

d'optimisations pourront être réalisées en fonction de critères comme la qualité, la complexité, la consommation, ... Ce raffinement du niveau d'abstraction, peut être réalisé sans la connaissance de l'architecture cible. Toutefois, une vue modélisée de celle–ci permet de calculer des estimateurs plus précis, qui permettront de guider plus efficacement les sélections et transformations à mettre en œuvre [3]. Cette architecture cible peut être figée, et il conviendra alors de chercher la meilleure adéquation entre les algorithmes et l'architecture existante. De manière duale, l'architecture pourra être adaptée à l'algorithme, lorsque elle est totalement libre, ou du moins lorsqu'elle ne doit respecter qu'un modèle générique, sans la description exacte ni du nombre ni de la nature des ressources mises en œuvre. Une dernière approche, que l'on qualifie de développement conjoint logiciel/matériel, vise à définir des flots de conception permettant de réaliser conjointement les aspects logiciel et matériel.

Le cycle de conception qu'il faut donc définir doit permettre l'exploration d'un espace de solutions, qui doit bien souvent être en plus parcouru sous contraintes : consommation, performances, coût du système. De plus, ce cycle de conception doit pouvoir être exécuté rapidement. En effet, lors de l'apparition de nouveaux systèmes, comme les nouveaux services téléphoniques, il s'agira de prouver rapidement la faisabilité technologique, ou de comparer entre elles différentes propositions. Enfin, tout au long du cycle du produit ainsi réalisé, on sera conduit à lui faire subir un certain nombre de migrations technologiques. Dans ce but, le cycle de conception doit également offrir, par une élévation du niveau de représentation des systèmes, une pérennité aux spécifications qui peuvent ainsi plus facilement évoluer au cours du temps.

Il faut donc offrir aux spécialistes du domaine, qu'ils soient chargés du développement comportemental ou structurel, des méthodes qui les guident dans l'exploration de l'espace de conception. Celles ci ne doivent pas opposer analyse et synthèse, mais doivent utiliser ces deux approches de manière conjointe.

La méthode de conception, exposée dans cet article, a pour objet de permettre la conception de systèmes multiprocesseurs sous contraintes du temps d'exécution, et de la minimisation du coût de mise en œuvre. Elle doit également permettre le prototypage d'applications, c'est à dire l'évaluation rapide de la complexité d'un nouveau système, ainsi que la comparaison de différentes solutions en travaillant aussi bien dans les domaines comportementaux que structurels. Afin de limiter l'espace de recherche, l'architecture générique retenue est composée de plusieurs processeurs programmables, auxquels, en fonction des besoins, seront associés des processeurs dédiés (ASIC). Lorsque les contraintes comme les performances ou la consommation seront très sévères, la possibilité de sélectionner une architecture complètement dédiée, devra être envisagée. Ce choix se justifie par le domaine d'application, traitement du signal et télécommunications, pour lequel les algorithmes sont d'une part fortement contraints en temps, et d'autre part, fortement orientés calcul, avec parfois des tâches de contrôle. Ce choix est également corroboré par les solutions industrielles existantes, souvent composées d'un nombre relativement restreint de processeurs, de traitement du signal ou non, d'architectures mixtes intégrées comme les processeurs vidéo ou encore les ASIP (*Application Specific Instruction set Processor*) [4], d'architectures mixtes non intégrées (association de processeurs de contrôle, de processeurs de calcul, de dispositifs d'entrées/sorties).

Après avoir présenté dans le deuxième chapitre la démarche générale, cet article présente dans le troisième chapitre l'application qui servira de support à notre exposé : Le filtrage basé sur des transformées de Fourier. Pour cette application un partitionnement matériel/logiciel est proposé. Pour des raisons de minimisation du coût de mise en œuvre, ce premier partitionnement favorise l'approche logicielle et aboutit à une proposition de pa-

rallélisation de l'algorithme sur une architecture complètement programmable dont les performances sont prévues par analyse (chapitre 4). Cette analyse permet de rétroagir sur l'algorithme par transformations, ou encore, de comparer plusieurs solutions algorithmiques (chapitre 5). Lorsque cette approche logicielle ne permet pas de satisfaire les contraintes imposées par l'application à un coût raisonnable, la migration de certaines fonctionnalités de l'algorithme vers du matériel dédié, doit être envisagée. Cet axe, utilisant les méthodes de synthèse architecturale, est présenté dans le chapitre 6. Pour une fonctionnalité donnée, une architecture reposant sur un modèle coeur de processeur de traitement du signal est synthétisée. L'association de ces deux démarches, analyse et synthèse, permet de définir un cycle de conception d'architecture hétérogène. Le septième chapitre, constitue une conclusion, mais fournit également quelques perpectives sur le sujet de la conception des systèmes dédiés, et son élargissement à d'autres domaines d'applications.

2. cycle de conception

2.1. présentation générale

Comme il a été précisé dans l'introduction, il ne s'agit pas d'opposer analyse et synthèse, mais plutôt de formaliser un cycle de conception utilisant ces deux démarches conjointement. Ainsi, et suivant la figure 2, depuis un cahier des charges spécifiant la nature de l'application (1), pour un modèle d'architecture (5) et un algorithme (3) choisis, il sera possible de produire une architecture répondant au modèle sélectionné (7), mais aussi, sans aller jusqu'à l'obtention de cette architecture, de rétroagir en modifiant les contraintes liées à l'application (2), de changer le modèle d'architecture ou le modifier (6), de sélectionner un nouvel algorithme ou de le transformer (4).

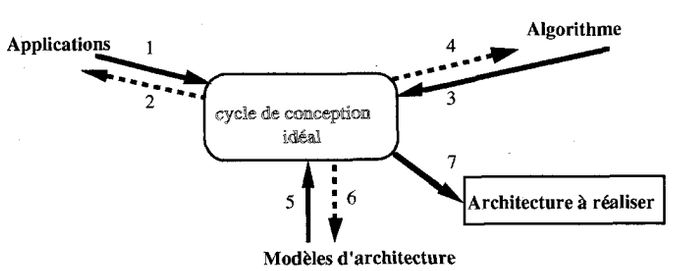


Figure 2. – Cycle de conception utilisant l'analyse et la synthèse.

Le cycle de conception revient donc à parcourir un espace de solution, l'axe contraintes pouvant lui même être décomposé en plusieurs sous-axes, qui pourront selon les domaines d'application, être privilégiés les uns par rapport aux autres.

Dans la méthodologie traditionnelle de conception de systèmes qui intégrent des composantes logicielles et matérielles, les différentes parties sont conçues de manière indépendante et ce, dès les premières étapes. Cette approche restreint les possibilités d'exploration des solutions mixtes, dans lesquelles, certaines fonctionnalités peuvent migrer du logiciel vers le matériel ou vice–versa. Or les dernières avancées technologiques ont permis des développements de composants spécifiques (ASICs) pour un coût et dans un temps raisonnables. Ceci suggère une méthodologie de conception plus souple, où le logiciel et le matériel peuvent interagir et donc être conçus parallèlement. Une telle approche doit permettre la réalisation de systèmes dédiés à des applications spécifiques (traitement du signal, télécommunications, ...) où le choix d'implantation matériel-logiciel peut évoluer en fonction des besoins.

Dans ce cycle, illustré par la figure 3, pour un partitionnement donné, la démarche développée pour la partie programmée, consiste à prédire le temps d'exécution d'un algorithme parallélisé (mode SPMD) sur une architecture MIMD à passage de messages. A partir de cette prédiction, il s'agit de proposer diverses modifications, aussi bien sur l'algorithme que sur l'architecture, permettant d'accroître les performances jusqu'à satisfaire les contraintes de temps et de coût. Cette prévision est effectuée à partir d'une analyse du code séquentiel de l'algorithme, donc au niveau nommé composant sur la figure 3. Cette prédiction repose sur une triple modélisation : processeur, réseau et algorithme. Les transformations pouvant alors être réalisées sont présentées dans le paragraphe suivant.

En fonction des contraintes, une partie de l'application peut être intégrée dans une architecture dédiée, concue par synthèse architecturale. Celle-ci constitue un thème essentiel pour le domaine de l'adéquation architecture algorithme. Elle a pour objet, à partir d'une description comportementale d'un algorithme, d'une bibliothèque de ressources, d'un énoncé des contraintes, et sans utiliser aucune directive architecturale, de synthétiser une architecture matérialisant cet algorithme. Dans la suite de l'article cette architecture est de type processeur de traitement du signal [5]. Des développements sur l'estimation d'architectures dédiées sont en cours actuellement. Celle–ci est réalisée au niveau composant, donc en amont de la synthèse et permet de guider les transformations sur l'algorithme à synthétiser [6]. Dans les paragraphes suivants cette démarche de conception sera utilisée afin de chercher à renforcer l'adéquation d'un algorithme à l'architecture ciblée.

2.2. les transformations

Un algorithme est souvent spécifié sans connaissance totale du matériel sur lequel il sera implanté. Afin de raccourcir le cycle de mise en œuvre il est alors indispensable de pouvoir aider le concepteur dans les transformations à réaliser sur l'algorithme développé, en tenant compte des connaissances partielles que l'on a sur l'architecture. Ces transformations peuvent être réalisées afin

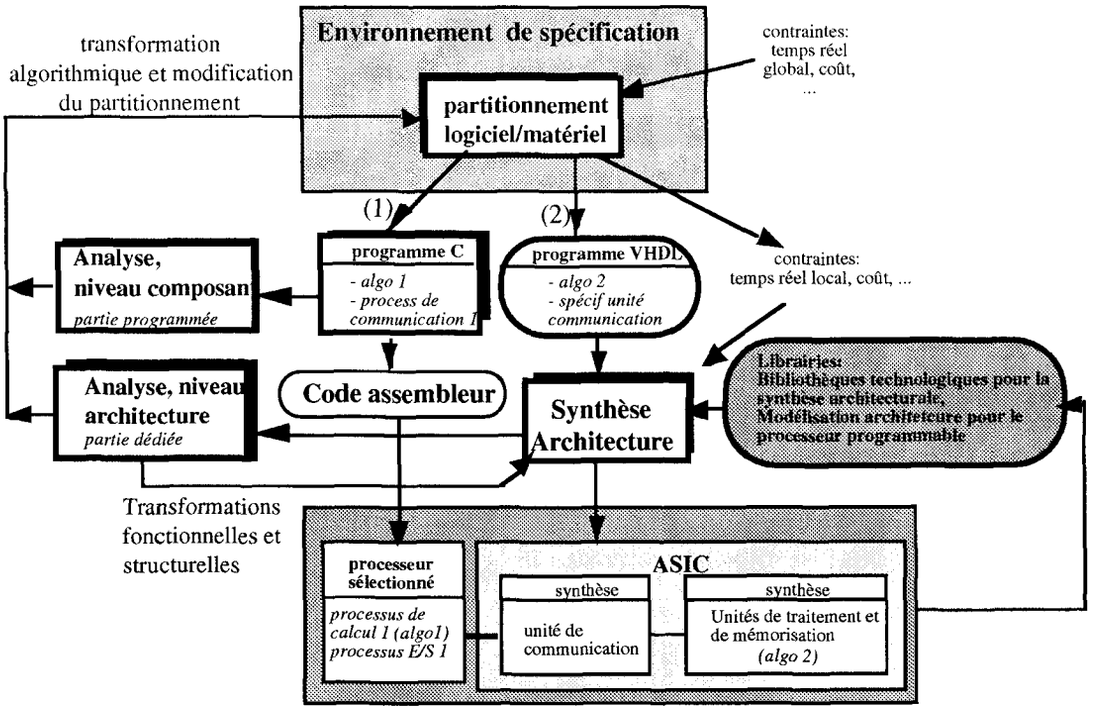


Figure 3. – Cycle de conception conjoint logiciel/matériel, permettant la conception d'architectures hétérogènes.

de minimiser, de maximiser, ou de borner des critères comme les performances, le coût, la consommation [7]. La définition de ces transformations pose de nombreuses questions telles leur niveau de réalisation, le type d'informations disponibles pour leur mise en œuvre, et leur pertinence, c'est-à-dire, le gain qu'elles apporteront par rapport au temps nécessaire à leur développement.

Trois classes de transformations sont exposées dans cet article : structurelle, fonctionnelle et algorithmique. Les deux premières sont plus particulièrement adaptées à des architectures cibles de type dédiées.

• Les transformations structurelles

Un algorithme peut être décrit sous différents modèles de type flot de données. Les techniques de transformation [8], [9], [10], [11], sont pour la plupart basées sur des notions d'associativité, de distributivité, de commutativité, de *retiming*, de redondance de calcul. Leur mise en œuvre permet par exemple de réduire le chemin critique du graphe.

• Les transformations fonctionnelles

Un nœud de graphe peut être matérialisé par différents opérateurs. Par exemple une multiplication peut être réalisée par un multiplieur, ou programmée sur l'association d'un additionneur et d'un décaleur. D'une manière générale, il existe plusieurs équivalences entre opérateurs matériels, ou entre opérateurs matériels et mixtes (matériel et logiciel) qui, fonctionnellement, produiront le même résultat mais n'auront pas le même coût, les mêmes performances, ...

Lorsque les ressources sont imposées, ce qui est le cas pour les processeurs programmables, ces transformations seront limitées par le jeu d'instructions du processeur.

• Les transformations algorithmiques

Une fonction de TDSI peut être réalisée en utilisant différentes versions algorithmiques qui diffèrent par des critères de qualité, de complexité. C'est par exemple le cas des algorithmes rapides qui cherchent à réduire la complexité calculatoire, comme par exemple la transformée de Fourier rapide, pour lequel plusieurs graphes de fluence existent (changement de base, de géométrie,...). A l'opposée des autres transformations, celles–ci peuvent conduire à simuler à nouveau le nouvel algorithme afin de vérifier son équivalence fonctionnelle avec le précédent.

3. la Transformée de Fourier Rapide (TFR)

3.1. domaine d'application : filtrage en annulation d'échos acoustiques

Deux applications d'annulation d'écho acoustique vont servir d'illustration à cette étude. La première concerne les communications « main libre » qui se caractérisent par des longueurs de réponse impulsionnelle courtes (une centaine de points), et une fréquence d'échantillonnage standard à 8 khz. La seconde application a été retenue pour les fortes contraintes qu'elle impose

à la conception d'architectures dédiées. Il s'agit de l'annulation d'écho acoustique en téléconférence, caractérisée par des filtres à réponse impulsionnelle longue (de l'ordre du millier de points) et des fréquences d'échantillonnage en bande élargie (16 khz). Ces deux applications mettent en œuvre les mêmes algorithmes de filtrage.

Deux grandes familles d'algorithmes de filtrage adaptatif sont étudiées : la famille des algorithmes du gradient stochastique (LMS) et la famille des moindres carrés récursifs (RLS). Actuellement, de nombreux travaux de recherche visent à réduire la complexité arithmétique des algorithmes de filtrage adaptatif. Le thème abordé consiste à évaluer la relation pouvant exister entre la réduction de complexité arithmétique d'un algorithme et la réduction en surface de silicium d'un circuit dédié conçu pour sa mise en œuvre.

La réduction de complexité arithmétique passe par l'utilisation de techniques mathématiques comme l'approximation de l'inversion de matrice (FTF8L), le calcul du produit de convolution en bloc dans le domaine fréquentiel (FLMS), la décomposition en bancs de filtres (GMDF). On trouvera dans [12] [13], une comparaison de la complexité de ces algorithmes. La suite de cette étude utilisera le FLMS dont la description est donnée figure 4. La suite de cette étude concernera la partie la plus complexe de cet algorithme : la transformée de Fourier rapide (FFT).

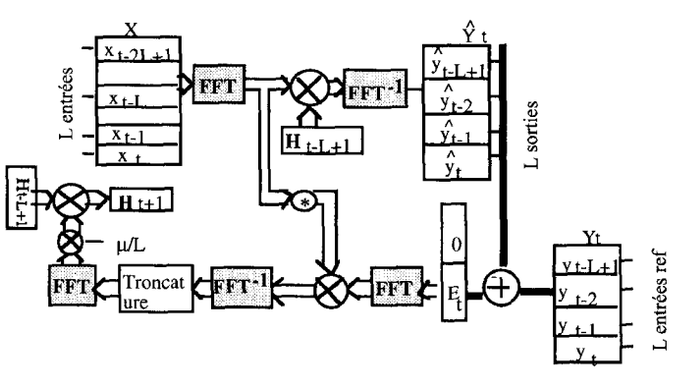


Figure 4. – Synoptique du filtre FLMS.

3.2. principes de la FFT

La transformation de Fourier rapide (TFR), ou encore Fast Fourier Transform (FFT), est directement issue d'une réorganisation du calcul de la transformée de Fourier discrète (TFD) dont la définition est donnée ci dessous. $x(n)$ et $X(k)$ sont, dans le cas général des nombres complexes.

$$X(k) = \sum_{n=0}^{I-1} x(n) \cdot e^{-2j\pi kn/I},\ k = 0, 1 \ldots I-1 \qquad (1)$$

Le principe fondamental de la TFR repose sur la décomposition du calcul d'une séquence de I échantillons en TFD successives sur un nombre inférieur de points [14]. Selon la manière dont ce principe est mis en œuvre, différents algorithmes sont obtenus, tous comparables au niveau de leur complexité :

- TFR partagée dans le temps ou DIT : $x(n)$ est décomposé en sous séquences.
- TFR partagée en fréquences ou DIF : $X(k)$ est décomposé en sous séquences.

• TFR partagée en fréquences (Decimation In Frequency : DIF)

Depuis la formule (1), $X(k)$ est partagé en une somme sur les $I/2$ premiers termes et une autre sur les $I/2$ autres termes. Cet exemple se limitera au cas ou I est une puissance de deux. On posera $W_N = e^{-2j\pi/I}$.

$$X(k) = \sum_{n=0}^{I/2-1} x(n) \cdot W_I^{nk} + \sum_{n=I/2}^{I-1} x(n) \cdot W_I^{nk} \qquad (2)$$

On effectue une séparation des indices pairs et impairs :

$$\begin{aligned} X(2r) &= \sum_{n=0}^{I/2-1} \left[x(n) + x(n = \frac{I}{2}) \right] W_I^{2rn} \\ &= \sum_{n=0}^{I/2-1} \left[x(n) + x(n = \frac{I}{2}) \right] W_{I/2}^{rn} \qquad (3.1) \\ X(2r+1) &= \sum_{n=0}^{I/2-1} \left[x(n) - x(n = \frac{I}{2}) \right] W_I^{2rn} \cdot W_I^n \\ &= \sum_{n=0}^{I/2-1} \left\{ \left[x(n) - x(n + \frac{I}{2}) \right] W_I^n \right\} W_{I/2}^{rn} \qquad (3.2) \end{aligned}$$

Les équations 3.1 et 3.2 sont des TFD d'ordre $I/2$ sur des fonctions $g(n)$ et $h(n)$, avec :

$g(n) = x(n) + x(n + I/2)$, et $h(n) = x(n) - x(n + I/2)$, où $n = 0, \ldots, I/2 - 1$.

Le remaniement des équations conduit au calcul de deux TFD d'ordre inférieur selon le schéma de la figure 5.

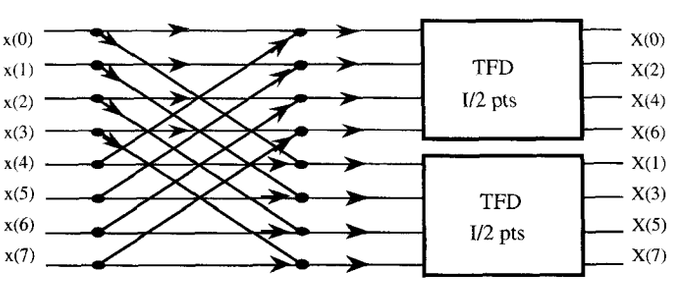
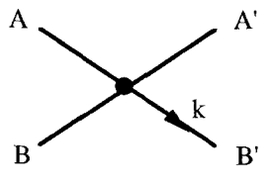


Figure 5. – Décomposition DIF de la TFD.

Comme $I/2$ est un nombre pair, chacune des TFD d'ordre $I/2$ peut être partagée en deux TFD d'ordre $I/4$ jusqu'à arriver à une TFD sur deux points (voir papillon figure 6) pour laquelle le graphe flot est de complexité $O(I)$. Finalement il en résulte un graphe flot possédant $\log_2 I$ étages du type de la figure 5;



$$\begin{cases} A' = A + B \\ B' = (A - B).W^k \end{cases}$$

Figure 6. – Papilon DIF de la TFR.

la complexité de la Transformée de Fourier est enfin réduite à $O(I \log_2 I)$.

Un papillon nécessite 1 multiplications et 2 additions/soustractions sur des nombres complexes. Le graphe complet (figure 7) nécessite le calcul de $\frac{I}{2} \log_2 I$ papillons ce qui donne donc en unités de calcul :

- $\frac{I}{2} \log_2 I$ multiplications de nombres complexes,
- $I \log_2 I$ additions/soustractions de nombres complexes,

ou

- $2I \log_2 I$ multiplications de nombres réels,
- $3I \log_2 I$ additions/soustractions de nombres réels.

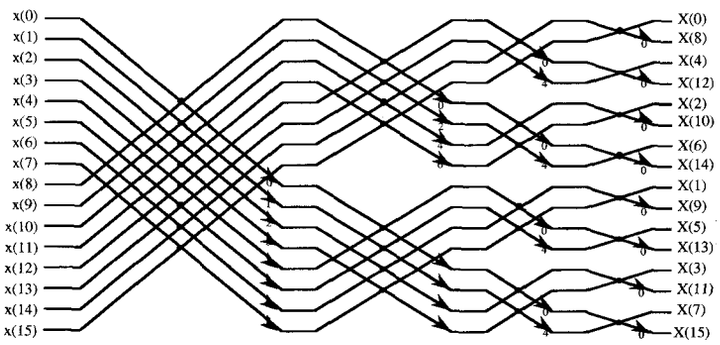


Figure 7. – NGRaphe de la TFRDIF sur 16 échantillons.

Cette complexité peut servir à évaluer le temps de calcul d'une TFR complète sur n'importe quel processeur.

A l'opposé de la TFD, la TFR génère des calculs d'adresse de complexité non négligeable. Entre deux étages m et $m+1$, les W^i ne varient pas linéairement avec les $x(i)$. De plus, il est nécessaire de présenter à l'algorithme les échantillons dans un ordre non régulier (bit–reverse) ce qui engendre un calcul d'adresse supplémentaire.

Le graphe de fluence choisi dans cette étude comme référence, est celui de la TFR radix 2 DIF. D'autres graphes existent, soit en changeant la base (TFR base 4), soit en changeant la géométrie du graphe (géométrie constante), ou encore en décimant dans le domaine temporel (TFR DIT). Certaines de ces transformations seront analysées dans la suite de cet article.

4. analyse de performances d'une architecture multiprocesseur

4.1. méthode de parallélisation de la FFT

La démarche d'analyse développée, consiste à prédire le temps d'exécution d'un algorithme parallélisé (mode SPMD) sur une architecture MIMD à passage de messages, et partant de là, à proposer diverses modifications permettant d'accroître les performances. La prévision est effectuée à partir d'une analyse du code séquentiel de l'algorithme. La prédiction repose sur une triple modélisation : processeur, réseau et algorithme. Ainsi, l'algorithme sera caractérisé par son taux de parallélisme ($p\%$), le temps d'exécution dans sa version séquentielle ($Tseq$), le nombre de données échangées entre les différentes tâches (Dp), la taille des messages ($Nmes$). Le réseau sera modélisé par la distance moyenne entre deux processeurs ($dmoy$) et son nombre de liens ($Nbbus$). Enfin, le processeur sera modélisé par son taux de communication (C), le temps nécessaire à échanger une donnée ($Tcom$), et la dégradation apportée par la réalisation des communications sur le calcul (Tnt). Diverses mesures sont alors prévues : efficacité, accélération, seuil de saturation, etc. (voir tableau 1). On trouvera dans [15], une description détaillée de la méthodologie appliquée au filtrage de Kalman. Ces différents paramètres sont

- **Temps de calcul idéal** : $Ti = \frac{T\,seq}{N}$
- **Temps de calcul** : $T\,\text{cal} = \left[(1-P) + \frac{P}{N}\right] : T\,seq$
- **Temps réel** :

$$T\,\text{réel} = T\,\text{cal} + T\,\text{nt} \bullet T\,com \frac{Dp}{N\text{mes}}\ \text{distance}$$

- **Temps des échanges transparents** :

$$T\,\text{ech} = \frac{N\,\text{tc}}{nbbus}\ \text{distance}\ T\,com$$

- **Temps des échanges non transparents** :

$$T\,ech_{\text{nt}} = T\,\text{nt} \bullet T\,com \frac{Dp}{N\,mes}\ \text{distance}$$

- **Taux de communication de l'algorithme** :

$$C = N\,\text{tc} \bullet \frac{T\,com}{T\,seq}$$

Tableau 1. Principaux paramètres de la méthode.

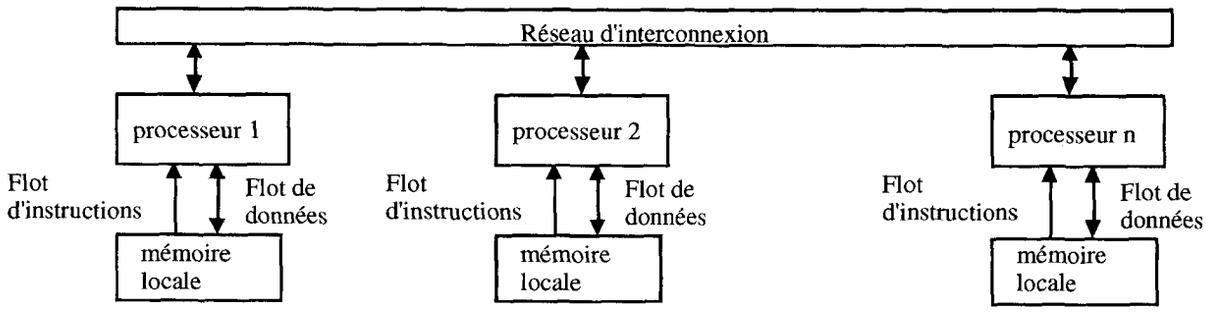


Figure 8. – Première architecture cible, multiprocesseur à passage de messages.

calculés ou résultent d'une analyse de l'exécution séquentielle de l'algorithme sur le processeur cible.

Notons que le calcul du seuil de saturation, permet de déterminer si le temps d'exécution parallèle d'un algorithme dépend du temps de traitement (les communications sont alors masquées, le réseau est hors saturation), ou du temps nécessaire à écouler les communications (les calculs sont masqués, le réseau est en saturation). La contrainte de coût conduira à rechercher le nombre de processeurs le plus faible associé au réseau le moins coûteux. Cette méthode a été intégrée dans un outil, ESPION [16]. Deux types de processeurs (transputer et DSP de la famille Texas) ont ainsi été expertisés avec une attention toute particulière sur l'interaction traitement–communication.

4.2. résultat de l'analyse

Les calculs de la FFT sont modélisés par un graphe de fluence [17], sur lequel les nœuds représentent les opérations élémentaires (papillons), et les arêtes les dépendances de données. Ainsi un nœud ne sera exécuté que lorsque ses opérandes seront disponibles. Le graphe représenté sur la figure 7, peut être découpé selon l'axe des temps en groupe d'actions que l'on appellera passes. Le premier modèle d'architecture analysé, est un multiprocesseur à passage de messages, dans lequel tous les processeurs sont identiques (figure 8). La topologie retenue sera celle de l'anneau, qui présente un coût faible, et peut être réalisée directement par de nombreux processeurs.

Ainsi sur un réseau de N processeurs, pour une FFT sur I échantillons, chaque processeur élémentaire (PE) se voit charger du calcul de $I/2N$ papillons sur une passe et doit donc posséder en mémoire les I/N échantillons correspondants. Entre deux passes du graphe de calcul (figure 9), dès lors que les opérandes nécessaires au calcul d'un papillon ne se trouvent pas dans la mémoire du processeur chargé de le calculer, les échanges nécessaires seront réalisés à travers le réseau d'interconnexion. On prendra dans la suite de l'article $I = 1024$ points pour les tableaux de résultats.

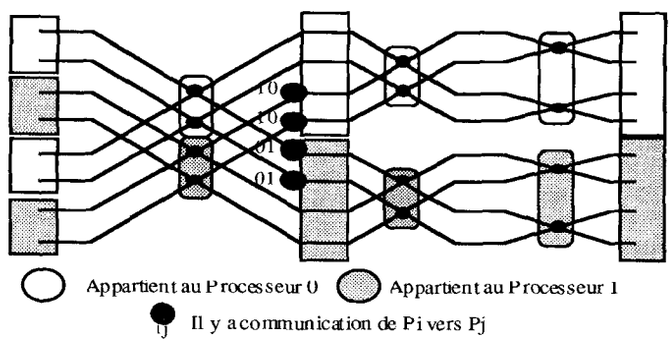


Figure 9. – Exemple d'implantation sur deux PE d'une FFT sur 8 points.

Une première analyse de ce graphe montre que la FFT parallèle se décompose en deux parties :

- Les $\log_2 N$ premières passes de l'algorithme nécessitent des échanges entre PE.
- Les $\log_2(I/N)$ autres sont sans échanges, l'efficacité de la parallélisation vaut ici 100%.

Au cours des premières passes nécessitant des échanges, le nombre d'échantillons globalement échangés sur le réseau par les N PE reste constant et égal à $I/2$ ce qui donne une quantité de $I/2N$ échantillons à communiquer pour un processeur.

Si le temps pour communiquer une donnée était très inférieur au temps de calcul, voire masqué par ce calcul, l'algorithme de la Transformée de Fourier Rapide Parallélisée (TFRP) serait globalement d'une efficacité de 100%. Mais, cette condition n'étant jamais atteinte, le calcul du seuil de saturation est un paramètre fondamental. Le tableau 2 contient les valeurs caractéristiques de l'algorithme de la FFT permettant d'appliquer la méthode.

La saturation ne peut intervenir que lorsqu'il y a calcul et échanges en même temps, donc durant l'une des $\log_2 N$ premières passes. En effet, si sur une passe, le temps consacré à faire des échanges est supérieur à celui nécessaire au calcul des $I/2N$ papillons, l'algorithme est globalement saturé. En outre, l'hypothèse initiale selon laquelle la distance entre processeur communiquant était constante, n'est pas ici vérifiée. Nous allons donc modifier notre formulation afin d'y introduire la notion de distance exacte. Lors d'une passe, un processeur de l'anneau communique avec un seul

• **Nombre de données par processeur à communiquer sur une passe** :

$$Dp = \frac{I}{2N}$$

• **Nombre total de communications que requiert l'algorithme** :

$$N\,\mathrm{tc} = \frac{I}{2}\log_2 N$$

• **Taille des blocs à transférer sur les liens** : $N\,mes = 1$
On considère ici l'équivalent d'un nombre complexe de 64 bits.

• **Temps de calcul séquentiel sur un processeur** :

$$T\,sep = \frac{I}{2}\log_2 I\; T\,pap$$

(Tpap étant le temps de calcul d'un papillon ($1 \otimes$,$2\oplus$ complexes et la gestion d'adressage)).

Tableau 2. – Paramètres liés à l'algorithme de la FFT DIF radix 2 sur 1 points.

PE, situé à une distance donnée d_β, et au cours de l'algorithme, communiquera avec $\log_2 N$ processeurs. Cette distance varie au cours des passes successives de la TFRP. Ici la distance d_β vaut $N \cdot 2^{-\beta}$, où β est le numéro de la passe. C'est au cours de la première passe que cette distance est maximale et a pour valeur $N/2$. Pour calculer la saturation sur la première passe, l'hypothèse suivante sera faite : si le réseau est saturé pour celle-ci, globalement l'algorithme TFRP se trouvera dans un état de saturation (voir tableau 3).

Le temps de calcul sur une passe β est égal à : $Tcal_\beta = I \cdot Tpap/2N$, tandis que le temps des échanges est égal à : $Tech_\beta = Ntc \cdot d_\beta \cdot Tcom/nbbus$. Le seuil de saturation intervient donc lorsque $Tech_\beta$ est supérieur à $Tcal_\beta$, ce qui implique $N > 4 \cdot Tpap/Tcom$. Des tableaux 1 et 2 ainsi que des paramètres liés au Transputer ($Tcom$ = 6.8 mus, $Tpap$ = 16μs), il ressort que la limite de saturation sera atteinte, dans le cas d'un anneau de Transputer, lorsque le nombre de processeurs devient supérieur à 8.

L'efficacité de cette architecture peut être maintenant évaluée. Avant saturation, l'efficacité sera définie comme le rapport entre le temps de calcul idéal et le temps de calcul (tableau 1). Après saturation, elle sera égale au rapport entre le temps idéal et le temps nécessaire aux communications. En effectuant la somme sur les passes nécessitant des échanges, on en déduit le temps non transparent global de l'algorithme, puis un calcul des efficacités.

$$Tech_{nt,\beta} = Tnt \cdot Tcom \frac{I}{2N}\frac{N}{N^\beta} = Tnt \cdot Tcom \cdot 2^{-(\beta+1)}.$$

$$Tech_{nt} = \sum_{b=1}^{\log 2N} Tnt \cdot Tcom \frac{I}{2} 2^{-\beta} = Tnt \cdot Tcom \frac{I}{2}\frac{N-1}{N}$$

$$\mathrm{Eff}_{hors\,sat} = \frac{1}{1 + \dfrac{Tnt \cdot Tcom}{Tpap}\dfrac{N-1}{\log_2 I}}$$

Rappelons que la saturation apparaît lorsque $N > 8$. L'efficacité après saturation dépend du nombre de passes quise trouvent dans l'état de saturation. Pour une valeur de N donnée, le temps des échanges $Tech_\beta$ sur les passes saturées, et le temps de calcul $Tréel_\beta$ des passes non saturées peuvent être calculés. L'efficacité globale résultera d'une étude sur l'ensemble des passes en tenant compte, passe après passe, du maximum entre calcul et communication :

$$\mathrm{Eff}_{sat} = \frac{Ti}{\sum_{\beta=1}^{\log_2 I} MAX(Tech_\beta, Tcal_\beta + Tech_{nt,\beta})}$$

Les performances de l'implantation d'un tel graphe sont donc conditionnées par les performances à calculer mais aussi à communiquer du processeur cible. Contrairement à d'autres algorithmes que nous avons étudiés par ailleurs[1], pour lesquels le temps de calcul global était largement supérieur au temps passé à faire des échanges, la TFR 1-D possède une granularité d'échanges assez fine, ce qui implique que les communications sont un facteur d'étranglement lors de l'implantation de l'algorithme, provoquant rapidement la chute de l'efficacité (voir tableau 3).

N	2	4	8	16	32
$Tcal$ *idéal* (ms)	40.5	20.2	10.1	5.1	2.5
$Tech_{nt}$ (ms)	7.04	10.56	12.32	13.2	13.64
$Tréel$ (ms)	47.54	30.76	22.42	18.30	16.14
Efficacité de la TFRP	85.2%	65.7%	45%	27.9%	15.5%

Tableau 3. – Efficacité de la TFRP sur 1024 points sur un anneau de Transputer.

Afin d'obtenir une efficacité maximale, cette granularité sera modifiée dans les paragraphes suivants afin d'accroître l'adéquation entre l'algorithme et l'architecture. A cet effet, des transformations algorithmiques seront effectuées, modifiant le rapport entre le poids des communications et celui du calcul, comme :

• Lissage des communications au cours des passes (graphe à géométrie constante),

• Développement du calcul afin de réduire le poids des communications,

• Modification de la structure des communications afin de réduire leur coût. Nous procéderons également à des modifications architecturales (changement de réseau, de processeur).

[1] Filtre de Kalman, transformée de Fourier deux dimensions.

5. application des transformations lors de l'analyse

5.1. mise en œuvre des transformations

Dans les lignes suivantes, une mise en œuvre des transformations structurelles et algorithmiques pour la transformée de Fourier sera exposée. Les résultats pratiques obtenus après implémentation sur les cartes Transputer confirment les prévisions effectuées.

5.1.1. ajout de nœuds de calcul

Modifions structurellement le graphe de la transformée de Fourier (TFR DIF en base 2), en ajoutant des calculs redondants; le graphe présenté sur la figure 10 est obtenu. Dans les premières passes, chaque processeur traite, en plus des papillons qui lui sont affectés, les papillons dont le résultat lui est nécessaire pour le calcul des passes suivantes.

Pour le même modèle d'architecture ainsi que le même principe de partitionnement et placement que précédemment, le processeur 0 voit le nombre de calcul qu'il doit traiter, augmenter (4 au lieu de 2 sur la première passe), mais ceci s'accompagne d'une réduction des communications (dans ce cas disparition des communications).

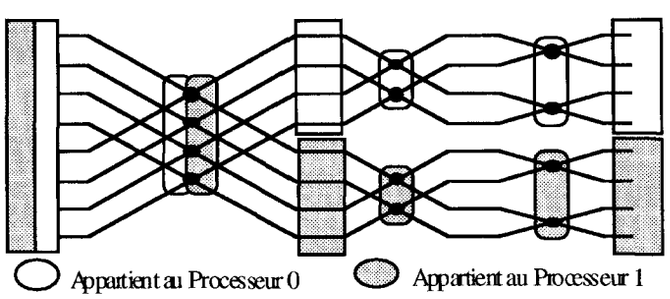


Figure 10. – Parallélisation de la TFR, rajout de nœuds de calcul.

Ce principe est généralisable au cas de N processeurs sur le calcul de la TFR d'un vecteur de I échantillons. Certaines données et certains calculs sont dupliqués sur plusieurs processeurs afin d'éviter tout échange entre eux. Précédemment, chaque processeur se voyait confier le traitement de $I/2N$ papillons, avec cette transformation il en traitera :

- Sur la 1ère passe, chaque PE calcule $\frac{I}{2}$ papillons,
- Sur la 2ème passe, chaque PE calcule $\frac{I}{4}$ papillons,
- Sur la βème passe, chaque PE calcule

$$\begin{cases} \frac{I}{2^\beta} & \text{papillons si} \quad 1 \leq \beta \leq \log_2 N \\ \frac{I}{2N} & \text{papillons si} \quad \log_2 N \leq \beta \leq \log_2 I \end{cases}$$

Soit un temps de calcul total sur un processeur :

$$Tcal = \left[\sum_{\beta=1}^{\log_2 N} \frac{I}{2^\beta} + \sum_{\log_2 N}^{\log_2 I} \frac{I}{2n} \right] Tpap$$

$$Tcal = \frac{I}{2N} \left[2(N-1) + \log_2 \left(\frac{I}{N} \right) \right]$$

L'efficacité est alors définie comme le rapport du temps idéal, et du temps obtenu par la mise en œuvre du calcul redondant. Le tableau 4 donne l'efficacité de cette nouvelle implantation.

$$\text{Efficacité}(I, N) = \frac{\log_2 I}{2(N-1) + \log_2 \frac{I}{N}}$$

N	2	4	8	16	32
$Tcal$ *idéal* (ms)	40,448	20,224	10,112	5,056	2,528
Texe (ms)	44,4928	28,3136	21,2352	18,2016	16,9376
Efficacité	90,91%	71,43%	47,62%	27,78%	14,93%

Tableau 4. – Efficacité de la TFR sur un anneau de transputer, performances prévues par l'outil Espion.

Ces performances sont très proches de celles obtenues sans utiliser la redondance (§ 4.2 et tableau 3), et ne sont donc pas satisfaisantes. Afin d'améliorer ceci, cette transformation sera appliquée sélectivement, c'est à dire qu'elle ne le sera pas sur l'ensemble des passes. Il est en effet possible d'examiner séparément sur chaque passe de la FFT les temps d'exécution obtenus, soit par les calculs et les échanges du graphe primitif (méthode du § 4.2), soit par les calculs redondants du graphe modifié. De plus, le placement des papillons sur les processeurs par rapport au premier algorithme a été changé, de manière à ce que la distance inter–PE augmente avec β, le numéro de l'étape. La distance est maintenant de 1 sur la première passe, puis vaut d_β $(2^{\beta-1})$ sur les passes suivantes d'indice β. Ceci ne change rien à l'étude précédente, la saturation apparaissant lors des dernières passes, et non plus lors des premières. Par contre ceci va avoir pour effet de faire varier en sens inverse la distance de communication, et la quantité de communications (figure 11). La démarche consiste alors à rechercher un seuil β_0 à partir duquel la duplication devient intéressante. En appelant $Tcal_{ECH}$ et $Tcal_{RED}$ respectivement les temps de la méthode initiale (§ 4.2), et celui obtenu par redondance, ces valeurs sont : si $\beta \leq \beta_0$, on a $Tcal_{ECH} < Tcal_{RED}$, et si $\beta > \beta_0$, on a $Tcal_{ECH} > Tcal_{RED}$.

Les résultats sont donnés dans le tableau 5. Dans la méthode consistant à réaliser des communications, le temps nécessaire aux échanges est égal à :

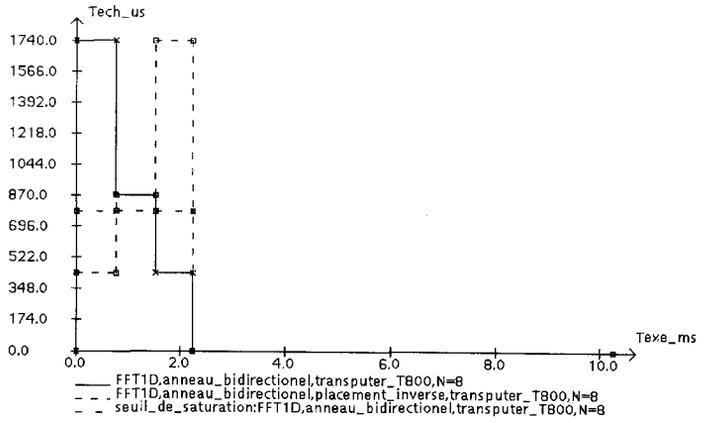


Figure 11. – Evolution de la distribution des échanges en fonction du temps.

$Tcal_{ECH} = \frac{I}{2N} Tpap + t\%Tcom \frac{Dp}{\text{Taille}}$ distance, et dans la méthode consistant à utiliser la redondance de calcul, celui–ci est égal à : $Tcal_{RED} = \frac{I}{2^\beta} Tpap$.

Une comparaison de ces résultats par rapport à ceux pour lesquels le recours à la duplication était systématique (tableau 4), fait apparaître une très nette amélioration dès lors que le nombre de processeurs devient important. Les performances deviennent similaires à celles obtenues avec le changement architectural consistant à utiliser un réseau hypercube plus coûteux, à la place de l'anneau (méthode du § 5.2).

N	2	4	8	16	32
$Tcal$ idéal (ms)	40,448	20,224	10,112	5,056	2,528
β_0 : seuil	0	1	1	2	3
$Tcal_{ECH}$ (ms) $\beta \leq \beta_0$	0	3,52	1,76	2,64	3,08
$Tcal_{RED}$ (ms) $\beta > \beta_0$	8,086	4,0448	6,0672	3,0336	1,5168
Tnt (ms)	8,0896	7,5648	7,8272	5,6736	4,5968
Passes suiv. $\beta > \log_2 N$	36,4032	16,1792	7,0784	3,0336	1,264
Texe (ms)	44,928	23,744	14,9056	8,7072	5,8608
Efficacité	90,91%	85,18%	67,84%	58,07%	43,13%

Tableau 5. – Performances globales prédites en mettant en œuvre sélectivement soit les communications soit les redondances de calcul.

5.1.2. modification du mécanisme des échanges

Jusqu'à présent une hypothèse simplificatrice consistant à dire que la dégradation en temps d'exécution apportée par la gestion des communications au niveau de l'architecture est nulle, a été utilisée. Ceci conduit à mettre en œuvre une communication dès que sur le graphe de la FFT apparaît un échange de données entre deux nœuds de calcul non présents sur le même processeur. Or il y a un prix à payer pour la gestion des communications, et qui dépend du coût lié à l'initialisation du DMA pour chaque échange ($Tinit$), et aux perturbations apportées par la gestion de l'échange ($Trout$).

Une technique bien connue des concepteurs, consiste à regrouper les données dans des messages, qui sont transmis une fois au lieu de faire autant de transmissions, et donc autant d'initialisations du DMA, qu'il y a de données dans le message. Le temps des échanges non transparents $Tnt \cdot Tcom$ est donc modélisé par : $Tinit + Trout \cdot Nmes$. Au niveau algorithmique cela consiste à faire la transformation présentée dans le tableau 6. Le temps de calcul global va donc être modifié car le nombre total d'initialisation des variables de communications va diminuer et donc également la dégradation apportée par les communications. Mais en contre partie, le transfert du dernier bloc d'échantillons, qui n'est plus masqué par du calcul, va augmenter Texe. Il existe donc un compromis dépendant de la variable $Nmes$ pour lequel Texe est minimum. Cet optimum est calculé en dérivant l'expression de $Tréel$ de l'algorithme présenté dans le tableau 6, par rapport à $Nmes$. Dans le cas du Transputer ($Tinit = 1.8\mu s$, $Trout = 25.7\mu s$), ces valeurs sont :

$$Tréel = Nmes \cdot Tpap + \left(\frac{I}{2NNmes} - 1\right) \cdot (Nmes \cdot Tpap + Tnt \cdot Tcom) + (Tnt \cdot Tcom + Tcom)$$

$$Nmes_{opt} = \sqrt{\frac{26.2\,I}{2N6.8}} = 32$$

Le tableau 7 s'en déduit, représentant les efficacités prévisionnelles pour $I = 1024$.

Ces résultats tendent à montrer que la transformation « échange modifié », fournit les meilleurs résultats pour une architecture à base de Transputer. La courbe de la figure 12 effectue une première comparaison des quatre implantations étudiées.

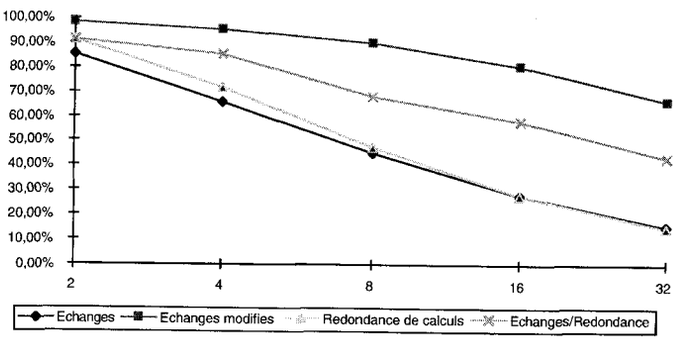


Figure 12. – Comparaison des méthodes des § 4.2, § 5.1.

5.1.3. sélection d'autres algorithmes

Dans le cadre des transformations algorithmiques, nous étudierons ici deux autres algorithmes. Nous prendrons deux exemples

Algorithme initial	Algorithme après transformation
FAIRE EN SEQUENTIEL SUR PEi ... Calcul du premier Papillon Pour les $\frac{I}{2N} - 1$ autres papillons de la passe FAIRE EN PARALLELE ... Echanges du résultat du papillon précédent avec PEj (Temps apparent : Tnt) ... Calcul du prochain Papillon (Temps : Tpap) FAIT ... Échanges du résultat du dernier papillon avec PEj Fin d'une passe	FAIRE EN SEQUENTIEL SUR PEi ... Calcul des Nmes premiers Papillons Pour les $\frac{I}{2NN\text{mes}} - 1$ autres papillons de la passe FAIRE EN PARALLELE ... Échanges des Nmes résultats des papillons précédents avec PEj ... Calcul des Nmes prochains Papillons FAIT ... Échanges des Nmes derniers résultats avec PEj Fin d'une passe
$T\text{réel} = T\text{pap} + \left(\frac{I}{2N} - 1\right).(T\text{ pap} + T\text{nt}.T\text{com}) + [T\text{nt}.T\text{com} + T\text{com}]$ Remarque : le terme entre crochets est négligeable, en effet $I/2N >> 1$.	$T\text{réel} = N\text{mes}.T\text{pap} + \left(\frac{I}{2NN\text{mes}} - 1\right).(N\text{mes}.T\text{pap} + T\text{nt}.T\text{com}) + (T\text{nt}.T\text{com} + T\text{com})$

Tableau 6. – Adaptation du mécanisme d'échanges.

N	2	4	8	16	32
taille	32	22	16	11	8
$Tcal$ *idéal* (ms)	40.5	20.2	10.1	5.1	2.5
Tnt (ms)	0,88	1.06	0,98	0.83	0.67
Texe (ms)	41,3	21,3	11,09	5,88	3.2
Efficacité	97.9%	95%	91.2%	85.9%	79%

Tableau 7. – Efficacité après transformation du mécanisme d'échanges.

: un changement de base (FFT DIF radix 4), et un changement de graphe de fluence (géométrie constante).

• Graphe à géométrie constante

Le graphe à géométrie constante possède une structure qui reste régulière au cours des passes de l'algorithme [18]. Pour une implantation sur un multiprocesseur, ceci se traduit d'une part, par une diminution de la distance moyenne entre les PE (par rapport à la TFR DIF), et d'autre part, par une augmentation du nombre total d'échanges nécessaires. Pour ce graphe le nombre de communications (Ntc) est égal sur chaque passe, et ceci tout au long des $\log_2 I$ passes de l'algorithme.

$$Ntc = \frac{N-1}{N} I \cdot \log_2(I)$$

De plus un placement bien étudié des papillons sur les processeurs du réseau doit permettre d'obtenir une distance unitaire puisque les communications sont régulières au cours des passes. La figure 13 illustre ce phénomène.

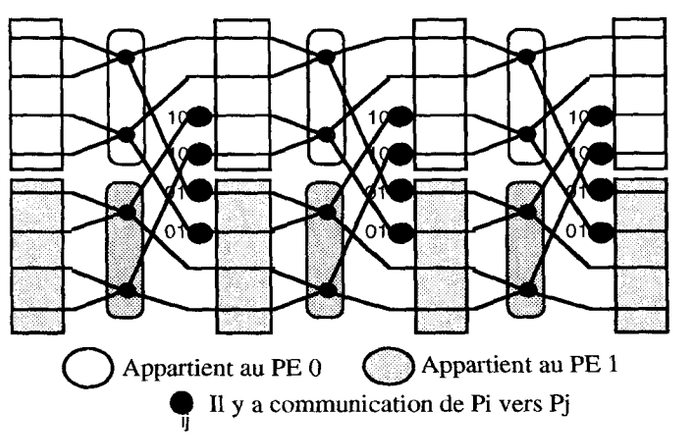


Figure 13. – Partage du graphe à géométrie constante sur les différents processeurs.

• Graphe de la TFR en base 4. Extension à une base B

Le graphe de la TFR en base 4 nécessite le calcul de $N/4 \cdot \log_4 N$ papillons, chaque papillon nécessitant 3 multiplications complexes et 8 additions/soustractions complexes. Ce nouveau graphe permet donc une diminution de la complexité en calcul d'une transformée de Fourier rapide.

Le graphe en base 4 se prête bien à une parallélisation. En effet le nombre total d'échanges décroît par rapport à la TFR base 2. Les échantillons sont regroupés par quatre lors du calcul d'un papillon. Un processeur devra donc, au pire, communiquer trois

échantillons pour un calcul de papillon. En outre, le nombre de passes nécessitant des échanges décroît de $\log_2(N)$ vers $\log_4(N)$ dans le cas d'un nombre de processeurs N égal à une puissance de quatre.
Le calcul du nombre de communications pour N valant une puissance de deux peut également être généralisé.

$$\text{si } N = 2^N : Ntc = \frac{I}{2} + \frac{3I}{4}\log_4\left(\frac{N}{2}\right) = \frac{3I}{4}\log_4(N) + \frac{7I}{8}$$
$$\text{si } N = 4^n : Ntc = \frac{3I}{4}\log_4(N)$$

L'évaluation de $Nsat$ pour le graphe en base quatre donne :

$$Tcal = \frac{I}{4N}\log_4(N) \cdot Tpap_{\text{base 4}}$$

et

$$Tech = Ntc\frac{dmoy}{nbbus}Tcom = \frac{3I}{4}\log_4(N)\left(\frac{N}{N-1}\right)$$

La saturation est obtenue pour :

$$N > \frac{8}{3}\frac{Tpap_{\text{base 4}}}{Tcom}\#25$$

• Comparaison avec la TFR DIF BASE 2

La comparaison du nombre de communications global de l'algorithme de la TFR, en fonction de son graphe de calcul, est donnée par le tableau 8, sur un exemple où $I = 1024$ échantillons :

Valeur de N	2	4	8	16	32
Ntc DIF radix 2	512	1024	1536	2048	2560
Ntc géométrie constante	10240	15360	17920	19200	19840
Ntc DIF radix 4	512	768	1280	1536	2048

Tableau 8. – Comparaison du nombre de communications pour divers algorithmes.

Les résultats montrent que le nombre de communications est minimum pour la TFR radix 4, et que de plus, le papillon élémentaire étant plus complexe, cela a pour effet d'augmenter la charge de calcul en parallèle avec les échanges, et donc d'éloigner le phénomène de saturation.

La TFR à géométrie constante quant à elle, souffre du fort volume de données à transférer. Les comparaisons de performances de ces algorithmes pour un anneau de transputer ont été prédites et sont résumées dans les courbes de la figure 14.

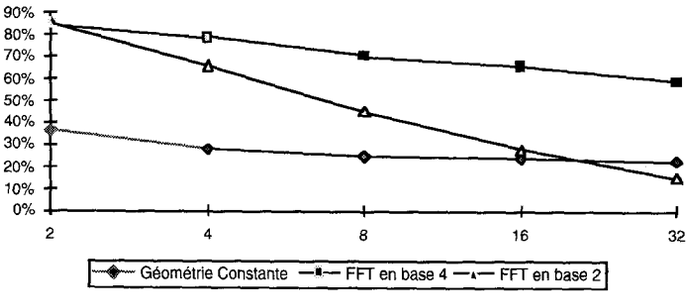


Figure 14. – Efficacité comparée des trois algorithmes.

5.2. autres transformations : changement de réseau

Pour cette expérience concernant l'algorithme TFR radix 2 DIF, l'architecture cible appartient au modèle MIMD à mémoire distribuée : un hypercube de transputer. Un hypercube est caractérisé par la présence de $N = 2^n$ processeurs interconnectés selon un cube de dimension n; chaque sommet est relié à $n = \log_2 N$ autres sommets (figure 15). Ces conditions vont permettre aux échanges de s'effectuer entre deux voisins tout au long de l'algorithme. En effet dans la TFRP en base 2, chaque PE communique avec $\log_2 N$ autres PE.

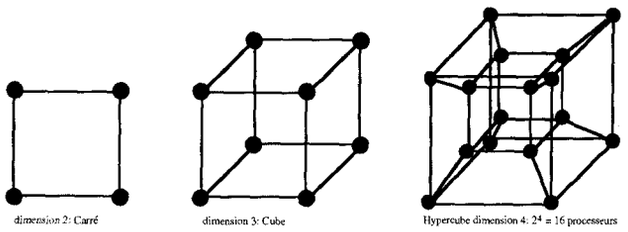


Figure 15. – Réseaux hypercube pour des dimensions de 2, 3 et 4.

La mise en œuvre des paramètres présentés dans les tableaux 1 et 2, montre que le seuil de saturation est atteint à condition que, C soit supérieur à 2, c'est à dire $Tcom > 2Tpap$. Cette condition n'est jamais atteinte dans le cas du Transputer. L'efficacité, dans le cas non optimal pour lequel les données sont transférées échantillon par échantillon, a pour expression :

$$\text{Eff} = \frac{Ti}{Tréel}$$
$$\text{avec } Tréel = Tcal + Tnt \cdot Tcom\frac{Dp}{Nmes}\text{ distance}$$
$$= Tcal + Tnt \cdot Tcom\frac{I\log_2 N}{2N}$$
$$\text{Eff} = \frac{1}{1 + Tnt \cdot Tcom\frac{\log_2 N}{2Tpap\,\log_2 I}}$$

L'efficacité a donc largement augmenté par rapport à l'anneau. Le tableau 9 présente les résultats de prévision de performances pour

N	2	4	8	16	32
$Tcal$ *idéal* (ms)	40.5	20.2	10.1	5.1	2.5
Tnt (ms)	7.04	7.04	5.28	3.52	2.2
Texe (ms)	47.54	27.24	15.38	8.62	4.7
Efficacité	85.2%	74.2%	65.7%	59.2%	53.2%

Tableau 9. – Efficacité de la TFR radix 2 DIF sur un hypercube de Transputer.

un hypercube. Par rapport à l'utilisation de l'anneau, l'hypercube permet de maintenir de bonnes performances, même pour un nombre de processeurs élevés.

D'autres réseaux peuvent être étudiés, permettant ainsi de trouver l'architecture offrant la meilleure efficacité au coût le moins élevé (figure 16).

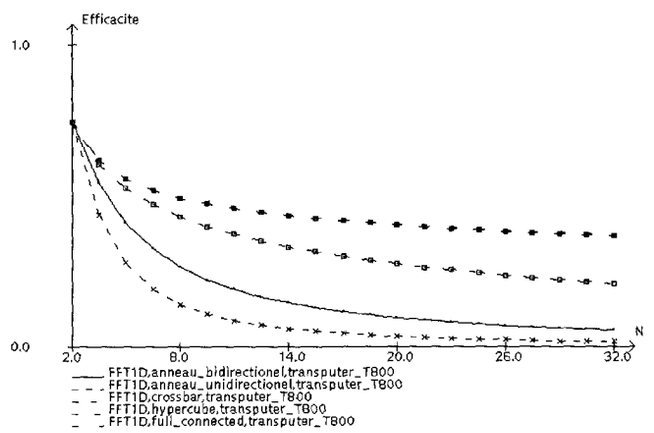


Figure 16. – Étude prévisionnelle de performances pour différentes topologies de réseaux à base de Tranputer.

5.3. bilan des transformations

La pertinence des transformations, dépend étroitement du processeur cible. Ce qui a été présenté pour le transputer, est bien évidemment transposable à d'autres processeurs, par exemple les processeurs de traitement du signal comme la famille Texas TMS32O. La figure 17 montre quelques résultats relatifs à cette famille de processeurs.

Notre proposition est donc, à travers une modélisation du problème et une identification des transformations possibles, d'explorer avec méthode l'espace de conception. Ceci suppose la mise en place d'un outil fortement interactif.

L'outil développé, ESPION, comporte une bibliothèque de composants (Transputer T800, TMS320C25, C30, C40, processeur hétérogène), de réseaux. Il permet d'explorer l'espace de conception avec comme critères : l'efficacité, l'accélération, le coût et le temps. Il est basé sur une estimation précise du surcoût des communications par des paramètres mesurés sur les processeurs, puis sur une exécution du code séquentiel sur la station de travail afin de mesurer les communications engendrées par un partage des données entre les différents processeurs. Cet outil permet de :

- Prédire, pour un algorithme, un nombre de processeurs et un partage donné, le comportement des échanges au cours du temps (figure 11) par rapport au seuil de saturation,
- Prédire, pour un algorithme, l'évolution de l'efficacité ou de l'accélération réelle de son implantation en fonction du nombre de processeurs, en tenant compte du phénomène de saturation. Ces estimations peuvent être itérées pour des réseaux et des processeurs appartenant à la bibliothèque (figures 16 et 17),
- Obtenir des propositions d'architectures permettant d'atteindre un temps d'exécution fixé par le développeur.

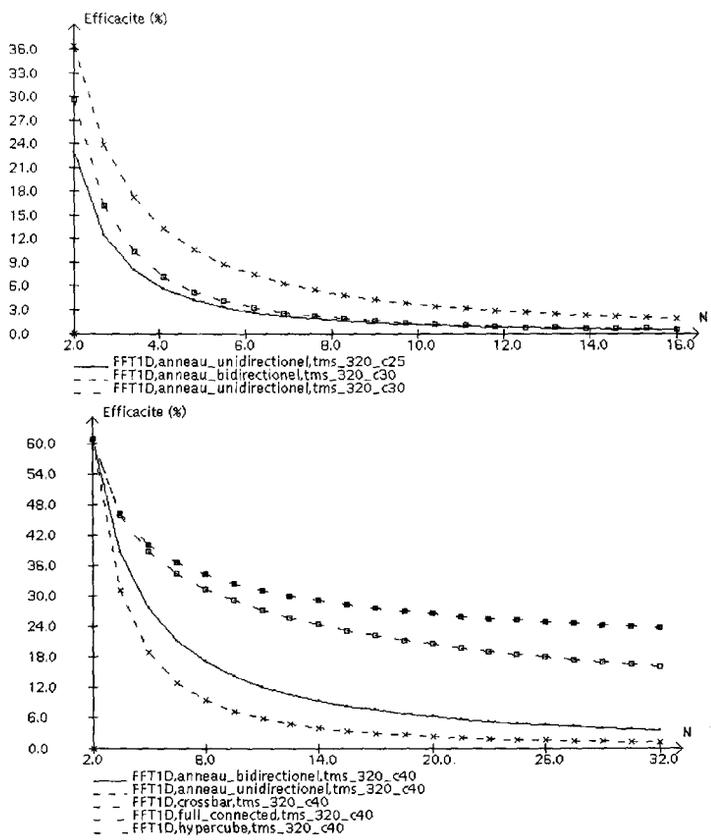


Figure 17. – Étude prévisionnelle de performances pour différentes topologies de réseaux à base de TMS320.

6. migrations de fonctions vers le matériel

Lorsque l'étude par analyse permet de penser que les diverses contraintes liées à l'application ne pourront pas être atteintes par une solution uniquement programmable, il convient de modifier le

partitionnement. Ainsi, certaines fonctionnalités, jugées critiques, migreront vers une solution synthétisée. Cette architecture dédiée aura à respecter une contrainte de temps locale (figure 3). Pour cette étude on supposera que la FFT, est à mettre en œuvre dans du matériel dédié, les processeurs programmables étant chargés du contrôle de l'ensemble.

6.1. présentation de la synthèse architecturale

La synthèse architecturale [19], permet de passer du domaine comportemental au domaine structurel, et, dans le domaine comportemental, de passer du niveau architecture au niveau RTL. Ceci est effectué à partir d'un point d'entrée composé de la spécification comportementale exprimée, généralement, sous forme d'un algorithme, d'un énoncé de contraintes, et d'une librairie contenant les composants (X, $+$, UAL, registre, ...), à partir de laquelle l'architecture sera conçue. Les différentes approches de synthèse architecturale se caractérisent par le domaine d'applications supportées (traitement vs contrôle), la gamme de contraintes, l'architecture cible (bit série, systolique, processeur, ...), ainsi que par la formalisation de la méthode de conception. Ainsi, nos travaux portent sur les architectures DSP, sous contraintes de temps, coût et consommation, pour des applications de traitement du signal et de l'image. La méthode distingue les 5 étapes traditionnelles de la synthèse architecturale : Compilation, Sélection du jeu de composants, Allocation du nombre d'opérateurs, Ordonnancement et Assignation des opérations sur les opérateurs, Optimisation.

Afin de supporter la notion de temps réel, l'unité de traitement sera conçue en premier. La méthode fournit, dans un premier temps, la description structurelle de l'unité de traitement, la description comportementale de l'unité de commande, ainsi que les spécifications, sous la forme d'un graphe de transfert, qu'aura à satisfaire l'unité de mémorisation qui est donc synthétisée ultérieurement [20].

Le flot de conception a été mis en œuvre dans l'outil GAUT [21], dont l'organisation est représentée sur la figure 18.

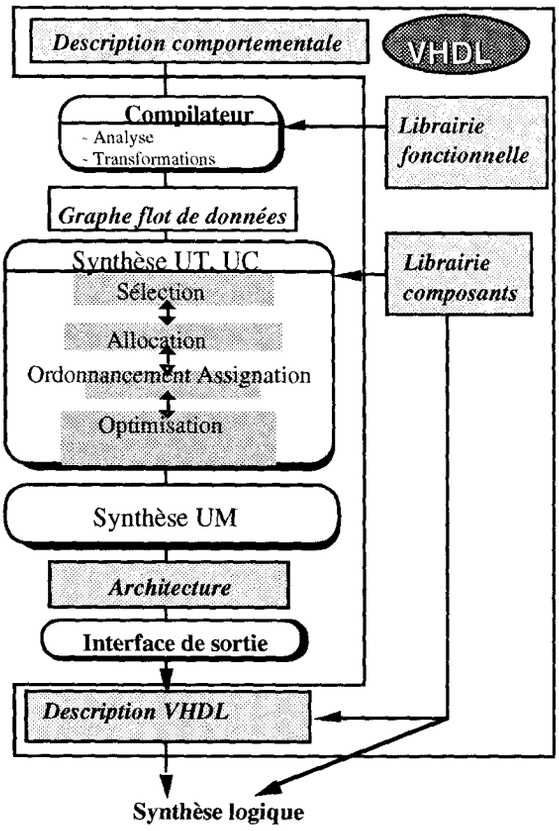


Figure 18. – Le flot de synthèse de GAUT.

6.2. application à la FFT

L'algorithme, traduit en VHDL, est communiqué à l'outil de synthèse d'architecturale, GAUT. Durant la première étape de la synthèse, l'algorithme est transformé en une modélisation flot de données, pour laquelle l'unité de calcul est ensuite synthétisée (tableau 10).

Mise en œuvre d'une transformation algorithmique

Comme il a été exposé dans la première partie de cet article, la TFR existe sous plusieurs formes algorithmiques qui peuvent être concurrentes à l'intégration dans un ASIC. Le tableau 11 représente les résultats de la synthèse pour quatre algorithmes candidats : La DIT radix 2, la DIF radix 2, la DIF radix 4 et la géométrie constante. Il donne la surface totale de l'unité de traitement du circuit en mm^2, la surface de l'unité de mémorisation en mm^2, ainsi que le rapport entre la surface de l'unité de mémorisation et la surface totale.

L'analyse des résultats montre que le choix d'une version algorithmique demande de pouvoir disposer des estimateurs les plus complets possibles. Par exemple la FFT radix 4 présente un coût inférieur en terme de surface allouée à l'unité de traitement, mais qui est partiellement compensé par l'accroissement du coût de l'unité de mémorisation.

Contrainte temps réel : 4 ms	**Contrainte temps réel : 0,2 ms**	**Contrainte temps réel : 0,1 ms**
3,51 mm^2 1 multiplieur (6%), 1 add/sous, 8 registres, 5 multiplexeurs	**5,3 mm^2** 2 multiplieurs (64%), 1 add/sous, 9 registres, 5 multiplexeurs	**9,5 mm^2** 3 multiplieurs (80%), 1 additionneur, 1 soustracteur, 18 registres, 14 multiplexeurs

Tableau 10. – Évaluation de la surface de l'unité de traitement de l'architecture.

	surface UT	surface UM	Rapport $\frac{S_{UM}}{S_{totale}}$
DIT rad. 2	16,18	13,40	45
DIF rad. 2	16,31	12,30	43
DIF rad. 4	11,79	16,79	59
geom const	17,45	10,99	39

Tableau 11. – Réultats de la synthèse architecturale des différents algorithmes (FFT 256 échantillons, 16 Khz).

7. conclusion

Cet article a montré l'interaction forte qui existe entre algorithme et architecture. Pour un type d'architecture sélectionné, tous les algorithmes répondant à l'application n'ont pas les mêmes performances. Quelques expériences de transformations permettant de renforcer l'adéquation de l'algorithme à l'architecture ont été présentées. Certaines d'entre elles sont réalisées en faisant migrer des fonctionnalités vers du matériel spécifique. Ceci conduit à une explosion du domaine des solutions architecturales à explorer. En effet, celui–ci dépend de l'application qui apporte ses contraintes (temps, coût consommation, ...), de l'algorithme sélectionné éventuellement en fonction de critères de qualité, ..., et du modèle d'architecture retenu. Une approche méthodologique et outillée, est donc nécessaire. Un essai de définition du cahier des charges d'un outil ayant pour rôle de guider le concepteur, conduit aux quelques remarques suivantes.

Une approche universelle permettant de prendre en compte toutes les applications (traitement vs contrôle), ainsi que les contraintes liées (traitement vs mémorisation, performances vs consommation), et tous les modèles architecturaux (programmable vs dédié, architecture régulière vs processeur de traitement du signal), est irréaliste. Il faut donc commencer par définir ces deux points afin de circonscrire la méthode, et/ou disposer d'estimateurs permettant de tester différentes possibilités de conception au niveau le plus haut. Le domaine de solutions ainsi délimité doit pouvoir être exploré par le concepteur, qui doit pouvoir guider l'approche : une approche totalement automatique n'est pas à ce jour envisageable, ni même souhaitable. Cela repose donc le problème des estimateurs qu'il faut, de l'endroit ou il faut les calculer, du rapport acceptable entre la pertinence de l'estimateur et le temps de calcul qu'il nécessite, et de leur indépendance vis-à-vis de l'architecture cible. Enfin une démarche totalement descendante n'est pas envisageable, il faut donc que certains des estimateurs soient suffisamment pertinents pour guider le concepteur dans ses transformations.

La poursuite des travaux est actuellement menée dans différentes directions : extension du domaine d'application en élargissant le modèle architectural supporté ainsi que les contraintes, et surtout formalisation d'estimateurs et des transformations pouvant être utilisés aux différents niveaux du cycle de conception.

BIBLIOGRAPHIE

[1] D. Gajsky & R. Kuhn, « New VLSI Tools », *Guest Editors' Introduction, IEEE Computer*, Vol. 6, n°12, Dec 1983, pp. 11–14.

[2] P. Mitchel, U. Lauther, P. Duzy, *The synthesis approach to digital design*, Kluwr Academics 1992.

[3] M. Potkonjak and J. Rabaey, « Optimizing Ressource Utilization Using Transformations », *IEEE transactions on C.A.D. of I.C.*, vol. 13, n°3, Mar. 94.

[4] P. Marwedel, G. Goosens, *Code generation for embedded processors*, Kluwer Academic Publishers, 1995.

[5] E. Martin, J.L. Philippe, H. Dubois « GAUT : An architectural synthesis tool for dedicated signal processor », *EURODAC 93*, 20–24 septembre 1993, Hambourg.

[6] J.Ph. Diguet, O. Sentieys, J.L. Philippe, E. Martin, « Probabilistic Resource Estimation for Pipeline Architectures under Latency Constraint », *IEEE Workshop on VLSI Signal Processing*, Japon.

[7] J. Rabaey, M. Pedram Schafer, *Low power design methodologies*, Kluwer Academic Publishers 1995.

[8] C.C. Leiserson, F.M. Rose, J.B. Saxe, « Optimizing synchronous system », *22^{n}d Ann. Symp. on Foundations of computer science*, 1981, pp. 23–36.

[9] D.A. Lobo, B.M. Pangrle, « Generating pipelined datapaths using reduction techniques to shorten critical paths », *EDAC 92*, pp. 390–395.

[10] K.K Pahri, D.G. Messerschmitt, « Pipeline interleaving and parallelism in recursive digital filters–Part 1 : Pipelining using scattered look–ahead and decomposition », *IEEE trans. on CAD*, vol. 37, N°7, july 89, pp. 1099–1117.

[11] K.K Pahri, D.G. Messerschmitt, « Pipeline interleaving and parallelism in recursive digital filters–Part 2 : Pipelined incremental block filtering », *IEEE trans. on CAD*, vol. 37, N°7, july 89, pp. 1118–1134.

[12] Numero spécial, « Algorithmes adaptatifs et soustraction de bruit », *Revue Traitement du Signal*, vol. 6, n°5, 1989.

[13] A. Gilloire, « Adéquation Algorithmes Architectures : Apllications en annulation d'écho acoustique », *Conférence AAA*, Lannion, 14–15 sept. 1992.

[14] J.W. Cooley & J.W. Tuckey, « An Algorithm for the machine calculation of Complex Fourier Series », *Mathematics of Computation*, Vol. 15, N°9, April 1965, pp. 297–301.

[15] O. Sentieys, E. Martin, H. Dubois, J.L. Philippe & M. Corazza, « Application de l'outil ESPION pour l'analyse des architectures multiprocesseurs au filtrage de Kalman 2–D rapide », *Revue Traitement du Signal*, Vol. 10, N°1, 1er trimestre 1993.

[16] O. Sentieys, *Analyse et synthèse d'architectures en TDSI : vers la conception d'architectures hétérogènes*, Thèse Université de Rennes 1993.

[17] A.V. Oppenheim & R.W. Schafer, *Digital Signal Processing*, Prentice–Hall, 1975.

[18] M. Kunt, « Traitement numérique des signaux », *Editions presses polytechniques romandes*, Dunod. 1981.

[19] H. De Man, « Silicon Compilation for real Time Signal Processing systems Tutorial on high Level Synthesis », *EDAC*, Glasgow 12–15 mars 1991.

[20] J.L. Philippe, D. Chillet, O. Sentieys, « Memory aspect in signal processing and HLS tool : some results », *EUSIPCO 1996*.

[21] J.L. Philippe, O. Sentieys, J.P. Diguet, E. Martin, « Synthesis : from digital signal processing to layout », *Logic and architecture synthesis state of the art and novel approaches Chapman and Hall*, 1995, pp. 307–313.

Manuscrit reçu le 5 Janvier 1996.

LES AUTEURS

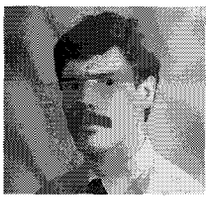
Olivier SENTIEYS

Ingénieur diplômé de l'ENSSAT en 1990, Olivier Sentieys passe une thèse de l'Université de Rennes en 1993 sur la conception des architectures hétérogènes. Maître de conférences à l'ENSSAT depuis 1993, il travaille sur la synthèse de haut niveau des circuits VLSI dans le cadre des applications de traitement du signal et de télécommunications. Il est impliqué dans le développement de l'outil de synthèse architecturale GAUT et dans ses évolutions pour les nouvelles contraintes technologiques et applicatives.

Hélène DUBOIS

A reçu le diplôme d'ingénieur ENSERB en 1985 puis a soutenu une thèse de l'Université de Rennes 1 en janvier 1991 sur l'analyse des systèmes multi-processeurs. Elle est actuellement maître de conférence à l'ENSSAT. Ses travaux de recherche portent sur l'étude des performances des architectures multi-processeurs pour le domaine du traitement du signal.

Eric MARTIN



Agrégation de génie électrique à l'ENS de Cachan en 1984, Maître de conférences à l'ENSSAT, puis Professeur à l'UBS depuis 1994, où il dirige le LESTER. Son domaine d'intérêt porte sur les outils de CAO en architectures dédiées aux applications de traitement du signal et de l'image.

Jean-Luc PHILIPPE

Il a passé sa thèse en 1984, à l'université de Rennes 1. Maître de conférence à l'ENSSAT, ses domaines d'intérêt portent sur l'interaction algorithme — architectures dans le domaine du traitement numérique du signal. Depuis Septembre 1996 il est Professeur à l'Université de Bretagne Sud (UBS).