# Conception d'un composant matériel réutilisable flexible pour la transformation en ondelettes 2D

## Design of a flexible, re-usable hardware component for the 2D discrete wavelet transform

**G. Savaton[1], E. Casseau[2], E. Martin[2], C. Lambert-Nebout[3]**

[1] ESEO, 4 rue Merlet de la Boulaye, BP 30926 – 49009 Angers cedex 01, E-mail : guillaume.savaton@eseo.fr

[2] LESTER, CNRS FRE 2734, Centre de Recherche UBS, Rue de Saint-Maudé, BP 92116 – 56100 Lorient cedex, E-mail : prenom.nom@univ-ubs.fr

[3] CNES, DTS/AE/SEA/ET, 18 avenue Edouard Belin – 31401 Toulouse cedex 4, E-mail : catherine.lambert@cnes.fr

Manuscrit reçu le 10 mars 2003

Résumé et mots clés

Dans cet article, nous nous intéressons à l'implantation matérielle de la transformation en ondelettes discrète 2D sous forme d'un composant réutilisable flexible. Ce composant, compatible avec le standard *JPEG2000*, est destiné à être intégré dans une variété d'applications embarquées de compression d'images.
Une méthodologie de conception originale reposant sur les nouveaux outils de synthèse de haut niveau nous a permis d'atteindre un degré élevé de flexibilité dans la spécification et la synthèse de ce composant. Celle-ci autorise en effet la personnalisation de paramètres fonctionnels (choix du banc de filtres lifting, nombre de niveaux de décomposition), de contraintes de communication (ordre de parcours des pixels de l'image, date de lecture/écriture des données) et de contraintes de performances (vitesse de traitement, parallélisme de calcul) qui facilitent ainsi sa réutilisation dans différentes applications et environnements d'intégration.
Dans cet article, nous dressons tout d'abord un état de l'art des nouvelles approches en conception de systèmes intégrés. Nous rappelons brièvement les bases théoriques de la transformation en ondelettes discrète et nous présentons les approches classiques pour son implantation sous forme d'architectures *VLSI*. Après avoir présenté les principes de notre méthodologie de conception, nous déclinons ses étapes successives, de l'algorithme aux architectures, dans le cas de la transformation en ondelettes 2D utilisant le *Lifting Scheme*. Nous concluons par des résultats de synthèse démontrant l'efficacité de la démarche suivie en termes de flexibilité de la spécification obtenue.

Transformation en ondelettes discrète, standard JPEG2000, adéquation algorithme-architecture, composants IP (Intellectual Property), synthèse de haut niveau.

Abstract and key words

This paper deals with the implementation of the 2D discrete wavelet transform in the form of a reusable, flexible hardware component. This component is compliant with the *JPEG2000* standard and targets a variety of embedded imaging applications.
A novel design methodology based on the emerging high-level synthesis tools allowed us to achieve a high degree of flexibility in the specification and synthesis of this component. Customization of functional parameters is supported (choice of the lifting-based filter bank, number of decomposition levels) as well as communication constraints (pixel

ordering and I/O scheduling) and performance constraints (computation speed and parallelism), facilitating reuse in various applications and integration environments.
In this paper, we first provide a summary of the recent trends in embedded system design. Then the theoretical bases of the discrete wavelet transform and the classical approaches for implementing it in hardware are briefly presented. After presenting the principles of our design methodology, we detail the successive design stages, from the algorithm to the architectures, in the case of the 2D lifting-based discrete wavelet transform. We conclude with synthesis results demonstrating the effectiveness of our approach for designing highly flexible hardware components.

Discrete wavelet transform, JPEG2000 standard, algorithm-to-architecture matching, Intellectual Property (IP) cores, high-level-synthesis.

# 1. Introduction

Les applications d'imagerie numérique manipulent des quantités de données croissantes. Leur stockage et leur transmission à travers des résaux de communication requièrent des formats de codage de l'information de plus en plus performants et flexibles, assurant de forts taux de compression tout en conservant une bonne qualité de restitution. Afin de répondre à ces nouvelles exigences, le standard *JPEG2000* propose un ensemble de techniques de codage dont les objectifs sont de satisfaire les exigences d'une variété d'applications [20]. La prise en compte de ces exigences, regroupées en un ensemble de *profils* applicatifs [20], font de *JPEG2000* un standard très flexible, dont la chaîne de compression peut être personnalisée au moyen d'une variété de paramètres (nombre de niveaux de décomposition en ondelettes, choix de la base d'ondelettes, valeurs des pas de quantification, etc) [21, 1].
La nécessité d'atteindre de hautes performances en compression se traduit cependant par un choix de techniques de codage à forte complexité calculatoire qui, associée aux exigences de flexibilité, pose un défi à la réalisation d'un codeur ou décodeur *JPEG2000* « universel » réutilisable dans une variété d'applications d'imagerie embarquées : pour des fonctions complexes de traitement du signal et de l'image, l'exigence de flexibilité donne généralement la préférence à des solutions logicielles. Cependant, le parallélisme limité des processeurs peut ne pas satisfaire les contraintes de vitesse de traitement. Des solutions matérielles, au contraire, permettent d'atteindre des vitesses de calcul élevées, mais l'optimisation des performances se fait souvent au détriment de la flexibilité, si bien qu'une architecture conçue dans le cadre d'une application donnée peut ne pas satisfaire les contraintes d'une autre application, ce qui limite les possibilités de réutilisation d'un composant matériel complexe. Dans cet article, nous développons une approche originale pour la conception de composants matériels réutilisables visant des applications de traitement du signal et de l'image. Cette approche repose sur de nouveaux outils de synthèse de haut niveau qui réalisent l'analogue matériel de la compilation dans le domaine logiciel. Ces outils permettent ainsi de générer une variété d'architectures matérielles – satisfaisant une variété de besoins applicatifs et de contraintes d'implantation – à partir d'une spécification de niveau algorithmique du comportement souhaité.
Nous avons appliqué cette méthodologie à la spécification d'un composant réutilisable réalisant la transformation en ondelettes discrète de *JPEG2000*. Nous avons ainsi pu dégager une nette amélioration du flot de conception concernant d'une part le temps de développement et de vérification du composant, la lisibilité de la spécification synthétisable, et d'autre part son degré élevé de flexibilité qui autorise sa réutilisation dans une variété d'applications en personnalisant les paramètres fonctionnels supportés par le standard *JPEG2000* et le parallélisme de calcul.

#### Contexte de l'étude

Le travail présenté dans cet article a été mené en collaboration avec le *Centre National d'Études Spatiales* (*CNES*) et la société *Astrium*. Il s'inscrit dans le cadre d'une étude concernant les prochaines générations de satellites d'observation de la Terre. Les images produites par ces satellites se caractérisent en effet par leur largeur significative (12 000 pixels dans les applications actuelles, 24 000 pixels dans les projets à venir) et par un mode d'acquisition en continu des lignes de pixels qui nécessite le traitement des données *au fil de l'eau* [20, 28]. Les techniques de compression d'images jouent un rôle essentiel afin d'une part de minimiser la quantité de mémoire embarquée, et d'autre part de tirer le meilleur parti de la bande passante disponible pour la transmission des données vers le sol. Les charges utiles des satellites *SPOT5*, *Phobos* et *Clementine* intègrent une chaîne de compression très proche du standard *JPEG* [30], mais dont les

algorithmes de quantification et de codage ont été adaptés aux caractéristiques spécifiques des images spatiales. L'implication du *CNES* dans le comité de standardisation *JPEG2000* [20, 21] pour le codage des images fixes, et la prise en compte depuis mars 1999 des exigences du profil *télédétection optique* dans le cahier des charges de ce nouveau standard [20], tendent vers une unification du format de codage des images à bord des véhicules spatiaux avec le format de diffusion de ces images.
L'objectif de l'étude que nous présentons dans cet article a consisté en la définition d'un composant générique et flexible permettant à la fois d'assurer les contraintes d'implantation (temps réel, quantité de mémoire embarquée, consommation) et de s'adapter rapidement aux paramètres sélectionnés pour une mission définie.

# 2. Nouvelles approches en conception de systèmes intégrés

En parallèle à l'évolution rapide de la complexité des applications – +90 % par an d'après l'International Technology Roadmap 2001 [22] –, la croissance de la capacité des technologies intégrées (loi de Moore [37]) va dans le sens d'une concentration des fonctions sur des surfaces de silicium de plus en plus réduites. Il est aujourd'hui permis d'envisager l'intégration d'un système complet sur un seul *ASIC* ou *FPGA*, donnant naissance à la notion de système sur puce, ou *System-on-a-Chip* (*SoC*) [22].
En revanche, les méthodes et outils de conception et de vérification couramment utilisés ne sont pas adaptés à gérer la complexité croissante des systèmes et de leur support physique. La capacité d'intégration effective des concepteurs ne croît en effet que de 32 % par an [22], de telle sorte que le fossé entre capacité d'intégration théorique et pratique ne cesse de se creuser. Les coûts non récurrents associés à la conception d'un nouvel *ASIC* ont été multipliés par dix au cours de la dernière décennie [22, 55] ; il est plus que jamais indispensable de fiabiliser le flot de conception de manière à fournir à la fabrication une spécification exempte d'erreur.

### 2.1 Accélérer le flot de conception : la réutilisation de composants virtuels

La démarche de rationalisation du flot de conception, vérification et synthèse engagée ces dernières années repose sur la constatation qu'un grand nombre de fonctionnalités sont communes à la plupart des applications et ont déjà été implantées de nombreuses fois en matériel ou en logiciel [45, 8]. Le flot de conception d'un système intégré peut ainsi être accéléré de manière significative en réutilisant des blocs matériels ou logiciels déjà conçus et vérifiés plutôt que de les re-concevoir. Un tel composant réutilisable est appelé « composant virtuel » [45], ou plus communément « bloc *IP* » pour *Intellectual Property*, la notion de *propriété intellectuelle* étant liée au fait que les concepteurs d'un tel composant sont *a priori* distincts de l'équipe chargée de l'intégration. La notion d'*IP* consiste à appliquer au matériel les principes de conception que le génie logiciel met en œuvre avec succès depuis des années sous forme de *composants logiciels* réutilisables.
La figure 1 présente dans sa partie gauche un flot typique, volontairement simplifié, de conception d'un système complet comprenant des parties matérielles et logicielles. Les étapes de conception détaillée et de synthèse du matériel font figurer explicitement les opérations de sélection, d'instanciation et de synthèse de composants réutilisables. La figure illustre également la tendance actuelle qui consiste à rendre indépendantes les activités de conception de composants d'une part, et de réutilisation de ces mêmes composants d'autre part.
Loin de se limiter à un fichier de description synthétisable ou compilable, un composant virtuel se définit ainsi comme l'ensemble des informations qui permettront la réutilisation d'un bloc matériel ou logiciel pré-conçu et pré-vérifié [24, 25]. Le principal acteur en matière de recommandation et de standardisation concernant les composants virtuel est l'organisation *VSIA* (*Virtual Socket Interface Alliance*) [44], formée en 1996 dans le but de promouvoir la conception de systèmes intégrés par la réutilisation de composants provenant de sources variées.

**Niveaux d'abstraction d'un composant virtuel** – *VSIA* définit trois classes de composants virtuels matériels en fonction du niveau d'abstraction de leur description synthétisable [45] : (1) un composant virtuel *hard* se présente sous la forme de masques prêts à fondre ; (2) un composant virtuel *firm* se présente comme une *netlist* optimisée au niveau logique, et munie éventuellement de directives de placement ; (3) un composant virtuel *soft* se présente comme une description synthétisable en *VHDL* ou *Verilog* au niveau transfert de registres. Ces niveaux de description sont repérés sur la figure 1 dans le flot de conception d'un composant *IP*. Dans la pratique, les intégrateurs de composants virtuels privilégient les *IP soft* pour leur portabilité et leur flexibilité. Ces derniers autorisent en effet la personnalisation de la technologie cible et des contraintes d'optimisation logique, mais aussi de paramètres architecturaux.

Aujourd'hui, la conception de ces composants *IP* repose ainsi majoritairement sur les méthodes de conception et de synthèse au niveau transfert de registres (*RTL*). Pour l'implantation de fonctions de calcul intensif – de traitement du signal et de l'image, par exemple – cette approche apparaît de moins en moins adaptée à la complexité croissante des algorithmes d'une part, et d'autre part au degré élevé de flexibilité requis pour permettre leur réutilisation dans des contextes variés de besoins applicatifs, d'environnements systèmes, de contraintes de fonctionnement et d'implantation [2, 38, 43]. Malgré la généricité possible

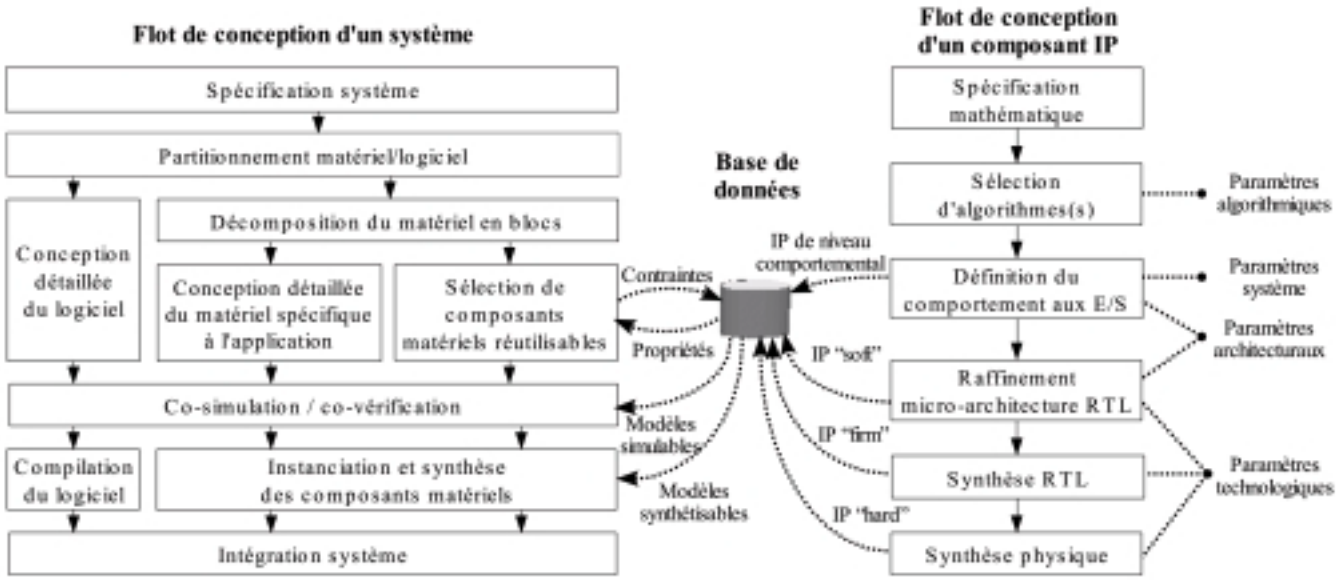


*Figure 1. Aperçu du flot de conception d'un système à base de composants matériels réutilisables*

avec les langages de description de matériel à ce niveau d'abstraction, la flexibilité est typiquement réduite au format des données, à des valeurs de constantes, d'indices de boucles, etc. L'architecture est par contre figée par la description elle-même et par conséquent le parallélisme est défini *a priori*, de même que l'ordre et le cadencement des entrées/sorties.

### 2.2 Explorer l'espace des solutions architecturales : la synthèse de haut niveau

La synthèse de haut niveau (*HLS* pour *High-Level Synthesis*) – également appelée synthèse *comportementale* ou *synthèse d'architecture* – s'inscrit dans une perspective d'automatisation du flot de synthèse système en réalisant une partie des tâches de raffinement des parties matérielles d'un système jusqu'au niveau transfert de registres (*RTL*) [16, 23]. Les techniques de synthèse de haut niveau font l'objet de développements et d'expérimentations dans les laboratoires académiques depuis une vingtaine d'années [32, 16, 34]. Elles sont à présent implantées dans des outils commerciaux distribués par des fournisseurs de *CAO* de premier plan [23] tels *Mentor Graphics* [13] et *Synopsys* [26] mais leur adoption par les concepteurs d'*ASIC* reste pour le moment très limitée [6].

La spécification d'entrée d'un outil de synthèse de haut niveau se veut aussi proche que possible du niveau algorithmique de manière à fournir, d'une part, un style de description intuitif, lisible et concis, et d'autre part à reporter les décisions d'implantation sur les étapes d'un flot de raffinement automatique ou interactif sous contraintes. Dans la plupart des outils de synthèse de haut niveau, la spécification d'entrée est dite « de niveau *comportemental* », c'est-à-dire qu'elle s'attache à décrire l'algorithme à implanter sous la forme d'un processus répétitif en utilisant des constructions syntaxiques proches de celles utilisées dans les langages de programmation.

De par son style d'écriture et sa faible complexité structurelle, une description comportementale est généralement plus facile à déboguer et plus rapide à simuler qu'une description *RTL*. Dans la majorité des cas, l'écriture d'un modèle comportemental est rendue d'autant plus aisée qu'un modèle fonctionnel de référence en *C* ou *C++* est généralement rédigé préalablement.

La synthèse de haut niveau peut ainsi être comparée à la compilation dans le domaine logiciel : il s'agit dans les deux cas de passer d'un modèle décrit à un haut niveau d'abstraction – en utilisant un style d'écriture de type algorithmique aussi indépendant que possible de la cible chargée de l'exécuter – à une description détaillée des instructions (code binaire exécutable par un processeur dans le cas du logiciel) ou des ressources matérielles (registres, opérateurs arithmétiques dans le cas d'un modèle *RTL*). Ainsi une description de matériel pour la synthèse de haut niveau manipule les concepts *abstraits* d'opérations, de variables, de structures de contrôle (boucles, instructions conditionnelles), laissant à des outils automatiques le soin de sélectionner, allouer et instancier des opérateurs, mémoires, et processus de communication de niveau logique.

### 2.3 Composants virtuels de niveau comportemental

Comme nous l'avons souligné plus haut, les niveaux d'abstraction applicables aujourd'hui aux descriptions synthétisables de composants virtuels matériels imposent de sérieuses limitations

en termes de flexibilité. Ces limitations ont un impact sur l'étendue des applications dans lesquelles un composant donné pourra être réutilisé. À terme, les techniques de *synthèse de haut niveau* ont, à notre avis, un rôle important à jouer dans le flot de réutilisation d'un composant, puisqu'elles autorisent une même description de haut niveau à être dérivée en une variété d'architectures matérielles en fonction des contraintes d'implantation fournies par l'intégrateur.

Dans cet article, nous introduisons ainsi la notion de composant virtuel *comportemental* : ce niveau de description apparaît sur la figure 1 dans le flot de conception d'un composant, en sortie de l'étape de raffinement du comportement aux entrées/sorties. Dans le flot de réutilisation, l'étape d'instanciation et de synthèse doit alors faire intervenir des outils de synthèse de haut niveau. La difficulté de cette démarche est alors de formaliser ce niveau de description, en mettant en évidence les propriétés d'un composant qui permettront de garantir sa *réutilisabilité*, en définissant les paramètres qui permettront à un concepteur de système d'adapter la fonctionnalité et les performances d'un composant aux exigences de son application, en identifiant les tâches qui sont du ressort de l'ingénieur et celle qui peuvent/doivent être traitées par des outils automatiques.

**Positionnement de notre approche vis-à-vis des travaux antérieurs** – Parmi les travaux ayant un lien avec la problématique que nous traitons, on trouvera essentiellement deux types de démarches :

- Utiliser les outils de synthèse de haut niveau pour concevoir rapidement des composants virtuels qui seront distribués sous forme de blocs *soft*, *firm* ou *hard*.

- Étendre la notion de synthèse de haut niveau à la synthèse système en automatisant la sélection et l'intégration de composants virtuels matériels.

Dans [36], une méthodologie de conception d'*IP soft* par synthèse de haut niveau est présentée. Cette méthodologie est fortement axée sur les fonctionnalités offertes par l'outil *Behavioral Compiler* de *Synopsys*. Essentiellement destinée aux applications orientées contrôle, elle se concentre sur le polymorphisme des interfaces de communication et ne s'intéresse pas à la personnalisation des aspects fonctionnels d'un composant.

Les travaux présentés dans [14] introduisent la réutilisation de composants virtuels dans le flot de synthèse de haut niveau proprement dit. Le principe est d'automatiser la sélection et l'allocation de composants fonctionnels complexes pour la synthèse de la partie matérielle d'un système.

À la différence de la synthèse de haut niveau telle que nous l'avons présentée plus haut, l'étape de sélection des ressources matérielles ne travaille donc plus sur des bibliothèques de composants de faible granularité – opérateurs arithmétiques – mais sur des bibliothèques de composants virtuels de plus ou moins grande complexité (filtre, transformation en cosinus discrète, etc). De tels composants ne sont plus purement combinatoires, mais possèdent chacun son propre protocole de communication. La synthèse d'une architecture à ce niveau d'abstraction s'accompagne par conséquent d'une synthèse automatique des interfaces entre composants complexes.

Dans les deux approches présentées ci-dessus, un composant réutilisable est délivré à l'intégrateur sous la forme d'une description de niveau *RTL* ou inférieur. Les techniques de synthèse de haut niveau ne sont pas exploitées pour rendre les composants flexibles, mais pour automatiser certaines activités de la conception jugées coûteuses en temps de développement. La démarche méthodologique que nous proposons vient en complément de ces deux approches. Comparée à [36], notre méthodologie élargit l'éventail des applications cibles aux fonctions de traitement du signal et de l'image. Comparée à [14], elle renverse le problème de la sélection d'un composant virtuel : il ne s'agit plus de trouver un composant présentant un compromis acceptable entre différents critères de choix, mais au contraire de personnaliser ce compromis au moyen des paramètres fournis par le composant choisi.

# 3.1 Transformations en ondelettes pour la compression d'images embarquée

La transformation en ondelettes discrète a largement démontré son efficacité lorsqu'elle est utilisée en tant que décorrélateur pour la compression des signaux en général, et des images en particulier [4]. Elle présente des avantages significatifs sur d'autres méthodes à base de transformations linéaires comme la transformation en cosinus discrète (*DCT*) supportée par le standard *JPEG* [19] : on retiendra notamment la grande variété des bases d'ondelettes disponibles, permettant d'ajuster les performances d'une chaîne de compression et sa complexité calculatoire en fonction des caractéristiques des images à compresser et des besoins de l'utilisateur. Ainsi le comité de normalisation *JPEG2000* [21] a retenu la transformation en ondelettes discrète pour sa flexibilité et ses performances supérieures à celles la *DCT* – performances mesurées en termes de rapport taux de compression/distorsion lorsque la transformation est insérée dans une chaîne de compression à pertes.

### 3.1 La transformation en ondelettes discrète

La transformation en ondelettes discrète peut être rapprochée de la notion d'analyse en sous-bandes [17, 41] dans laquelle le spectre du signal à analyser est décomposé récursivement en une sous-bande de *détails* (hautes fréquences) et une sous-bande d'*approximation* (basses fréquences). Pour un signal à

deux dimensions, ce ne sont pas deux, mais quatre sous-bandes qui sont extraites à chaque niveau de décomposition [41 31, 4] : une sous-bande d'*approximation* (*LL*) contenant l'information basse fréquence selon les deux directions spatiales ; une sous-bande de *détails horizontaux* (*LH*) et une sous-bande de *détails verticaux* (*HL*) contenant respectivement l'information basse fréquence suivant les directions horizontale et verticale, et l'information haute fréquence suivant l'autre direction ; une sous-bande de *détails diagonaux* (*HH*) contenant les composantes haute fréquence suivant les deux directions.

La transformée en ondelettes d'une image représente chacune de ces sous-bandes dans le domaine spatial et fournit ainsi une cartographie des détails en termes de localisation, résolution et direction. Le processus de décomposition récursive est représenté figure 2 pour trois niveaux de résolution [41, 31].

La méthode actuellement la plus employée pour calculer la transformation en ondelettes de signaux numériques est la méthode des bancs de filtres due à Stéphane MALLAT [31, 41]. Dans le cas de signaux monodimensionnels, la méthode des bancs de filtres repose sur un algorithme récursif dans lequel les cœfficients de la sous-bande d'approximation $v_f(j,k)$ à un niveau $j$ sont fournis simultanément à une paire de filtres numériques passe-bas ($H$) et passe-haut ($G$) produisant respectivement les cœfficients d'approximation et de détails au niveau $j-1$ (figure 3) [41, 31].

$$v_f(j-1,k) = \sum_{n\in\mathbb{Z}} h(n) v_f(j,2k-n)$$
$$w_f(j-1,k) = \sum_{n\in\mathbb{Z}} g(n) v_f(j,2k-n) \qquad (1)$$

Dans le cas de signaux bidimensionnels, chaque niveau de la tranformation en ondelettes discrète peut être calculé en appliquant successivement sur les lignes, puis sur les colonnes le même banc de filtres 1D. À chaque niveau, l'algorithme récursif prend alors pour point d'entrée l'image résultant d'un filtrage passe-bas dans les deux directions spatiales.

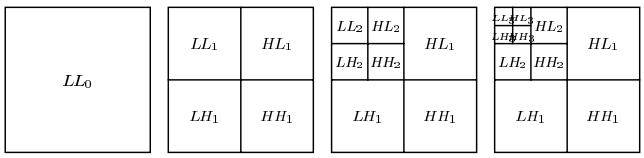


*Figure 2. Décomposition en ondelettes 2D sur trois niveaux*

Les deux principaux types de bancs de filtres utilisés sont d'une part les *filtres miroirs en quadrature* (*QMF*) et d'autre part les bancs de filtres *biorthogonaux* [9, 41]. La norme *JPEG2000* [21] recommande l'utilisation des bancs de filtres biorthogonaux symétriques 9/7 et 5/3 de COHEN, DAUBECHIES et FEAUVEAU [9]. Pour le banc 9/7, la réponse impulsionnelle du filtre passe-bas possède 9 cœfficients et celle du filtre passe-haut 7 cœfficients. Pour le banc 5/3, les filtres passe-bas et passe-haut possèdent respectivement 5 et 3 cœfficients.

### 3.2 Le Lifting Scheme

L'implantation la plus courante des bancs de filtres repose sur un calcul de convolution entre le signal à transformer et la réponse impulsionnelle des filtres [31, 41]. Les propriétés spécifiques des bancs de filtres biorthogonaux autorisent cependant des simplifications arithmétiques qui conduisent à une réduction significative de leur complexité calculatoire.

Le *Lifting Scheme*, mis au point par Ingrid DAUBECHIES et Wim SWELDENS [11], consiste à représenter un banc de filtres *1D* sous la forme d'une structure en échelle (figure 4). À chaque niveau de décomposition, une transformation en ondelette triviale est tout d'abord appliquée, qui consiste à séparer les échantillons d'indice pair des échantillons d'indice impair. Ensuite, la structure en échelle fait alterner

- des pas de *dual lifting* – ou *prédiction* – au cours desquels les échantillons d'indice impair sont remplacés par la différence $d_i(k)$ entre leur valeur et une valeur estimée calculée au moyen d'un filtre $P_i(z)$ en fonction des échantillons d'indices pair du voisinage ;

- des pas de *lifting* au cours desquels les nouveaux échantillons impairs sont utilisés pour mettre à jour les échantillons pairs $s_i(k)$ au moyen de filtres $U_i(z)$ (*Update*).

La norme *JPEG2000* recommande une implantation *lifting* du banc de filtres 9/7 dans laquelle la structure en échelle comprend deux pas de dual lifting, deux pas de lifting et un pas de scaling [11, 21] :

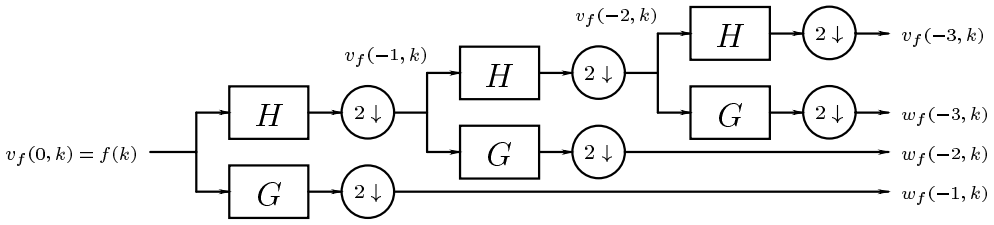


*Figure 3. Banc de filtres 1D pour la transformation en ondelettes sur trois niveaux*

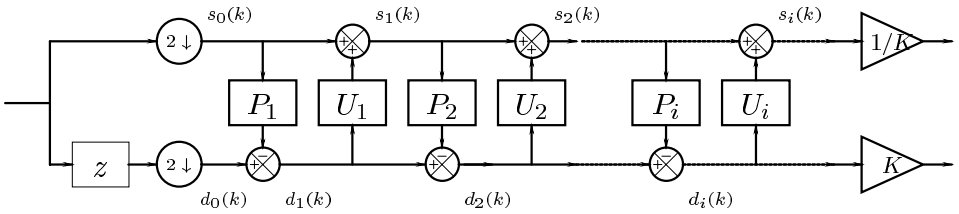


*Figure 4. Structure lifting d'un banc de filtres 1D*

$$
\begin{array}{lrcl}
\text{Séparation pairs/impairs} & s_0(k) & = & v_f(j,2k) \\
 & d_0(k) & = & v_f(j,2k+1) \\
\text{Premier pas de dual lifting}(P_1) & d_1(k) & = & d_0(k) + \alpha \times \big(s_0(k) + s_0(k+1)\big) \\
\text{Premier pas de lifting } (U_1)\} & s_1(k) & = & s_0(k) + \beta \times \big(d_1(k-1) + d_1(k)\big) \\
\text{Deuxième pas de dual lifting } (P_2)\} & d_2(k) & = & d_1(k) + \gamma \times \big(s_1(k) + s_1(k+1)\big) \\
\text{Deuxième pas de lifting } (U_2) & s_2(k) & = & s_1(k) + \delta \times \big(d_2(k-1) + d_2(k)\big) \\
\text{Scaling } (K,\, 1/K) & v_f(j-1,k) & = & s_2(k)/K \\
 & w_f(j-1,k) & = & d_2(k) \times K
\end{array}
\qquad (2)
$$

Les valeurs des cœfficients sont les suivantes [11, 21] :

$\alpha = -1{,}586134342$ ; $\beta = -0{,}05298011854$
$\gamma = 0{,}8829110762$ ; $\delta = 0{,}4435068522$ ; $K = 1{,}149604398$

La version *lifting* du banc de filtres 5/3 [11, 21] comprend un pas de dual lifting, un pas de lifting et s'accompagne d'opérations de troncature visant à produire des images transformées à valeurs entières tout en restant strictement réversible (équations 3). Si ce banc de filtres présente des performances en décorrélation nettement inférieures au banc 9/7, son intérêt réside cependant dans sa faible complexité [12]. Les divisions par 2 et par 4 ne nécessitent pas en effet d'opérateur diviseur : diviser ou multiplier par une puissance de 2 revient en fait à un simple décalage de bits.

$$
\begin{array}{lcl}
w_f(j-1,k) & = & v_f(j,2k+1) - \left\lfloor \frac{1}{2}\big(v_f(j,2k) + v_f(j,2k+2)\big) \right\rfloor \\
v_f(j-1,k) & = & v_f(j,2k) + \left\lfloor \frac{1}{4}\big(v_f(j,2k-1) + v_f(j,2k+1) + \frac{1}{2}\big) \right\rfloor
\end{array}
\qquad (3)
$$

Le tableau 1 compare la complexité calculatoire des algorithmes de transformation par convolution d'une part et en utilisant le *Lifting Scheme* d'autre part. La mesure de complexité est donnée en nombre d'opérations arithmétiques nécessaires au calcul d'une paire de cœfficients passe-bas/passe-haut à un niveau de résolution donné. La dernière colonne du tableau indique le nombre moyen de données – échantillons source ou résultats intermédiaires de filtrages précédents – utilisées lors de ce calcul. Cette valeur est étroitement liée à la quantité de mémoire requise pour l'implantation du banc de filtres. Une estimation plus complète de la quantité de mémoire nécessaire dans le cas monodimensionnel est donnée dans le tableau 2 pour deux algorithmes classiques de décomposition multi-niveau (voir paragraphe 3.3).

On observe que l'utilisation du *Lifting Scheme* permet de réduire de moitié le nombre d'opérations arithmétiques. Le choix du *Lifting Scheme* pour l'implantation matérielle de la transformation en ondelettes est ainsi doublement pertinent, tant du point de vue de la complexité calculatoire que du point de vue de la quantité de mémoire *a priori* nécessaire [11].

*Tableau 1. Complexité des bancs de filtres recommandés par la norme JPEG2000 pour différents algorithmes de filtrage*

| Banc de filtres | Algorithme | Additions/ soustraction | Multiplication décalages | Données |
|---|---|---|---|---|
| 9/7 | Convolution | 14 | 16 | 9 |
| | Lifting scheme | 8 | 6 | 6 |
| 5/3 | Convolution | 6 | 7 | 5 |
| | Lifting scheme | 5 | 2 | 4 |

### 3.3 Architectures VLSI de transformation en ondelettes discrète

Les architectures de bancs de filtres biorthogonaux reposent dans la majorité des cas sur une structure classique de filtre à réponse impulsionnelle finie, constituée d'une chaîne de cellules *MAC* (Multiplication-ACcumulation) et d'une ligne à retard. Un exemple d'une telle architecture est présenté dans [35]. Différentes améliorations de cette architecture sont possibles en tenant compte des caractéristiques des filtres utilisés (symétrie, antisymétrie) et du sous-échantillonnage en sortie des filtres de décomposition, qui permet de réduire la complexité calculatoire en ne calculant effectivement qu'un échantillon de sortie sur deux. Les tentatives d'implantation matérielle du *Lifting Scheme* sont nettement moins nombreuses que celles reposant sur un calcul de convolution. On notera en particulier dans [12, 3] des propositions pour les bancs de filtres biorthogonaux 9/7 et 5/3, reposant sur les équations 2 et 3 modifiées de manière à assurer la causalité.

Ces architectures de bancs de filtres fournissent la brique de base pour la réalisation d'architectures de transformation en ondelettes mono ou bidimensionnelles sur plusieurs niveaux. Elles peuvent s'insérer dans deux types d'architectures : les architectures *déroulées – unfolded –*, sont obtenues en mettant en cascade autant d'étages de filtrage que de niveaux de décomposition à effectuer [7]. La figure 3 représente fidèlement la structure d'une telles architecture dans le cas monodimensionnel. Ce type d'architectures présente deux inconvénients liés d'une part à sa complexité architecturale, du fait de la réplication du matériel nécessaire au calcul, et d'autre part à l'inhomogénéité de la répartition de la charge de calcul au cours du temps, chaque étage travaillant à la moitié de la cadence de son prédécesseur.

Les architectures *rebouclées – folded –* effectuent au contraire tous les niveaux de décomposition à l'aide d'un unique banc de filtres [7]. Celui-ci est couplé à un *réseau de routage* (figure 5) chargé de piloter la circulation des données en fonction d'un ordonnancement pré-établi des passes de filtrage. Un tel réseau de routage doit prendre en charge deux types de tâches qui sont d'une part le contrôle du flot de données et d'autre part la mémorisation des résultats intermédiaires de calcul [7]. La complexité du réseau de routage dépend de l'algorithme d'ordonnancement appliqué.

Différents modèles d'architectures de routage pour les architectures de type *unfolded* sont décrits dans [7]. Les deux principaux algorithmes d'ordonnancement sont [12, 7] : (1) l'*algorithme en pyramide* (*PA*), qui consiste à effectuer l'intégralité des passes de filtrage à un niveau de décomposition avant de passer au niveau suivant, et (2) l'*algorithme en pyramide récursif* (*RPA*), qui permet d'enchaîner des passes de filtrage sur plusieurs niveaux de décomposition à chaque nouvel échantillon parcouru. Le *RPA* réalise une transformation « au fil de l'eau » : au début de la transformation, le calcul et la production des données de sortie commence dès qu'une quantité suffisante de données d'entrée est disponible ; par la suite, calculs et sorties s'enchaînent au rythme de l'arrivée des entrées. Un tel procédé de calcul permet de minimiser la durée de vie des résultats intermédiaires de calcul, et par conséquent d'optimiser l'utilisation des ressources mémoire. Le tableau 2 donne une estimation de la quantité de mémoire nécessaire à l'implantation de ces deux algorithmes. Cette estimation est illustrée par l'exemple de l'ondelette 9/7. On observe que dans le cas du *RPA*, le *lifting scheme* permet une réduction significative (38 %) de la mémoire nécessaire, comparé à l'algorithme classique de filtrage par convolution.

**Extrapolation au cas bidimensionnel** – La transformation en ondelettes *2D* séparable permet d'utiliser la même architecture de banc de filtres pour chaque passe de décomposition horizontale ou verticale [31]. Au cours du parcours d'une image, l'algorithme en pyramide récursif enchaîne pour chaque pixel des passes de filtrage horizontal et des passes de filtrage vertical. Les filtres verticaux traitant directement les données en sortie des filtres horizontaux au même niveau, l'ordre de parcours des pixels est le même pour les deux directions du filtrage. Si l'on se borne à un parcours de type *raster* – pour lequel les pixels sont fournis ligne de blocs par ligne de blocs de haut en bas, et bloc par bloc de gauche à droite –, des filtres *1D* avec une structure de mémorisation de type ligne à retard sont bien adaptées au fil-

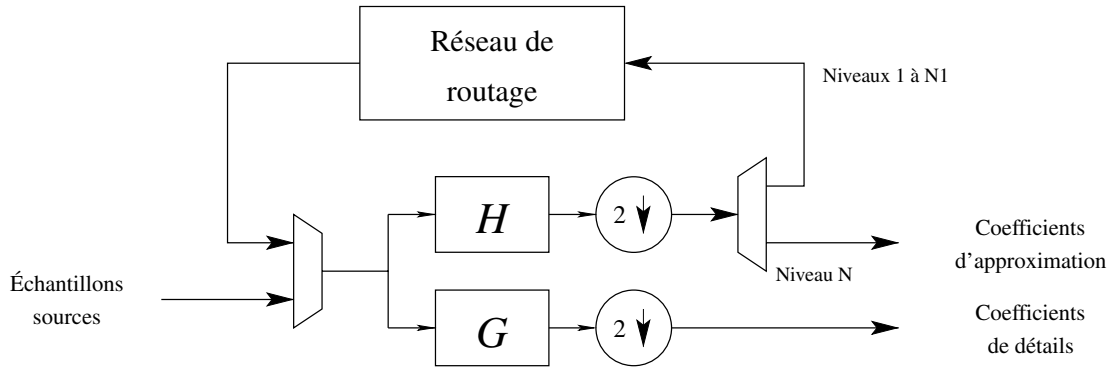


*Figure 5. Architecture rebouclée pour la transformation 1D multi-niveaux*

trage horizontal. En revanche, les filtres verticaux travaillent perpendiculairement à la direction du parcours (figure 6) et nécessitent une structure de mémorisation différente. La figure 6 montre que le temps séparant deux passes successives de filtrage suivant une colonne donnée de l'image correspond à la durée du parcours d'une ligne de pixels. Il s'ensuit que la taille mémoire nécessaire aux filtres verticaux est multiple de la largeur de l'image.

Dans [12], le banc de filtres lifting horizontal reçoit les pixels deux par deux (figure 7) et comporte deux cellules mémoire (opérateurs $z^{-1}$). Le banc de filtres vertical est conçu selon une architecture similaire où les éléments de mémorisation deviennent des lignes à retard ($z^{-L}$ pour des images de largeur $L$). Le banc de filtres vertical travaille une ligne sur deux et nécessite l'insertion de deux *FIFOS* supplémentaires à ses entrées.

L'architecture lifting présentée dans [3] comprend de même un bloc mémoire entre les bancs de filtres horizontaux et verticaux. Ce bloc mémoire, partitionné en deux à quatre bancs mémoire selon le banc de filtres choisi, sert à stocker aussi bien les données issues du filtrage horizontal que des résultats partiels de filtrage vertical. L'architecture globale, de type *folded*, requiert un bloc mémoire supplémentaire en entrée du banc de filtres horizontal, où seront stockés soit des pixels source, soit des cœfficients d'approximation issus des filtres verticaux au niveau de décomposition précédent.

L'implantation efficace du *RPA* nécessite un ordre parcours des pixels adapté à la structure récursive de l'algorithme : dans [27], l'architecture proposée réalise un parcours selon une courbe en *Z* fractal (ou courbe de MORTON). Les auteurs montrent qu'un tel parcours assure un débit régulier d'accès à la mémoire. La quantité de mémoire nécessaire à la transformation est réduite de 75 % comparée à l'implantation d'un parcours de type *raster*.

*Tableau 2. Estimation de la mémoire nécessaire pour l'implantation des algorithmes en pyramide (PA) et en pyramide récursif (RPA) monodimensionnels. $L =$ nombre d'échantillons à transformer ; $C =$ nombre moyen de données en entrée des filtres (cf tableau 1) ; $N =$ nombre de niveaux de décomposition. Application à l'ondelette 9/7*

| Algorithme | Mémoire au niveaux $n$ | Mémoire pour N niveaux | 9/7, $L = 256$ ; $N = 3$ Convulsion | Lifting |
|---|---|---|---|---|
| PA | $L/2^n$ | $<L/2$ | 96 | 96 |
| RPA | $C - 1$ | $(C - 1) \times N$ | 24 | 15 |

**Flexibilité et «réutilisabilité» de ces architectures** – La plupart des architectures recensées dans la littérature supportent un degré restreint de flexibilité, permettant de personnaliser un certain nombre de paramètres fonctionnels : nombre de niveaux de décomposition, réponse impulsionnelle des filtres d'ondelettes, taille des images. Pour un jeu donné de paramètres fonctionnels,

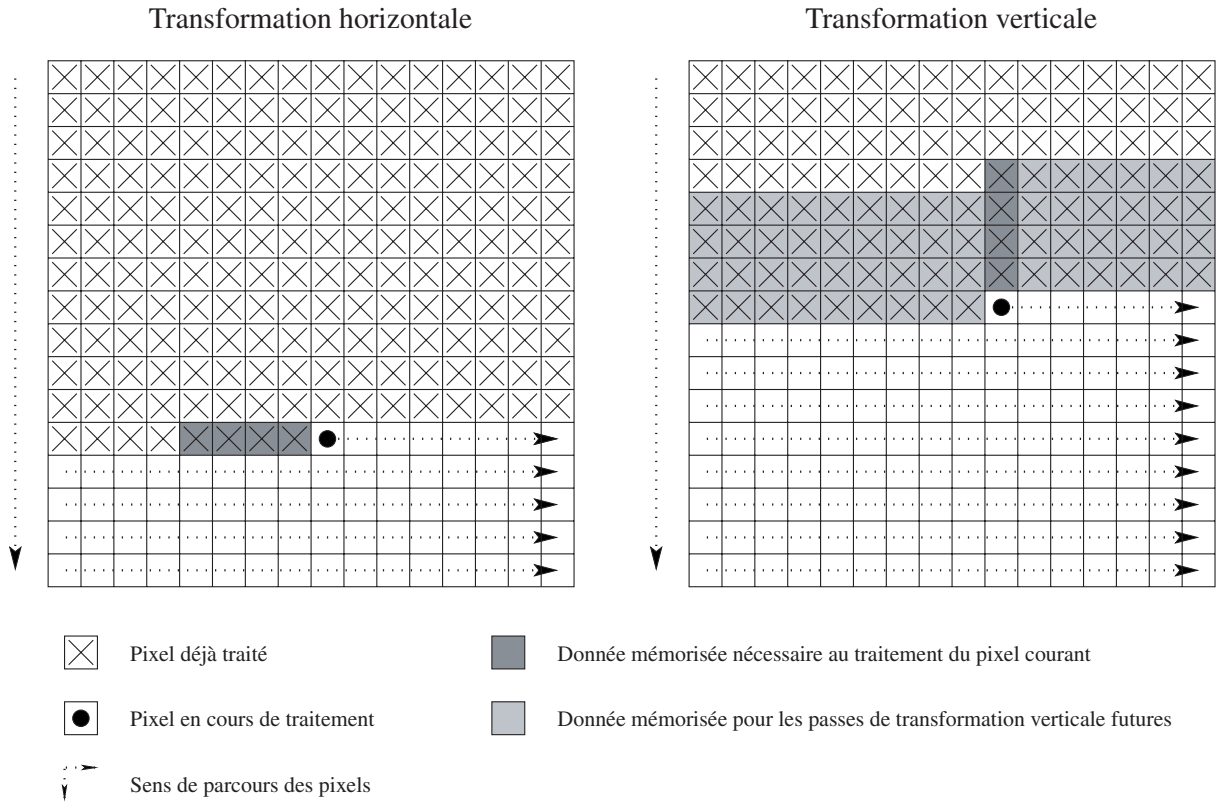


*Figure 6. Comparaison des besoins en mémoire des transformations horizontale et verticale pour un parcours de type raster*

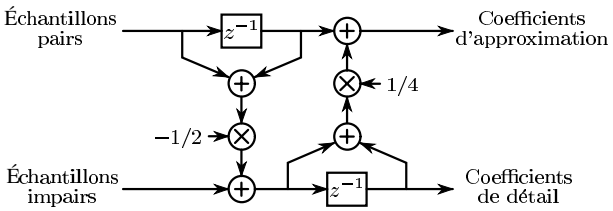


*Figure 7. Implantation du banc de filtres lifting 5/3 avec lecture des échantillons deux par deux*

nous obtenons pour chaque type d'implantation une architecture de complexité et de performances fixes : par exemple, le choix de la réponse impulsionnelle des filtres et du nombre de niveaux de décomposition conditionne directement la quantité de matériel (ressources de calcul et de mémorisation) à allouer. D'autre part, toutes ces architectures reposent sur un ordre et une cadence fixe des entrées/sorties. L'absence de paramètres liés à la communication est un des obstacles majeurs à la réutilisation : intégrer de telles architectures dans un système utilisant un ordre et une cadence d'entrées/sorties différents nécessite de concevoir des interfaces de communication qui peuvent être coûteuses en temps de développement et augmenter la complexité du système [10, 5, 18, 48].

Ces restrictions en termes de réutilisabilité sont principalement dues aux limitations des flots de conception classiques au niveau transfert de registres, qui ne permettent pas de traiter indépendamment les propriétés fonctionnelles d'un composant et les contraintes de complexité et de performances de son architecture. Dans le paragraphe qui suit, nous montrons comment notre démarche méthodologique, fondée sur l'utilisation rationnelle d'outils de synthèse de haut niveau, nous a permis de concevoir un composant virtuel de transformation en ondelettes hautement flexible, permettant de personnaliser séparément les propriétés liées à la fonctionnalité du composant et celles liées à ses performances.

# 4. Conception de composants virtuels flexibles de haut niveau

Les travaux méthodologiques qui ont permis la définition d'un flot de conception de composants virtuels de niveau comportemental s'inscrivent dans le cadre du projet *RNRT MILPAT* (*Méthodologie et développement pour les InteLlectual Properties pour Applications Télécom* [39]). Ce flot repose sur une démarche descendante au cours de laquelle chaque étape fait passer la spécification d'un niveau d'abstraction au niveau inférieur. Ces étapes successives de raffinement de la spécification d'un composant permettent d'ajouter de manière incrémentale des détails d'implantation, le point de départ étant une description de niveau algorithmique ne contenant aucune information de nature temporelle ou structurelle. Chaque étape enrichit également l'ensemble des paramètres et contraintes permettant de personnaliser les propriétés du composant à différents niveaux d'abstraction. Il est important de noter que l'objectif de la démarche proposée n'est pas d'aboutir à un flot entièrement automatisé dans lequel la contribution de l'ingénieur se limiterait à l'écriture d'un algorithme. Au contraire, nous nous sommes attachés à mettre en évidence, et à répartir sur chacune des étapes identifiées, les choix techniques qui relèvent de la compétence de l'ingénieur. Même s'il s'agit idéalement de conserver une certaine indépendance vis-à-vis des outils de conception, ces compétences doivent permettre une analyse critique des problèmes soulevés à chaque étape : analyse des besoins et des contraintes au niveau système ; sélection d'un compromis entre performances et complexité matérielle lors du réglage du parallélisme de calcul ; compréhension de la relation entre les contraintes de synthèse supportées par les outils et les modèles d'architecture de niveau *RTL*. Nous nous sommes ainsi attachés à caractériser les leviers dont le concepteur dispose pour effectuer ses choix et les résultats qu'il peut en attendre.

Les étapes de conception d'un composant virtuel comportemental telles que nous les décrivons ci-dessous sont transposables à une variété d'outils de *CAO* : elles ne sont pas liées à un fournisseur d'outils de synthèse de haut niveau particulier et peuvent même servir de point de départ au raffinement manuel de la description comportementale en une description détaillée de niveau transfert de registres.

### 4.1 Spécification au niveau algorithmique

Un algorithme décrit un calcul de manière purement transformationnelle. D'un point de vue externe, il ne fait ainsi intervenir aucune notion de temps, c'est-à-dire que les données qu'il utilise sont supposées être toutes disponibles au moment où le calcul commence. D'un point de vue interne, la description de l'algorithme précise les calculs à réaliser en termes d'opérations et de dépendances de données.

### 4.2 Raffinement des formats de données

Le choix des format des données aura généralement un impact sur des propriétés, en premier lieu de niveau fonctionnel (bruit de calcul), mais également de niveau *RTL* lors de l'intégration (largeur de bus, surface et temps de traversée des opérateurs arithmétiques, taille mémoire). L'analyse de la propagation des erreurs de calcul au cours du déroulement de l'algorithme permet de dimensionner le chemin de données de manière à sélectionner un compromis acceptable entre bruit de calcul, vitesse et complexité matérielle [33].

Sans négliger la question de la précision dans le dimensionnement du chemin de données, nous avons choisi de ne pas

détailler cet aspect dans le présent article. Nous avons choisi au contraire de privilégier la problématique de la génération d'un chemin de données dont le parallélisme de calcul respecte des contraintes de temps et d'entrée/sortie. Nous recommandons au lecteur de se référer à [33] pour une présentation plus détaillée des aspects liés à la précision pour la synthèse de haut niveau.

#### 4.3 Extraction d'un motif répétitif de calcul

Les algorithmes de traitement du signal et de l'image possèdent dans la majorité des cas une forte régularité qui autorise leur écriture sous forme de *nids de boucles*, c'est-à-dire d'une séquence d'opérations élémentaire incluse dans une hiérarchie de boucles imbriquées. Le raffinement d'une description de niveau algorithmique en une description comportementale, c'est-à-dire possédant une notion de temps, consiste ainsi à mettre en évidence des séquences répétitives dans le déroulement de l'algorithme, qui permettront de répartir de manière régulière la charge de calcul au cours du temps. Les graphes de dépendances polyédraux (*PDG*) [15], utilisés notamment en synthèse d'architectures systoliques avec l'outil *MMAlpha* [47, 46], fournissent un cadre formel efficace pour l'analyse de dépendances et la mise en évidence de tels motifs. Un motif répétitif de calcul sera modélisé sous la forme d'un processus activé périodiquement au cours du temps et effectuant à chacune de ses itérations une partie du calcul à réaliser. Chaque nouvelle itération consomme une quantité constante de données d'entrée et produit une quantité constante de données de sortie. À ce stade de la conception, le modèle de communication du composant avec son environnement est de type *flot de données synchrone* (*SDF*), c'est-à-dire que le déclenchement d'une itération du processus est conditionné par la présence d'une quantité minimale, constante, de données à ses entrées. Un même algorithme peut autoriser l'extraction d'une variété de motifs présentant différentes propriétés de complexité et de temps. Afin d'être flexible, la description comportementale devra supporter un certain nombre de paramètres permettant de configurer :

- le *nombre d'opérations par itération* : ce nombre est proportionnel à la *taille* du motif ;

- le *parallélisme* maximal exploitable entre opérations à l'intérieur d'une itération du motif ;

- le nombre de *transferts* de données entre l'itération courante, l'environnement et les itérations précédentes ;

- *la durée de vie* des données, c'est-à-dire le nombre d'itérations du motif s'écoulant entre la production et la dernière utilisation de chaque donnée.

L'analyse de la quantité de transferts de données par itération et de la durée de vie de ces données permet de choisir le type de ressources de mémorisation qui minimisera le coût en surface tout en assurant le respect de la contrainte temporelle : typiquement, les données à durée de vie courte seront mises en registres tandis que les données à durée de vie longue seront stockées en *RAM*

#### 4.4 Insertion des contraintes de temps et d'entrée/sortie

À ce niveau de spécification, le composant est muni de paramètres temporels permettant à l'intégrateur système de personnaliser la date au cycle près des entrées/sorties et la durée d'exécution d'une itération du motif de calcul. Ces paramètres se traduiront par un degré plus ou moins élevé de parallélisme d'entrée/sortie, fixant une taille minimale des ports d'entrée/sortie.
Dans certains outils de synthèse de haut niveau, les contraintes de temps et d'entrée/sortie peuvent être spécifiées au cycle près, soit directement dans le code comportemental *Monet*, *Behavioral Compiler* et *CoCentric SystemC Compiler* en mode *cycle-fixed* ; *Get2Chip Architectural Compiler*), soit au moyen de directives de synthèse (version de *GAUT* en cours de développement au *LESTER* [29], projet « *Tsunami* » de *Mentor Graphics* [42]). La définition exhaustive du comportement aux entrées/sorties est garante de la prédictibilité des performances d'un composant : elle permet ainsi de s'affranchir des algorithmes utilisés dans les outils de synthèse. Une fois les contraintes de temps et d'entrée/sortie figées par l'intégrateur système, les libertés offertes aux outils pour ordonnancer les calculs sont en effet délimitées par un cadre rigide qu'ils s'efforceront de respecter – et ce quelle que soit la technique d'ordonnancement utilisée.

#### 4.5 Décomposition structurelle d'un motif

Lors de la synthèse de haut niveau, le modèle raffiné d'un composant virtuel comportemental sera décomposé en trois unités : (1) l'unité de *traitement* décrit la partie purement calculatoire du motif de calcul ; (2) l'unité de *mémorisation* modélise la structure de stockage des données intermédiaires à durée de vie longue ; (3) l'unité de *contrôle* modélise le déroulement temporel du calcul sous la forme d'une machine à état fini.
L'action de l'unité de contrôle se situe à deux niveaux. Au niveau *cycle*, elle pilote les transferts de données entre opérateurs arithmétiques de l'unité de traitement au cours d'une itération du motif. Au niveau *itération*, elle permet de modéliser les exécutions successives du motif comme autant de *macro-états* dont peuvent éventuellement dépendre les calculs à réaliser.
Un cas typique illustrant ce deuxième point est celui où un traitement spécifique – s'étalant éventuellement sur plusieurs itérations du motif – doit être appliqué au voisinage des bords du domaine des données d'entrée/sortie. Nous rencontrons par exemple cette situation dans le cas de la transformation en ondelettes appliquée à des signaux de taille finie comme les images : au voisinage des bords, les algorithmes de filtrage valides en régime établi ne sont plus adaptés car ils consommeraient des pixels situés hors des limites de l'image. La parade généralement adoptée consiste à étendre artificiellement l'image à tout le plan, soit en affectant une valeur nulle aux pixels situés au-delà de ses bords (zero-padding), soit en la périodisant, ou encore en réalisant une extension par symétrie périodique. Le standard *JPEG2000* a adopté cette dernière méthode d'extension.

# 5. Conception d'un composant virtuel comportemental pour la transformation en ondelettes discrète 2D

Dans cette section, nous nous intéressons tout d'abord à l'extraction d'un motif répétitif de calcul pour la transformation en ondelettes *au fil de l'eau* d'une image, aboutissant à la rédaction d'une description comportementale supportant une large gamme de paramètres. Nous présentons ensuite des résultats de synthèse réalisés à partir de cette description comportementale pour différentes valeurs de ses paramètres.

### 5.1 Algorithme de transformation en ondelettes lifting 2D

Le document de référence qui a orienté le choix de l'algorithme de transformation en ondelettes est le *Final Committee Draft* de la norme de codage d'images *JPEG2000* [21]. L'algorithme préconisé consiste en une décomposition à base d'ondelettes biorthogonales calculée en appliquant la méthode du *Lifting Scheme*. Nous présentons dans le listing 1 l'algorithme de transformation en ondelettes $2D$ sur $N$ niveaux sous forme de pseudo-code. Pour chaque niveau de résolution, une passe de filtrage *1D* est appliquée d'abord suivant chaque ligne, puis suivant chaque colonne. Chaque passe de filtrage *1D* se fait en $S+1$ étapes, alternant tout d'abord $S/2$ pas de *dual lifting* et $S/2$ pas de *lifting*, et se terminant par un pas de *scaling*. Dans l'écriture de cet algorithme, nous avons tenu compte des propriétés spécifiques aux ondelettes utilisées dans le standard *JPEG2000* [21] : (1) le nombre $S$ de pas de lifting/dual lifting pour un étage de transformation *1D* est supposé pair ; (2) une passe de transformation *1D* commence toujours par un pas de dual lifting ; (3) chaque filtre de prédiction $P_i(z)$ ou de mise à jour $U_i(z)$ a une réponse de la forme : $\zeta(1+z)$ ou $\zeta(1+z^{-1})$, respectivement.
Notre implantation du composant virtuel impose des restrictions aux valeurs de paramètres supportées par le standard. Les paramètres de niveau fonctionnel sont les suivants :

- choix de l'ondelette : 9/7 ou 5/3 ;
- nombre de niveaux de décomposition : $N \geq 1$ ;
- largeur/hauteur d'une tuile, en pixels : $L$, $H$ ($H$ éventuellement infinie).

Pour une question de régularité de l'algorithme, nous imposons à $L$ et $H$ d'être multiples de $2^N$.
Le graphe de dépendances de l'algorithme de transformation *2D* est présenté en figure 8 pour une décomposition sur 3 niveaux d'une image de 32 pixels de large avec l'ondelette 9/7 ($N=3$ et $S=4$). Chaque nœud est repéré par un quintuplet d'indices $(l,k,n,d,s)$ où

- $l$ est l'indice de la ligne de pixels courante dans l'image ;
- $k$ est l'indice de la colonne de pixels courante dans l'image.
- $n$ est l'indice du niveau de décomposition courant : $1 \leq n \leq N$ ;
- $d$ représente la direction de la passe de filtrage courante : $d=0$ pour un filtrage horizontal ; $d=1$ pour un filtrage vertical ;
- $s$ est l'indice du pas de calcul courant : $s=0$ pour les données source ; $s$ impair et $1 \leq s < S$ pour un pas de *dual lifting* ; $s$ pair et $1 < s \leq S$ pour un pas de *lifting* ; $s=S+1$ pour un pas de *scaling*.

La figure 8 donne une vue « en coupe » du graphe complet, dans laquelle seuls les nœuds relatifs aux passes de transformation horizontale suivant une ligne de l'image ont été représentés. Les nœuds en grisé représentent chacun, selon l'indice $s$ des données qu'ils consomment, une opération de *dual lifting*, de *lifting* ou de *scaling*.

**Propriétés** – Le tableau 3 donne le nombre moyen d'opérations arithmétiques par pixel d'entrée ou de sortie, exprimé à différents niveaux de granularité : (1) en pas de *lifting* ou *dual lifting* et multiplications de *scaling* d'une part, et (2) en additions et multiplications d'autre part. Ce nombre d'opérations est donné tout d'abord dans le cas d'une transformation *1D* sur un niveau. Pour une transformation *2D* sur 1 niveau, ce nombre doit être multiplié par deux afin de tenir compte des deux directions spatiales. Pour $N$ niveaux de décomposition *2D*, nous effectuons la somme des nombres moyens d'opérations par pixel pour chaque niveau de décomposition, la contribution d'un niveau d'indice $n$ étant égale au quart de la contribution du niveau $n-1$ du fait du sous-échantillonnage par deux dans les deux directions spatiales. On observe que l'augmentation du nombre de niveaux de décomposition n'a que peu d'influence sur le nombre moyen d'opérations par pixel, qui reste borné par une quantité ne dépendant que de la complexité des filtres *lifting* (complexité représentée par le nombre de pas de *lifting* et *dual lifting* $S$).
L'algorithme présente une structure très régulière qui consiste en la répétition d'une opération *lifting* ou *dual lifting* composée de deux additions et d'une multiplication (figure 9).
Au niveau algorithmique, la spécification du composant virtuel n'intègre aucune notion de temps. Il est cependant possible d'estimer la durée minimale de calcul d'une donnée de sortie en faisant l'hypothèse que toutes les données d'entrée dont elle dépend sont disponibles simultanément et que le calcul exploite le maximum de parallélisme possible entre opérations. Pour ce faire, nous dénombrons les opérations figurant sur la plus longue chaîne de dépendances entre une donnée de sortie et les données d'entrée dont elle dépend. Le tableau 4 indique les quantités obtenues pour les cœfficients de détail à chaque niveau $n$ ($1 \leq n \leq N$) et pour les cœfficients d'approximation au niveau $N$. La durée minimale du calcul est exprimée en fonction de la durée d'une addition ($T_{add}$) et de la durée d'une multiplication ($T_{mult}$), sans présupposer de la présence d'une horloge rythmant les calculs et en négligeant les temps de synchronisation et de communication.

*Listing 1. Transformation en ondelettes lifting 2D pour JPEG2000*

```
' pour chaque niveau de décomposition
pour n de 1 à N faire
    ' ———————— Transformation horizontale
    ' pour chaque ligne de l'image
   pour l de 0 à H par 2^(n-1) faire
      ' Pour chaque pas de lifting/dual lifting
      pour s de 1 à S faire
         kmin ←2^n − 1 si s pair, 0 sinon
         kmax ← L − 2^n − 1 si s pair, L − 2^n sinon
         pour k de kmin à kmax par 2^n faire
            x(l,k) ← x(l,k) + ζ(s) × (x(l,k − 2^n − 1) + x(l,k + 2^n − 1))
         fin pour (k)
     fin pour (s)
     ' Scaling
     pour k de 0 à L − 2^n par 2^n faire
         x(k) ← x(k)/K   '  Passe-bas
         x(k + 2^n − 1) ← x(k + 2^n − 1) × K ' Passe-haut
     fin pour (k) fin pour (l)
     '———————— Transformation verticale
     ' pour chaque colonne de l'image
     pour k de 0 à L par 2^(n-1) faire
         ' Pour chaque pas de lifting/dual lifting
         pour s de 1 à S faire
            {...}
         fin pour (s)
          ' Scaling
         pour l de 0 à H − 2^n par 2^n faire
            {...}
         fin pour (l)
     fin pour (k)
fin pour (n)
```

*Tableau 3. Nombre d'opérations par pixel pour la transformation en ondelettes lifting 2D*

| | *DWT 1D*<br>1 niveau | *DWT 2D*<br>$N$ niveaux | *DWT 2D}*<br>$N = 3, S = 4$ |
|---|---|---|---|
| *(Dual) Lifting* | $S/2$ | $S\sum_{n=1}^{N} 1/4^{n-1} < 4S/3$ | 5,3 |
| *Scaling* | 1 | $2\sum_{n=1}^{N} 1/4^{n-1} < 8/3$ | 2,6 |
| Additions | $S$ | $2S\sum_{n=1}^{N} 1/4^{n-1} < 8S/3$ | 10,5 |
| Multiplications | $S/2+1$ | $(S+2)\sum_{n=1}^{N} 1/4^{n-1} < (4S+8)/3$ | 7,9 |

mier type d'implantation, le composant virtuel est maître des accès mémoire et traite les données à son propre rythme. Le second type est plus adapté aux applications temps réel dans lesquelles le traitement doit être effectué au rythme de l'acquisition des données.

Dans le contexte de la compression d'images spatiales, le coût du stockage des données à bord des satellites ne permet pas de conserver en mémoire la totalité des données image [30, 28]. L'acquisition en continu de lignes de pixels impose le traitement en temps réel des données. Les techniques de compression d'image *au fil de l'eau* permettent de coder et de transmettre les données au fur et à mesure qu'elles sont acquises en minimisant la mémoire nécessaire au traitement : typiquement, à chaque fois qu'une ligne ou un groupe de lignes de pixels a été codé, ne sont conservées en mémoire que les données nécessaires au codage de la prochaine ligne ou du prochain groupe de ligne.

### 5.2 Transformation en ondelettes 2D au fil de l'eau

En fonction des contraintes de l'application et du système cible, deux types d'implantation d'un composant virtuel de transformation en ondelettes peuvent être envisagés : (1) le premier consiste à implanter l'algorithme de transformation présenté dans la section précédente en supposant que tous les pixels de l'image source sont disponibles en mémoire dès le début du calcul ; (2) le second suppose que les pixels sont fournis au composant dans un ordre déterminé au cours du temps. Dans le pre-

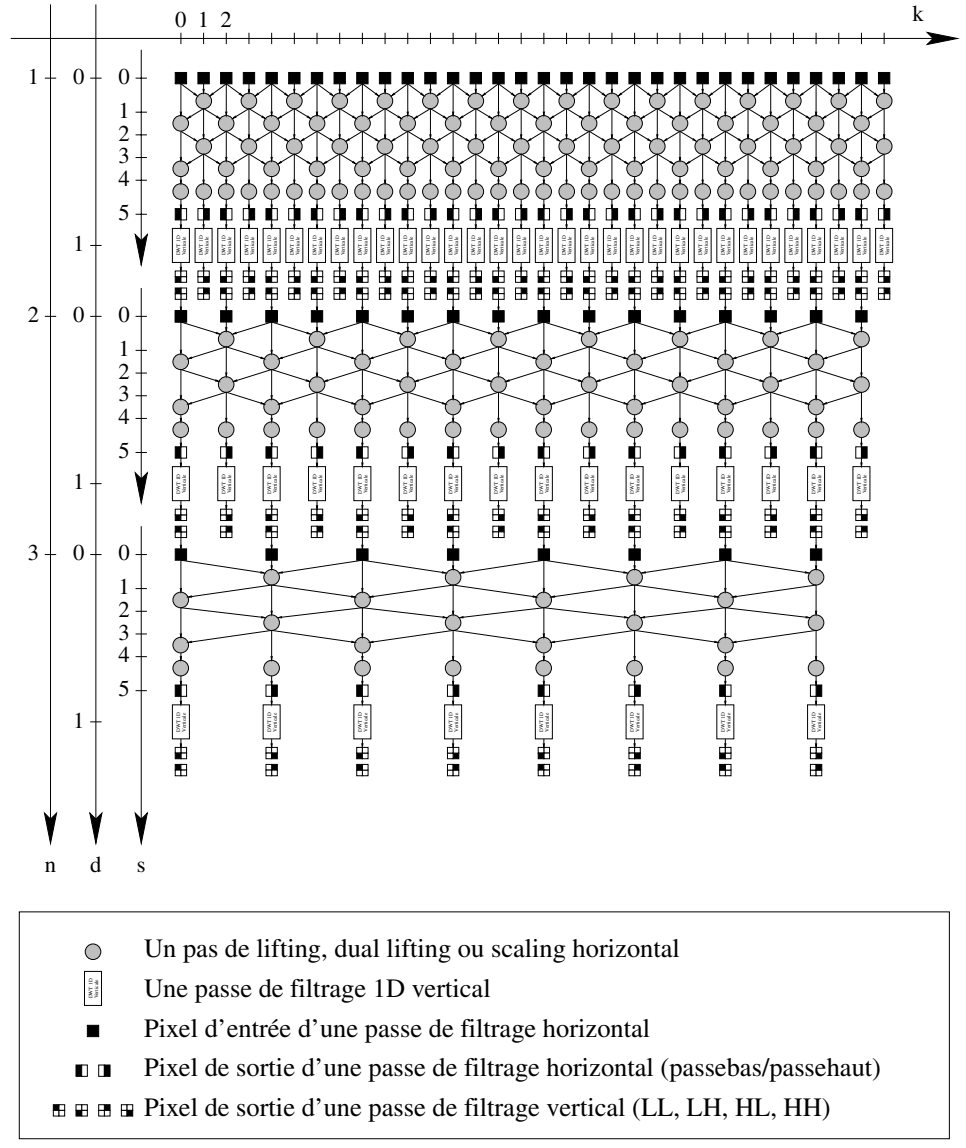


*Figure 8. Graphe de dépendances de la DWT 2D Lifting sur trois niveaux pour l'ondelette 9/7*

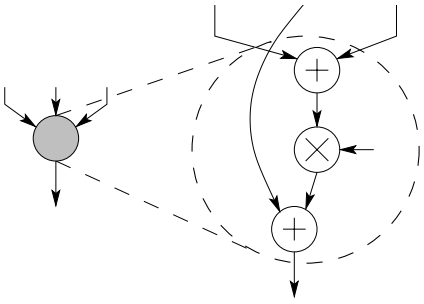

*Figure 9. Anatomie d'un nœud lifting ou dual lifting*

*Tableau 4. Estimation de la durée minimale de calcul d'une donnée de sortie de la transformation en ondelettes lifting 2D*

| | Détail (niveau $n$) | Approximation (niveau $N$) |
|---|---|---|
| Additions | $2(Sn-1)$ | $2SN$ |
| Multiplications | $(S+1)n-1$ | $(S+1)N$ |
| Latence pour $N=3$ et $S=4$ | $(8n-2)T_{add}$ $+(5n-1)T_{mult}$ | $24T_{add}$ $+15T_{mult}$ |

Cette problématique nous a conduit à rechercher dans le graphe de dépendances de la transformation *lifting* d'une image entière des motifs simples correspondant à différentes stratégies de transformation au fil de l'eau et répondant à différents compromis vitesse/complexité/quantité de mémoire [40]. Nous nous sommes efforcés de spécifier en une unique description comportementale générique un éventail de motifs pouvant répondre à différents jeux de contraintes.

### 5.2.1 Choix des axes temporels et causalité

Par sa définition même, la notion de *motif répétitif de calcul* (voir paragraphe 4.3) nécessite une modélisation de l'écoulement du temps à travers le graphe de dépendances. À ce stade de la conception, les seules informations dont nous disposons pour ordonnancer les opérations nous sont fournies par l'analyse des dépendances de données entre ces mêmes opérations. À cette contrainte *interne* à l'algorithme, nous allons superposer de manière progressive des contraintes *externes*, c'est-à-dire définissant de plus en plus précisément l'ordre et la date de consommation (resp. de production) des données d'entrée (resp. de sortie).

Dans le cas de la transformation en ondelettes 2D, nous posons tout d'abord des contraintes sur l'ordre de parcours des pixels de l'image à transformer. Pour assurer la flexibilité du composant dans son comportement aux entrées/sorties, ces contraintes visent non pas à figer un parcours, mais à délimiter un ensemble de parcours possibles. Nous posons par exemple que le parcours se fait dans l'ordre croissant des indices de ligne ($l$) d'une part, et dans l'ordre croissant des indices de colonne ($k$) d'autre part, sans privilégier une dimension par rapport à l'autre. Ces contraintes définissent ce que nous appellerons les *axes temporels* du graphe : dans notre cas, nous aboutissons à la définition de deux *axes temporels* portés respectivement par l'axe des $l$ et l'axe des $k$.

Si nous tentons à présent de recouper cette contrainte externe avec la contrainte interne que sont les dépendances de données, la lecture de la figure 8 montre cependant que certaines chaînes de dépendances se déroulent dans l'ordre décroissant des indices $l$ et $k$, mettant ainsi en évidence des dépendances *non causales*. Si les deux axes choisis sont pertinents pour fixer des *directions* de parcours, ils apparaissent inadaptés à la représentation des *dates* d'exécution. Un changement d'indices $(l,k) \Longrightarrow (r,q)$ permet de résoudre cette difficulté :

- pour $d = 0$ et $0 \leq s \leq S$ :

$$q = k + \Big(S \times (1+d) + s \times (1-d)\Big)2^{n-1} - S\,;$$

$$r = l + (S + s \times d)2^{n-1} - S$$

- pour $s = S + 1$ :

$$q = k + S \times (2^n - 1)\ ;\ r = l + S \times (2^{n+d-1} - 1)$$

Du point de vue des données d'entrée, les axes définis par $r$ et $q$ sont identiques aux axes des $l$ et $k$ définis précédemment. Du point de vue des opérations, les nouveaux indices donnent une représentation cohérente de l'ordre d'exécution possible en respectant la causalité le long des chaînes de dépendances. Le changement d'indice selon la direction horizontale est représenté en figure 10 pour la transformation sur 3 niveaux avec l'ondelette 9/7 [40].

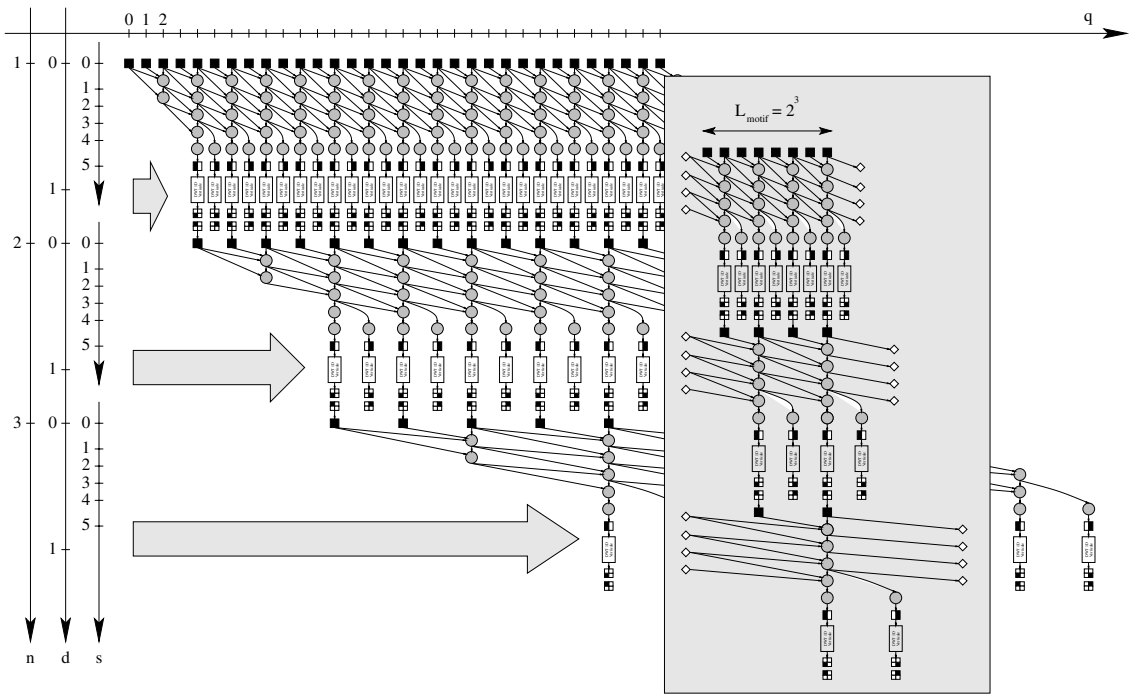


*Figure 10. Graphe de dépendances causal et extraction d'un motif répétitif de calcul pour la DWT 2D Lifting 9/7*

### 5.2.2 Choix de motifs de calcul

L'extraction d'un motif répétitif de calcul pour la transformation en ondelettes *lifting* repose sur l'observation que le graphe de dépendances – à l'exception des nœuds situés au voisinage des bords – présente une périodicité de $2^N$, ($N$ étant le nombre de niveaux de décomposition) suivant les indices $q$ et $r$ définis plus haut [40]. L'étude des périodicités et l'extraction de motifs sont pour le moment réalisées manuellement ; ces étapes reposent sur des parcours systématiques de graphes et peuvent être automatisées.

L'exploitation directe de cette périodicité permet d'extraire un motif de la forme schématisée en figure 10. Un tel motif a une largeur $L_{motif}$ (selon l'axe des $q$) et une hauteur $H_{motif}$ (selon l'axe des $r$) multiples de $2^N$. En d'autres termes, on peut définir deux entiers strictement positifs $w$ et $h$ tels que :

$$L_{motif} = w \times 2^N$$

$$H_{motif} = h \times 2^N$$

Sur la figure 10, le motif présenté correspond ainsi au cas $w = 1$ ; l'axe des $r$ n'étant pas représenté, cette figure peut correspondre à toute valeur de $h$. Chaque itération du motif consomme un bloc de $L_{motif} \times H_{motif}$ pixels de l'image d'origine et produit la même quantité de pixels de l'image transformée (cœfficients d'approximation au niveau $N$ et de détail aux niveaux 1 à $N$). Le composant peut ainsi être inséré dans un environnement de communication conforme au modèle *flot de données synchrone* (*SDF*).

À chaque itération, ce motif consomme également des données intermédiaires (ou données de contexte) calculées lors d'itérations précédentes. Une partie des données intermédiaires produites lors de l'itération courante sera ainsi réutilisée lors d'itérations ultérieures. La durée de vie de ces données, exprimée comme le nombre d'itérations du motif entre leur production et leur dernière consommation, dépend de l'ordre de parcours des blocs de pixels dans l'image. Typiquement, dans un parcours de type *raster* (paragraphe 3.3) le contexte horizontal est mis à jour à chaque itération du motif tandis que les données de contexte vertical en sortie de l'itération courante ne seront réutilisées que lorsqu'une ligne de blocs complète aura été traitée.

Le tableau 5 donne les grandeurs caractéristiques de la complexité du motif, en nombre d'opérations arithmétiques d'une part, et en nombre de transferts de données par itération d'autre part.

Le tableau 6 donne la durée de vie des données du contexte horizontal et vertical dans le cas d'un parcours *raster*. La quantité de mémoire est exprimée comme le nombre de valeurs produites par le motif sur un intervalle de temps égal à la durée de vie des données de contexte. La quantité de mémoire nécessaire au stockage du contexte horizontal ne dépend que de la *hauteur* $H_{motif} = h \times 2^N$ du motif, de la complexité $S$ du banc de filtres *lifting* et du nombre $N$ de niveaux de décomposition. Observons par ailleurs que la quantité de mémoire requise par le contexte vertical est bornée en valeur supérieure par une quantité ne dépendant que de la largeur de l'image et de la complexité de l'ondelette. Ces deux quantités ne dépendent pas de la largeur $L_{motif}$ du motif : il est donc possible de choisir un motif arbitrairement large – *e.g.* afin d'ajuster le parallélisme de calcul exploitable – sans pour autant augmenter la quantité de mémoire nécessaire. La quantité importante de données du contexte vertical devant être stockées et leur longue durée de vie privilégie un support de stockage de type *RAM* externe, le temps d'accès n'étant pas critique pour les traitements. Selon les performances en vitesse souhaitées, les données du contexte horizon-

*Tableau 5. Propriétés du motif de transformation en ondelettes lifting 2D*

| | Cas général | $w = h = 1$<br>$N = 3$ ; $S = 4$ |
|---|---|---|
| Additions | $8S(4^N - 1)wh/3$ | 672 |
| Multiplications | $(4S + 8)(4^N - 1)wh/3$ | 504 |
| Pixels entrée | $4^N \times wh$ | 64 |
| Pixels sortie | $4^N \times wh$ | 64 |
| Contexte H. E/S | $2Sh(2^N - 1)$ | 56 |
| Contexte V. E/S | $2Sw(2^N - 1)$ | 56 |
| Total entrées | – | 176 |
| Total sorties | – | 176 |

*Tableau 6. Durée de vie des données et contraintes de mémorisation pour le motif de transformation en ondelettes lifting 2D, avec un parcours raster par blocs des pixels de l'image source*

| | Cas général | | $L = 256$ ; $w = h = 1$<br>$N = 3$ ; $S = 4$ | | |
|---|---|---|---|---|---|
| Données | Durée de vie | Qté de mémoire | Durée de vie | Qté de mémoire | Support |
| Contexte horizontal | 1 | $2Sh(2^N - 1)$ | 1 | 56 | Registres *RAM* interne |
| Contexte vertical | $L/L_{motif}$ | $2S(1 - 1/2^N)L$<br>$< 2SL$ | 32 | 1792 | *RAM* externe |

tal pourront être soit stockées dans une *RAM* interne au composant virtuel – le rapport temps d'accès aux données/temps de traitement associé à un nœud risque alors de limiter la cadence des traitements – soit mises en registres si on favorise avant tout la cadence de traitement mais au prix d'une plus grande complexité matérielle de l'architecture.

### 5.2.3 Synthèse de haut niveau et optimisations logiques

La structure très régulière du *Lifting Scheme* nous permet de considérer des possibilités de regroupement des opérations de manière à optimiser la synthèse du chemin de données [40]. Il est en effet possible de prendre en compte dans le flot de synthèse de haut niveau les possibilités d'optimisation au niveau logique propres aux outils de synthèse *RTL* en synthétisant séparément sous forme de circuits combinatoires des composants constitués de plusieurs opérateurs arithmétiques. Là où un outil de synthèse de haut niveau cherchera par exemple à ordonnancer les trois opérations d'un nœud *lifting* soit dans des cycles d'horloge séparés, soit en les chaînant à l'intérieur d'un même cycle, un outil de synthèse *RTL* produit un composant unique et optimisé dont les performances sont généralement meilleures que lorsque des opérateurs indépendants sont mis en cascade. Le concepteur a donc tout intérêt à tester plusieurs solutions d'optimisation *locale* au niveau *RTL* avant de procéder à la synthèse de haut niveau de l'ensemble.

La figure 11 présente trois possibilités de regroupement des opérations associées au motif de transformation *1D*. Ce motif est représenté en figure 11a à base d'opérations arithmétiques élémentaires (additions, multiplications). Il est possible de synthétiser chaque nœud *lifting* ou *dual lifting* sous la forme d'un composant combinatoire intégrant deux additionneurs et un multiplieur (figures 9 et 11b). Un degré supplémentaire de regroupement est proposé en figure 11c, qui consiste à intégrer une chaîne de *S* nœuds *lifting* et *dual lifting* dans un même composant combinatoire. Ici, des optimisations logiques supplémentaires sont possibles du fait que tous les cœfficients de *lifting* sont câblés à l'intérieur de ce composant, ce qui permet de simplifier l'architecture des multiplieurs. Le tableau 7 montre que ces optimisations se traduisent par une réduction de 37 % du nombre de portes, comparées à la mise en cascade de quatre composants *lifting* indépendants. Le gain de temps de traversée dans le cas du composant *lifting step*, n'est pas très significatif. Autre solution envisageable (figure11d) : les deux multiplieurs de *scaling* sont également insérés dans le composant combinatoire.

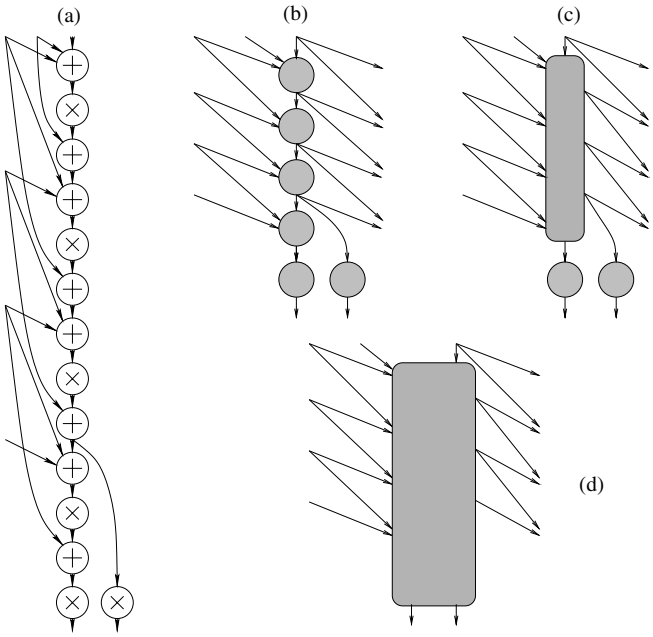


*Figure 11. Utilisation de composants combinatoires dédiés pour la décomposition en ondelettes 9/7*

### 5.2.4 Contraintes d'entrée/sortie et d'accès mémoire

Il est nécessaire de fournir à l'intégrateur du composant virtuel un moyen de limiter le parallélisme d'accès aux données en fonction des contraintes d'implantation. Jusqu'à présent, nous avons considéré que chaque itération du motif de calcul consomme et produit des blocs de pixels. Afin de limiter le parallélisme d'entrée/sortie, nous décomposons chaque bloc de pixels en *paquets* de taille fixe. Un paquet représente un ensemble de données lues ou écrites simultanément sur les ports du composant. Des paquets distincts de données d'entrée (*resp.* de sortie) sont nécessairement lus (*resp*. écrits) sur des fronts d'horloge différents.

*Tableau 7. Composants combinatoires dédiés et optimisations logiques*

| | Un composant «Lifting Step» | Quatre composants «Lifting Step» (non optimisés) | Quatre composants «Lifting Step» (optimisés) | Amélioration des performances |
|---|---|---|---|---|
| Temps de traversée (ns) | 10,5 | 42 | 40 | 5 % |
| Surface estimée (portes) | 4030 | 16120 | 10200 | 37 % |

Cette décomposition nécessite la définition de plusieurs paramètres : (1) la quantité de données $P_i$ et $P_o$ contenues respectivement dans un paquet de pixels d'entrée ou de sortie ; (2) la répartition des pixels d'un bloc dans les différents paquets ; (3) la date de lecture/écriture des paquets à l'intérieur d'une itération du motif. Afin de ne pas alourdir le jeu de paramètres du composant, nous avons proposé une répartition des dates d'entrée/sortie fondée sur des intervalles de temps fixes $\Delta T_i$ et $\Delta T_o$ entre deux paquets successifs de pixels d'entrée et de sortie, respectivement.
Les accès à la mémoire de données *contexte vertical* sont traités d'une manière similaire aux entrées/sorties. Il s'agit essentiellement de paramétrer le parallélisme d'accès à la mémoire en fonction de la quantité de données de contexte vertical consommées et produites à chaque itération. La répartition des données de contexte vertical en paquets permet d'une part de dimensionner la largeur du bus d'accès à la mémoire qui les contient, et d'autre part de définir leur placement : ce dernier est en effet fonction de la position d'une donnée à l'intérieur d'un paquet et de la date à laquelle ce paquet est accédé au cours de la transformation.
Les paquets étant accédés dans un ordre fixe, et une seule fois au cours de la transformation d'une ligne de pixels, la génération des adresses vers la mémoire de contexte verticale s'avère en fait assez systématique. Dans notre cas, l'utilisation d'un outil de synthèse de haut niveau permet de décharger le concepteur de ce travail.

#### 5.2.5 Résumé

Dans cette section, nous avons présenté les étapes successives de la conception d'un composant virtuel comportemental pour la transformation en ondelette *lifting* bidimensionnelle. À la différence de la description algorithmique qui nous a servi de point de départ (listing 1), la description comportementale obtenue intègre des informations relatives à un ensemble de décisions d'implantation. Ces décisions portent sur : (1) le choix des formats de données (qui n'a pas été développé ici); (2) le choix du motif répétitif de calcul ; (3) le choix de la granularité des opérateurs arithmétiques en bibliothèque ; (4) les contraintes de temps et d'entrée/sortie.
Le composant virtuel obtenu se caractérise par sa flexibilité, permettant à l'intégrateur d'adapter les propriétés fonctionnelles, temporelles et matérielles à ses besoins. Dans le paragraphe suivant nous présentons des résultats de synthèse d'une variété de solutions architecturales obtenues en faisant varier certains des paramètres du composant.

### 5.3 Synthèse du composant virtuel comportemental

#### 5.2.2 Instanciation

Nous nous plaçons dans un cas d'application typique utilisant l'ondelette 9/7 recommandée par le standard *JPEG2000* ($S = 4$) et réalisant la décomposition sur $N = 3$ niveaux. Dans les exemples de synthèse effectués, la largeur maximale d'une image est fixée à 512 pixels. La hauteur d'une image est illimitée.
Afin de limiter la complexité de la description, nous avons fixé la taille du motif à $L_{motif} = H_{motif} = 8$ ($w = h = 1$) et nous avons réduit la granularité des opérateurs en regroupant quatre nœuds *lifting* et *dual lifting* dans une même fonction (figure 11c) que nous avons décrite au niveau transfert de registres et synthétisée séparément sous la forme d'un composant combinatoire. Les multiplications de *scaling* restent isolés afin de laisser à l'outil de synthèse une marge de manœuvre pour les ordonnancer.
Les pixels d'entrée sont représentés sur $N_i = 8$ bits non signés et les pixels de sortie sur $N_o = 16$ bits signés dont $F_o = 2$ bits de partie fractionnaire. Les résultats intermédiaires de calcul sont représentés sur $N_d = 21$ bits pour une dynamique maximale de $N_r = 32$ bits en sortie des opérateurs internes à un nœud *lifting*.
L'ordre de parcours des blocs de pixels dans l'image suit un schéma *raster*. La mémoire nécessaire au stockage des données *contexte horizontal* est implantée sous forme de registres dont l'utilisation est entièrement gérée par l'outil de synthèse de haut niveau. Les données *contexte vertical* sont mémorisées dans une *RAM* externe. L'ordre d'entrée des pixels à l'intérieur d'un bloc de taille $8 \times 8$ suit la courbe fractale de MORTON afin de minimiser la durée de vie des données internes au motif. L'intervalle de temps entre deux paquets de pixels d'entrée/sortie est fixé à $\Delta T_i = \Delta T_o = 1$.

L'exploration architecturale a été réalisée en faisant varier la valeur des paramètres suivants :

- Parallélisme d'entrée/sortie : nous avons considéré un ensemble de cas dans lesquels le parallélisme d'accès aux données est le même en entrée et en sortie. Le nombre de bits de représentation des pixels transformés étant double de celui des pixels d'entrée, les solutions proposées définissent le nombre de pixels $P_o$ par paquet de sortie comme égal à la moitié du nombre de pixels $P_i$ par paquet d'entrée (à l'exception du cas $P_i = P_o = 64$ également traité).
Les couples $(P_i; P_o)$ testés sont : $(64; 64)$, $(64; 32)$, $(32; 16)$, $(16; 8)$, $(8; 4)$ et$(4; 2)$.

- Cadence de traitement : la durée d'exécution d'une itération du motif varie de 18 à 141 cycles, soit une cadence moyenne de traitement allant de 3,5 à 0,45 pixels par cycle.

#### 5.3.2 Résultats de synthèse

La synthèse a été réalisée à l'aide de l'outil *Monet* de *Mentor Graphics* (version *v8.7_2.1*, *release R44*) pour la technologie *AMS* $0{,}35\mu m$. La période d'horloge utilisée est de $50ns$, ce qui correspond à la latence d'un composant *chaîne lifting* (figure 11c), estimée à $40ns$ après synthèse du composant combinatoi-

re sous *Design Compiler* de *Synopsys* et augmentée de 10*ns* pour la prise en compte du délai des couches combinatoires et des interconnexions entre opérateurs et registres du chemin de données.
Notre description comportementale étant exacte au cycle près, l'ordonnancement sous *Monet* a été effectué en mode *cycle-fixed*.

**Espace des solution architecturales** – Les paramètres présentés ci-dessus nous ont permis d'obtenir, à partir d'une même description comportementale de la transformation en ondelettes 2D, 48 solutions architecturales présentant différentes performances vitesse/surface. Cet espace de solutions est représenté en figure12.

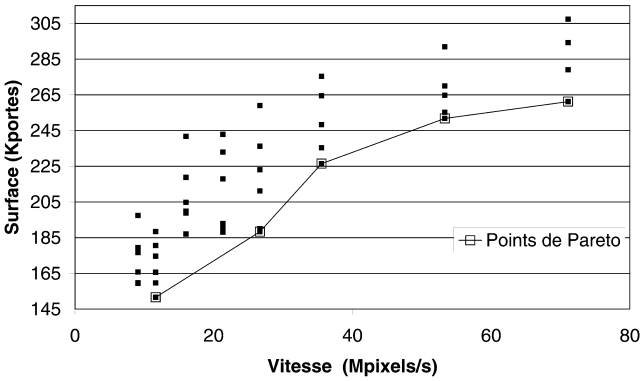


*Figure 12. Espace des solutions architecturales sous contraintes de vitesse de traitement et de surface*

Pour une période d'horloge de 50*ns*, les cadences de traitement varient de 9 Mpixels/s (0,45 pixels/cycle) à 71 Mpixels/s (3,5 pixels/cycle). La surface de l'architecture varie de 151 000 à 307 000 portes.
La courbe représentée en figure 12 contient les *points* de *Pareto* de l'espace des solutions proposé. Ces points correspondent aux solutions réalisant la meilleure performance en vitesse pour une surface donnée, et présentant la plus petite surface pour une vitesse donnée. Les cinq points obtenus correspondent aux cinq solutions *pertinentes* si les seuls critères de sélection sont la vitesse et la surface. La moins coûteuse en surface (151 Kportes) correspond à une vitesse de 11,6 Mpixels/s tandis que la plus rapide (71 Mpixels/s) présente une surface de 261 Kportes.

**Vitesse de traitement et allocation des ressources de calcul** – Le nombre d'opérateurs alloués est quasiment indépendant du parallélisme d'entrée/sortie. En revanche, la contrainte de cadence de traitement a une forte influence sur le parallélisme de calcul à mettre en œuvre. La figure 13 donne l'évolution du nombre d'opérateurs *chaîne lifting* et du nombre de multiplieurs de *scaling* en fonction de la vitesse. Nous nous plaçons ici dans le cas d'un parallélisme d'entrée/sortie maximal ($P_i = P_o = 64$) où l'ordonnancement des opérations n'est pas contraint par les entrées/sorties.
À la cadence de traitement la plus lente, nous observons que le nombre de composants *chaîne lifting* se réduit à 1, pour un nombre de multiplieurs égal à 3. Ces quantités atteignent respectivement 8 et 16 à la vitesse maximale. Dans cette dernière situation, ils représentent à eux seuls 48 % de la surface totale de l'architecture.

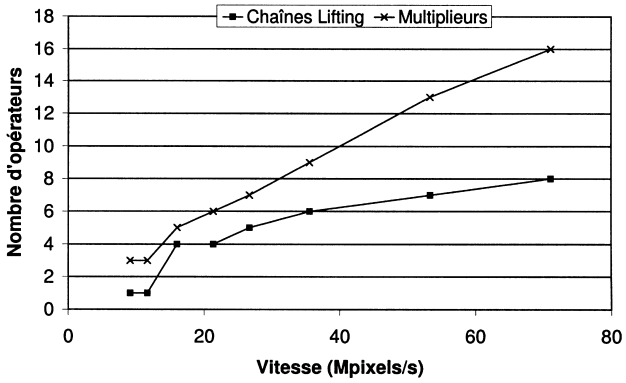


*Figure 13. Nombre d'opérateurs arithmétiques en fonction de la vitesse de traitement*

**Parallélisme d'entrée/sortie et vitesse de traitement** – La figure 14 présente la relation entre le parallélisme d'entrée/sortie appliqué et la vitesse maximale de traitement réalisable. Le parallélisme d'entrée/sortie est ici représenté par le nombre de pixels $P_o$ que le composant est capable de produire simultanément sur ses ports de sortie. À l'exception du cas $P_o = P_i = 64$, le parallélisme d'entrée $P_i$ est égal au double de $P_o$ dans les résultats présentés. Sur la figure 14, nous observons que pour $P_o < 8$ ($P_i < 16$), le nombre de cycles nécessaires à la lecture et à l'écriture des pixels sur les ports du composant limite le parallélisme de calcul exploitable, et donc la vitesse maximale de traitement. À partir de $P_o = 8$, la vitesse n'est plus limitée que par la longueur du chemin critique du motif.

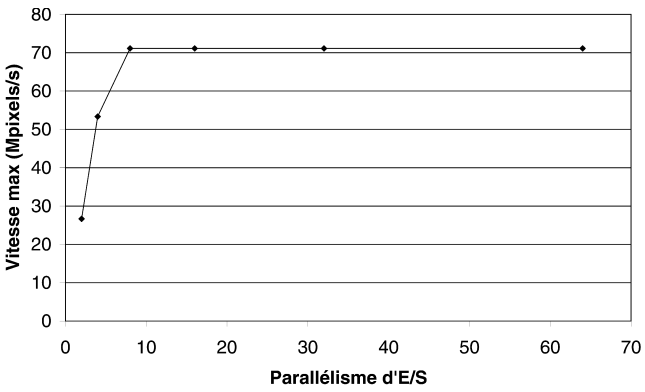


*Figure 14. Vitesse de traitement maximale autorisée en fonction du parallélisme d'entrée/sortie*

### 5.3.3 Discussion

Nous avons montré que l'utilisation de la synthèse de haut niveau autorise la génération d'une variété d'architectures *RTL* à partir d'une unique description comportementale paramétrée. L'éventail de paramètres disponibles et de performances pouvant être atteintes autorise la réutilisation de ce composant dans des applications soumises à des contraintes diverses.

À titre de comparaison des performances obtenues, une architecture de transformation *2D* de type *folded* utilisant des bancs de filtres *1D* de la forme décrite dans [35] nécessiterait au moins 12 bancs de filtres afin de réaliser en 18 cycles les 168 opérations de filtrage nécessaires à la transformation d'un bloc de $8 \times 8$ pixels sur trois niveaux. La surface estimée d'un tel banc de filtre est de 14 000 portes [35], d'où une surface totale de 168 000 portes pour les 12 bancs de filtres.

À ces 12 bancs de filtres, il convient d'adjoindre un réseau d'interconnexion et de mémorisation des données consommées et produites à chaque étape de la transformation. Les ressources de mémorisation couvrent une surface significative : dans les solutions que nous avons obtenues cette surface représente 50 000 à 90 000 portes. La surface de l'architecture *folded* complète atteint ainsi aisément 200 000 à 250 000 portes, sans considération du contrôle nécessaire, c'est-à-dire une surface du même ordre de grandeur que les solutions obtenues en synthèse de haut niveau.

Par ailleurs, notons que la majorité des architectures recensées dans la littérature ne traitent pas le problème de l'extension de l'image par symétrie périodique au voisinage des bords. Comme le montre le listing 1 en page 13, l'implantation d'un tel mécanisme dans une description destinée à la synthèse de haut niveau est très naturelle et l'effort de conception supplémentaire est négligeable.

Les résultats que nous présentons sont cependant loin d'épuiser les possibilités qu'offrent les paramètres supportés par notre composant virtuel de transformation en ondelettes *2D*. Notamment, il aurait été intéressant de faire varier la taille du motif en même temps que la contrainte de temps afin de tester un jeu de solutions à parallélisme moyen constant. Nous nous heurtons ici aux limitations, en termes de complexité calculatoire des descriptions, de l'outil de synthèse de haut niveau *Monet*, limitations que nous croyons essentiellement dues au manque de maturité de l'outil.

# 6. Conclusion et perspectives

Dans cet article, nous avons mis en application une démarche méthodologique permettant d'insérer les outils de synthèse de haut niveau dans un flot de réutilisation de composants virtuels. Cette démarche définit la notion de *composant virtuel comportemental* comme un niveau d'abstraction supplémentaire – venant s'ajouter aux niveaux *hard*, *firm* et *soft* définis par l'organisation *VSIA* – à partir duquel un composant *IP* peut être délivré et synthétisé.

La méthodologie et le flot de conception proposés établissent un lien fort entre les nouvelles approches de conception au niveau système et la synthèse de haut niveau. Ils permettent ainsi d'insérer les outils de synthèse de haut niveau dans un flot cohérent de *synthèse système* à base de composants réutilisables. Du fait du haut degré de flexibilité d'un composant virtuel comportemental, l'utilisateur d'un tel composant ne se voit plus délivrer une unique architecture, mais dispose au contraire d'un ensemble de solutions qu'il lui sera loisible d'explorer au moyen du jeu de paramètres associé au composant.

Cette méthodologie a pu être expérimentée sur la réalisation d'un composant de transformation en ondelettes discrète compatible avec la norme *JPEG2000* pour le codage des images fixes. Cette étude a été conduite en collaboration avec le *CNES* et la société *Astrium* dans la perspective d'applications d'imagerie spatiale. L'algorithme fondé sur le *lifting scheme* nous a permis de mettre en œuvre une exploration systématique des possibilités de répartition des calculs et des entrées/sorties au cours du temps, au moyen d'une modélisation à base de graphes de dépendances. Les résultats de synthèse de haut niveau présentés montrent le haut degré de flexibilité autorisé par la démarche de conception suivie.

La variété de solutions architecturales obtenues pour une application complexe comme la transformation en ondelettes bidimensionnelle illustre l'aptitude des outils de synthèse de haut niveau à exploiter des choix d'implantation originaux auxquels un concepteur humain ne se risquerait pas nécessairement. Dans le cas de la transformation en ondelettes multi-niveaux, nous avons montré que les précédentes tentatives d'implantation considéraient le banc de filtres comme la brique de base de l'architecture, celle-ci étant conçue soit comme la mise en cascade de bancs identiques, soit comme l'association d'un banc de filtres avec une architecture de contrôle et de mémorisation. L'utilisation de la synthèse de haut niveau nous a permis d'obtenir un ensemble de solutions dans lesquelles le calcul de transformation sur plusieurs niveaux est vu au contraire comme un unique flot de données auquel un ensemble de ressources de calcul – limité par le parallélisme maximal souhaité par l'utilisateur – est affecté. Les possibilités d'optimisation exploitées par un outil de synthèse de haut niveau ne sont par conséquent plus locales à une passe de filtrage 1D sur un niveau, mais globales à la transformation complète d'un bloc de pixels. Malgré le manque de maturité des outils commerciaux existants – qui sont pour la plupart encore dans leur phase de développement et dont la maîtrise exige encore un savoir-faire significatif de la part du concepteur – les techniques de synthèse de haut niveau devraient jouer un rôle de premier plan dans les futures méthodologies de synthèse système.

# Références


[1] M. D. Adams et R. Ward, Wavelet Transforms in the JPEG-2000 standard. Dans *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, volume 1, pp. 160-163, août 2001.

[2] R. Airiau, A. Carer, E. Casseau, E. Martin et O. Sentieys : Méthodologie de Conception de Composants Virtuels pour les Applications de TDSI. Dans *Actes Conférence Adéquation Algorithme Architecture (AAA)*, janvier 2000.

[3] K. Andra, C. Chakrabarti et T. Acharya, A VLSI Architecture for Lifting-Based Wavelet Transform. Dans *Proc. IEEE Workshop on Signal Processing Systems (SiPS) Design and Implementation*, pp. 70-79, octobre 2000.

[4] M. Antonini, M. Barlaud, P. Mathieu et I. Daubechies, Image Coding using Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2) :205-220, avril 1992.

[5] J.-Y. Brunel *et al.*, COSY Communication IPs. Dans *Proc. Design Automation Conference (DAC)*, pp. 406-410, 2000.

[6] W. O. Cesario, Z. Sugar, I. Moussa et A. A. Jerraya, Efficient Integration of Behavioral Synthesis within Existing Design Flows. Dans *Proc. 13th International Symposium on System Synthesis (ISSS)*, pp. 85-90, septembre 2000.

[7] C. Chakrabarti, M. Vishwanath et R. M. Owens, Architectures for Wavelet Transforms: A Survey. *Journal of VLSI Signal Processing, Image and Video Technology*, 14(2) :171-192, novembre 1996.

[8] H. Chang *et al.*, *Surviving the SoC Revolution - A Practical Guide toPlatform-Based Design*. Kluwer Academic Publishers, novembre 1999.

[9] A. Cohen, I. Daubechies et J. Feauveau, Bi-orthogonal Bases of Compactly Supported Wavelets. *Communications on Pure and Applied Mathematics* : 45, 485-560, 1992.

[10] P. Coussy, A. Baganne et E.Martin, A Design Methodology for IP Integration. Dans *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, mai 2002.

[11] I. Daubechies et W. Sweldens, Factoring Wavelet Transforms into Lifting Steps. *Journal of Fourier Analysis and Applications*, 4(3) : 247-269, 1998.

[12] C. Diou, L. Torres et M. Robert, VLSI implementation of Lifting Scheme Wavelet Transform. Dans *Proc. Design Automation and Test in Europe (DATE) conference, Designers' forum*, pp. 67-72, mars 2001.

[13] J. P. Elliott, *Understanding Behavioral Synthesis. A Practical Guide to High-Level Design*. Kluwer Academic Publishers, 2000.

[14] N. Fan, V. Chaiyakul et D. Gajski, Usage-Based Characterization of Complex Functional Blocks for Reuse in Behavioral Synthesis. Dans *Proc. Asian and South Pacific Design Automation Conference (ASP-DAC)*, janvier 2000.

[15] F. Franssen, F. Balasa, M. van Swaaij, F. Catthoor et H. De Man, Modeling Multidimensional Data and Control Flow. *IEEE Transactions on VLSi Systems*, 1(3), septembre 1993.

[16] D. Gajski *et al.*, *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1991.

[17] C. Gasquet et P. Witomski, *Analyse de Fourier et Applications*. Masson, 1995.

[18] D. Hommais, F. Pétrot et I. Augé, A Practical Toolbox for System Level Communication Synthesis. Dans *Proc. IEEE International Workshop on Hardware/Software Co-Design (CODES)*, pp. 48-53, 2001.

[19] ISO/IEC JTC1/SC29, Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines. Rapport technique 10918-1:1994, 1994.

[20] ISO/IEC JTC1/SC29, Information technology – JPEG 2000 Requirements and Profiles Version 6.3. Rapport technique, juillet 2000.

[21] ISO/IEC JTC1/SC29, Information technology – JPEG 2000 Part I Final Committee Draft Version 1.0. Rapport technique FCD15444-1, mars 2001.

[22] ITRS, International Technology Roadmap for Semiconductors. Rapport technique, 2001.

[23] A.-A. Jerraya, N.-E. Zergainoh, W. Cesario, F. Rousseau, D. Borrione, F. Hessel et E. Martin, *Conception de haut niveau des systèmes monopuces*. Traité EGEM, Série électronique et micro-électronique. Hermes Science, mai 2002.

[24] M. Keating et P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Design*. Kluwer Academic Publishers, 1999.

[25] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey et A. Sangiovanni-Vincentelli : System-Level Design: Orthogonalization of Concerns and Platform-Based Design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(12), décembre 2000.

[26] D. W. Knapp, *Behavioral Synthesis. Digital System Design Using the Synopsys Behavioral Compiler*. Prentice Hall, 1996.

[27] G. Lafruit, F. Catthoor, J. Cornelis et H. De Man, An Efficient VLSI architecture for 2-D Wavelet Image Coding with Novel Image Scan. *IEEE Transactions on Very Large Scale Integration Systems*, 7(1):56-68, mars 1999.

[28] C. Lambert-Nebout, G. Moury et J.-E. Blamont, Status of Onboard Image Compression for CNES Space Missions. Dans *Proc. of SPIE'99*, volume 3808, pp. 242-256, octobre 1999.

[29] LESTER, Outil GAUT.: http://web.univ-ubs.fr/gaut.

[30] Ph. Lier, G. Moury, C. Latry et F. Cabot, Selection of the SPOT5 Image Compression Algorithm. Dans *Proc. of SPIE'98*, volume 3439, pp. 541-552, octobre 1998.

[31] S.G. Mallat, A Theory of Multi-Resolution Signal Decomposition: the Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7) : 674-693, juillet 1989.

[32] H. De Man, J. Rabaey, P. Six et L. Claesen, Cathedral-II: A silicon compiler for digital signal processing. *IEEE Des. Test*, 3(6) : 13-125, décembre 1986.

[33] E. Martin, C. Nouet et J.-M. Toureilles, Conception Optimisée d'Architectures en Précision Finie pour les Applications de Traitement du Signal. *Traitement du Signal*, 18(1), 2001.

[34] E. Martin, O. Sentieys, H. Dubois et J.-L. Philippe, GAUT: An Architectural Synthesis Tool for Dedicated Signal Processors. Dans *Proc. European Design Automation Conference (EuroDAC)*, pp. 14-19, septembre 1993.

[35] S. Masud et J. V. McCanny, Design of Silicon IP Cores for Biorthogonal Wavelet Transforms. *Journal of VLSI Signal Processing*, 29 : 179-196, 2001.

[36] A. Melikian, D. Altas et G. Fayad, A High Level Synthesis Approach to Soft IP Reuse. Dans *Proc. Synopsys Users Group Forum (SNUG)*, 1999.

[37] Gordon E. Moore, Cramming More Components onto Integrated Circuits. *Electronics*, avril 1965.

[38] L. M. Reyneri, F. Cucinotta, A. Serra et L. Lavagno, A Hardware/Software Co-design Flow and IP Library Based on Simulink. Dans *Proc. Design Automation Conference (DAC)*, pp. 593-598, juin 2001.

[39] RNRT, LESTER, LASTI et France Telecom R&D, Site Internet du Projet MILPAT. http://web.univ-ubs.fr/lester/milpat.

[40] G. Savaton, E. Casseau, E. Martin et C. Lambert-Nebout, Behavioral Virtual Components for Embedded Image Compression Systems. Dans *Proc. 17th Conference on Digital Circuits and Integrated Systems (DCIS)*, novembre 2002.

[41] G. Strang et T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[42] A. Takach, P. Gutberlet et S. Waters, Hardware Design using Algorithmic C. Dans *Proc. Forum on Design Languages (FDL)*, septembre 2001.

[43] A. Vermeulen, F. Catthoor, D. Verkest et H. De Man, Formalized Three-Layer System-Level Reuse Model and Methodology for Embedded Data-Dominated Applications. Dans *Proc. Design Automation and Test in Europe (DATE)*, mars 2000.

[44] VSI Alliance, Site Internet. http://www.vsi.org.

[45] VSI Alliance, Architecture Document – Version 1.0. Rapport technique, 1997.

[46] D. K. Wilde, The Alpha Language. Rapport technique, IRISA, mai 1994. Publication interne 827.

[47] D. K. Wilde et Oumarou Sié, Regular Array Synthesis using Alpha. Rapport technique, IRISA, mai 1994. Publication interne 829.

[48] S. Yoo *et al.*, A Generic Wrapper Architecture for Multi-Processor SoC Cosimulation and Design. Dans *Proc. IEEE International Workshop on Hardware/Software Co-Design (CODES)*, pp. 195-200, 2001.

Guillaume **Savaton**

Guillaume Savaton est enseignant-chercheur à l'ESEO (École Supérieure d'Électronique de l'Ouest, Angers). Titulaire du diplôme d'ingénieur ESEO, il a obtenu le titre de docteur de l'Université de Bretagne Sud dans la spécialité *Sciences pour l'Ingénieur* en 2002. Ses travaux de recherches au sein de l'équipe *Composants Virtuels* du LESTER l'ont amené à s'intéresser aux méthodes de conception de composants matériels réutilisables pour le traitement du signal et de l'image. Ses activités portent aujourd'hui sur les méthodes de conception et de vérification de systèmes embarqués temps réel.

Emmanuel **Casseau**

Emmanuel Casseau est Maître de Conférences à l'Université de Bretagne Sud (UBS). Il a obtenu l'Habilitation à Diriger des Recherches en 2002. Il est responsable du projet *Composants Virtuels* du laboratoire LESTER (Laboratoire d'électronique des Systèmes Temps Réels). Ses travaux de recherche concernent les méthodes de synthèse de haut niveau pour la conception de systèmes sur puce.

Eric **Martin**

Eric Martin est professeur agrégé de l'ENS de Cachan, Professeur des Universités et directeur du LESTER à l'Université de Bretagne Sud. Son domaine d'intérêt concerne l'adéquation algorithme architecture en traitement du signal et des images, et les méthodes de conception de haut niveau des circuits dédiés. Il pilote depuis 1988 le développement du projet GAUT.