Extrait ANNALES DES TELECOMMUNICATIONS, tome 40, n[os] 9-10
Septembre - Octobre 1985

Algorithmes de transformations de Fourier et en cosinus mono et bidimensionnels

Martin VETTERLI *

Henri J. NUSSBAUMER *

Analyse

Cet article présente un algorithme rapide pour le calcul de la transformée de Fourier discrète et de la transformée en cosinus discrète, et ceci lorsque la longueur de la transformation est une puissance de 2. Il atteint le nombre minimal connu d'opérations (multiplications et additions) pour la transformation de Fourier discrète de séquences réelles, complexes, symétriques et antisymétriques, pour la transformation de Fourier discrète impaire ainsi que pour la transformation en cosinus discrète. L'extension au cas bi-dimensionnel des transformations de Fourier et cosinus discrètes est également présentée.

Mots clés : Algorithme, Transformation Fourier discrète, Transformation cosinus, Complexité algorithme, Symétrie, Signal discret, Signal bidimensionnel, Mise en œuvre.

ALGORITHMS FOR THE MONO- AND BI-DIMENSIONAL FOURIER AND COSINE TRANSFORMS

Abstract

This paper presents a fast algorithm for the computation of the discrete Fourier and cosine transform, and this for transform lengths which are powers of 2. This approach achieves the lowest known number of operations (multiplications and additions) for the discrete Fourier transform of real, complex, symmetrical and antisymmetrical sequences, for the odd discrete Fourier transform and for the discrete cosine transform. The extension to the two-dimensional Fourier and cosine transform is presented as well.

Key words : Algorithm, Discrete Fourier transformation, Cosinus transformation, Algorithm complexity, Symmetry, Discrete signal, Two dimensional signal, Implementation.

Sommaire

I. INTRODUCTION

Depuis la publication de l'algorithme FFT de transformation de Fourier rapide par Cooley et Tukey [1], la recherche sur les algorithmes de ce type s'est concentrée sur 2 types distincts de transformations : d'une part, les algorithmes *classiques* pour des longueurs qui sont des puissances de 2 (ou d'autres nombres *petits*) ou des produits de ceux-ci ont été améliorés [2-5] et d'autre part, pour des longueurs ayant des facteurs premiers entre eux, de nouveaux algorithmes rapides ont été développés, en particulier celui de Winograd [6] ainsi que celui de PFA [7].

Récemment, les algorithmes correspondant aux longueurs en puissances de 2 ont connu un regain d'intérêt, surtout en raison de leur structure plus simple que celle des algorithmes basés sur une décomposition en facteurs premiers (en particulier, l'algorithme de Winograd n'a pas apporté jusqu'ici les améliorations espérées une fois qu'il fut implanté [8]). Dans cette optique, plusieurs algorithmes ont été proposés récemment [9-11], et ils atteignent tous un même nombre d'opérations minimal connu [2].

L'algorithme présenté dans cet article [10] résout également le problème de la transformation en cosinus

* Ecole polytechnique fédérale de Lausanne, 16, chemin de Bellerive, CH-1007 Lausanne, Suisse.

discrète, dont l'utilisation est fréquente en codage de parole et d'image [12]. De surcroît, l'application aux séquences avec symétries est immédiate, ainsi que celle à la transformée de Fourier impaire. Comme l'algorithme est réel, il est particulièrement adapté à la transformation de séquences réelles ou hermitiennes, donc à la convolution circulaire par transformation de séquences réelles. L'extension aux transformations de Fourier et cosinus bi-dimensionnelles est également proposée, compte tenu de leur importance dans le traitement d'image [12].

L'algorithme proposé a été mis en œuvre par un programme d'ordinateur [13] en vue d'une implantation sur processeur de signal, et un circuit intégré réalisant une transformation en cosinus de 8 points à 120 Mbit/s a été conçu [14]. Ces deux applications seront également décrites brièvement.

II. L'ALGORITHME DE TRANSFORMATION DE FOURIER - COSINUS RAPIDE

« ... diviser chacune des difficultés à examiner en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre. »

R. Descartes, « Le discours de la méthode »

Introduisons les définitions suivantes, où x(n) est un élément d'un vecteur réel de longueur N (N étant une puissance de 2) avec $n = 0, ..., N-1$:

— la transformation de Fourier discrète (« discrete Fourier transform », DFT) est définie par :

$$(1)\quad \text{DFT}(k, N, \boldsymbol{x}) := \sum_{n=0}^{N-1} x(n)\, \mathrm{e}^{-\mathrm{j}2\pi nk/N}, \qquad k = 0, ..., N-1, \quad \mathrm{j} = \sqrt{-1},$$

— la transformation en cosinus discrète (« discrete cosine transform », DCT) :

$$(2)\quad \text{DCT}(k, N, \boldsymbol{x}) := \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi(2n+1)k}{4N}\right), \qquad k = 0, ..., N-1,$$

— la transformation de Fourier-cosinus (« discrete cosine Fourier transform », cos-DFT) :

$$(3)\quad \text{cos-DFT}(k, N, \boldsymbol{x}) := \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi nk}{N}\right), \qquad k = 0, ..., N-1,$$

— la transformation de Fourier-sinus (« discrete sine Fourier transform », sin-DFT) :

$$(4)\quad \text{sin-DFT}(k, N, \boldsymbol{x}) := \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi nk}{N}\right), \qquad k = 0, ..., N-1.$$

Les relations (3) et (4) correspondent ici simplement aux parties réelles et imaginaires (avec un signe moins) de la transformation de Fourier définie en (1). D'autre part, et afin de simplifier la présentation, le facteur multiplicatif $1/\sqrt{2}$ pour $k = 0$ a été omis dans la définition (2) de la transformation en cosinus discrète [15].

L'algorithme présenté ci-dessous suit une approche par réductions successives où chaque transformée est systématiquement décomposée en une transformée de longueur moitié, plus deux transformées de longueur un quart. Cette méthode est similaire à celle utilisée dans [9], et nous la présenterons ici brièvement en nous référant à [10] pour plus de détails.

Pour évaluer une DFT, on considère séparément le calcul de sa partie réelle et de sa partie imaginaire. La première étant une cos-DFT, et la fonction cosinus étant paire, on peut récrire la cos-DFT de longueur N comme suit :

$$(5)\quad \text{cos-DFT}(k, N, \boldsymbol{x}) = \sum_{n=0}^{N/2-1} \text{x}(2n) \cos\left(\frac{2\pi nk}{N/2}\right) + \sum_{n=0}^{N/4-1} (x(2n+1) + \text{x}(N-2n-1)) \cos\left(\frac{2\pi(2n+1)k}{4\,N/4}\right),$$

ou, plus succinctement :

$$\text{cos-DFT}(k, N, \boldsymbol{x}) = \text{cos-DFT}(k, N/2, \boldsymbol{x}_1) + \text{DCT}(k, N/4, \boldsymbol{x}_2), \qquad k = 0, ..., N-1,$$

avec :

$$(6)\quad x_1(n) = x(2n), \qquad n = 0, ..., (N/2)-1,$$

$$x_2(n) = x(2n+1) + x(N-2n-1), \qquad n = 0, ..., (N/4)-1.$$

Dans (6), on tiendra compte du fait que cos-DFT($N-k$, N, $\boldsymbol{x}$) = cos-DFT(k, N, $\boldsymbol{x}$) et DCT($2N-k$, N, $\boldsymbol{x}$) = – DCT(k, N, $\boldsymbol{x}$) lorsque cela est nécessaire.

La relation (5) permet de réduire une cos-DFT de longueur N en une cos-DFT de longueur $N/2$ et une DCT de longueur $N/4$, au prix de 3 $N/4$ additions. L'évaluation de la cos-DFT de longueur moitié peut être traitée par la même méthode, et le calcul de la DCT sera considéré ultérieurement.

Le calcul de la partie imaginaire de la DFT se réduit à celui d'une simple sin-DFT, et il peut être organisé de la façon suivante en tirant parti du fait que la fonction sinus est impaire :

$$(7)\quad \text{sin-DFT}(k, N, \boldsymbol{x}) = \sum_{n=0}^{N/2-1} x(2n) \sin\left(\frac{2\pi nk}{N/2}\right) + \sum_{n=0}^{N/4-1} (x(2n+1) - x(N-2n-1)) \sin\left(\frac{2\pi(2n+1)k}{4\,N/4}\right).$$

En notant que :

$$(8)\quad \sin\left(\frac{2\pi(2n+1)k}{N}\right) = (-1)^n \cos\left(\frac{2\pi(2n+1)(N/4-k)}{N}\right),$$

la relation (7) devient :

$$(9)\quad \text{sin-DFT}(k,\ N,\ \mathbf{x}) = \text{sin-DFT}(k,\ N/2,\ \mathbf{x}_1) + \text{DCT}(N/4 - k,\ N/4,\ \mathbf{x}_3),\quad k = 0, \ldots, N-1,$$

avec :

$$x_3(n) = (-1)^n\,(x(2n+1) - x(N-2n-1)).$$

Dans (9), on tiendra compte au besoin du fait que $\text{sin-DFT}(k,\ N,\ \mathbf{x}) = -\text{sin-DFT}(N-k,\ N,\ \mathbf{x})$ et $\text{DCT}(2N-k),\ N,\ \mathbf{x}) = -\text{DCT}(k,\ N,\ \mathbf{x})$.

On voit donc que le calcul par la relation (9) permet de réduire une sin-DFT de longueur N en une sin-DFT de longueur $N/2$ et une DCT de longueur $N/4$ au prix de $3N/4 - 2$ additions. Comme résultat intermédiaire, nous notons qu'une DFT de longueur N se réduit à l'évaluation d'une DFT de longueur $N/2$ et de deux DCT de longueur $N/4$, plus $3N/2 - 2$ additions auxiliaires (Fig. 1).

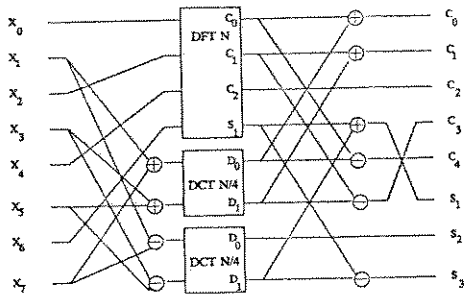

FIG. 1. — Evaluation d'une DFT de longueur N avec une DFT de $N/2$ et de deux DCT de $N/4$ (cas $N = 8$).

Evaluation of a length-N DFT *with a* DFT *of* $N/2$ *and two* DCT*'s of* $N/4$ *(case* $N = 8$*).*

Considérons maintenant l'évaluation de la transformation en cosinus discrète (DCT). La DCT peut être ramenée à une DFT en réarrangeant la séquence d'entrée selon la méthode proposée en [16], avec :

$$(10)\quad x_4(n) = x(2n),$$
$$x_4(N-n-1) = x(2n+1),\ n = 0, \ldots, N/2-1,$$

et la DCT devient :

$$(11)\quad \text{DCT}(k,\ N,\ \mathbf{x}) = \sum_{n=0}^{N-1} x_4(n) \cos\left(\frac{2\pi(4n+1)k}{4N}\right),\quad k = 0, \ldots, N-1,$$
$$= \cos\left(\frac{2\pi k}{4N}\right) \text{cos-DFT}(k,\ N,\ \mathbf{x}_4) - \sin\left(\frac{2\pi k}{4N}\right) \text{sin-DFT}(k,\ N,\ \mathbf{x}_4),\quad k = 0, \ldots, N-1.$$

En utilisant les symétries des fonctions trigonométriques, la relation (11) devient :

(12)

$$\begin{bmatrix} \text{DCT}(k, N, \mathbf{x}) \\ \text{DCT}(N-k, N, \mathbf{x}) \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{2\pi k}{4N}\right) & -\sin\left(\frac{2\pi k}{4N}\right) \\ \sin\left(\frac{2\pi k}{4N}\right) & \cos\left(\frac{2\pi k}{4N}\right) \end{bmatrix} \times \begin{bmatrix} \text{cos-DFT}(k,\ N,\ \mathbf{x}_4) \\ \text{sin-DFT}(k,\ N,\ \mathbf{x}_4) \end{bmatrix},\quad k = 0, \ldots, N/2-1,$$

$$\text{DCT}(N/2,\ N,\ \mathbf{x}) = \cos(\pi/4)\ \text{cos-DFT}(N/2,\ N,\ \mathbf{x}_4).$$

Comme le produit matriciel (12) correspond à une rotation, il peut être évalué au moyen de 3 multiplications et de 3 additions [10, 17].

En utilisant toutes ces simplifications, l'évaluation d'une DCT avec une DFT réelle de même dimension par (12) requiert $3N/2 - 2$ multiplications et $3N/2 - 3$ additions. Schématiquement, le procédé est esquissé sur la figure 2. L'algorithme se poursuit ensuite de façon répétitive par décomposition de la DFT de N en une DFT de $N/2$ et deux DCT de $N/4$.

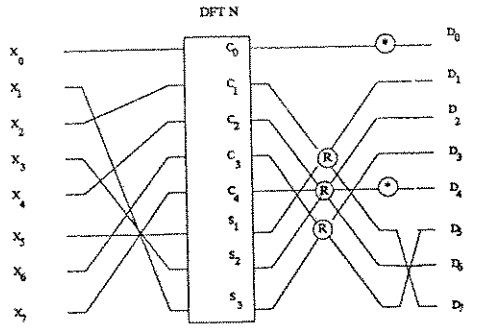

FIG. 2. — Evaluation d'une DCT de longueur N à l'aide d'une permutation, d'une DFT de N et de rotations de sortie (cas $N=8$).

* : $1/\sqrt{2}$,

R : rotation.

Evaluation of a length-N DCT *with a permutation, a* DFT *of* N *and output rotations (case* $N = 8$*).*

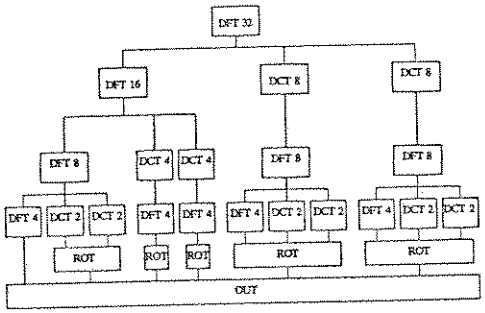

FIG. 3. — Evaluation d'une DFT de longueur 32 par des DFT et DCT de longueurs réduites.

Evaluation of a DFT *of length 32 with* DFT*'s and* DCT*'s of reduced length.*

Afin d'illustrer l'emploi récursif de cette méthode, l'exemple d'une transformation de Fourier discrète de 32 points est présenté sur la figure 3. La transformation est réduite en DFT et DCT de dimensions de plus en plus petites, jusqu'à ce qu'il ne reste plus

que des transformations simples (de longueur 2 ou 4). Les résultats sont alors recombinés à l'aide de rotations, de sommes et de différences afin d'obtenir le résultat final.

Il est possible d'évaluer la complexité de calcul qui résulte de cet algorithme en partant de transformations triviales ($N = 2, 4$) dont la complexité est connue, et en en dérivant la charge de calcul pour des transformations de n'importe quelle longueur. La solution des formules de récursion [10] donne les complexités suivantes (où $O_A[\ ]$ signifie nombre d'additions réelles et $O_M[\ ]$ nombre de multiplications réelles respectivement) pour la transformation de Fourier discrète d'une séquence réelle, avec $N = 2^M$:

$$(13)\quad O_M[\text{DFT}(N, \mathbf{x}_R)] = (N/2)(\log_2(N) - 3) + 2,$$
$$O_A[\text{DFT}(N, \mathbf{x}_R)] = (N/2)(3\log_2(N) - 5) + 4.$$

Pour le calcul de la DFT d'une séquence complexe, on prend séparément la transformation de la partie réelle et imaginaire, puis on les recombine (à l'aide de $2N - 4$ additions). On a ainsi pour la transformée de Fourier discrète d'une séquence complexe, avec $N = 2^M$:

$$(14)\quad O_M[\text{DFT}(N, \mathbf{x}_C)] = N(\log_2(N) - 3) + 4,$$
$$O_A[\text{DFT}(N, \mathbf{x}_C)] = 3N(\log_2(N) - 1) + 4.$$

Finalement, pour la transformée en cosinus discrète d'une séquence réelle avec $N = 2^M$, la complexité devient :

$$(15)\quad O_M[\text{DCT}(N, \mathbf{x}_R)] = (N/2)\log_2(N),$$
$$O_A[\text{DCT}(N, \mathbf{x}_R)] = (N/2)(3\log_2(N) - 2) + 1.$$

Afin de comparer la complexité de calcul de l'algorithme FFCT que nous venons de présenter avec celle des algorithmes couramment utilisés, on peut d'abord considérer l'évaluation de la transformation de Fourier d'une séquence réelle. Avec la méthode classique, la transformée de la séquence réelle est calculée grâce à une transformée complexe de dimension moitié moyennant des pré-multiplications. En utilisant l'algorithme de Rader-Brenner [3] pour la transformation complexe, on aboutit ainsi au nombre d'opérations indiqué dans le tableau I. On voit sur cet exemple que l'algorithme FFCT réduit très sensiblement le nombre de multiplications et d'additions par rapport

TABL. I. — Comparaison de la complexité de calcul pour la transformation de Fourier discrète d'un signal réel.

Comparison of the computational complexity for the discrete Fourier transform of a real signal.

Longueur	FFT de base 2 [18]		FFCT	
N	multiplications	additions	multiplications	additions
8	10	42	2	20
16	26	122	10	60
32	66	290	34	164
64	162	710	98	420
128	386	1 678	258	1 028
256	898	3 870	642	2 436
512	2 050	8 766	1 538	5 636
1 024	4 610	19 582	3 586	12 804
2 048	10 242	43 262	8 194	28 676

TABL. II. — Comparaison de la complexité de calcul pour la transformation de Fourier discrète d'un signal complexe.

Comparison of the computational complexity for the discrete Fourier transform of a complex signal.

Longueur	FFT de Rader-Brenner [3]		FFCT	
N	multiplications	additions	multiplications	additions
8	4	52	4	52
16	20	148	20	148
32	68	424	68	388
64	196	1 104	196	964
128	516	2 720	516	2 308
256	1 284	6 464	1 284	5 380
512	3 076	14 976	3 076	12 292
1 024	7 172	34 048	7 172	27 652
2 048	16 388	76 288	16 388	61 444

TABL. III. — Comparaison de la complexité de calcul pour la transformation en cosinus discrète d'un signal réel.

Comparison of the computational complexity for the discrete cosine transform of a real signal.

Longueur	DCT de Chen *et al.* [19]		FFCT	
N	multiplications	additions	multiplications	additions
8	16	26	12	29
16	44	74	32	81
32	116	194	80	209
64	292	482	192	513
128	708	1 154	448	1 217
256	1 668	2 690	1 024	2 817
512	3 844	6 146	2 304	6 401
1 024	8 708	13 826	5 120	14 337
2 048	19 460	30 722	11 264	31 745

à l'approche classique. Dans le cas de l'évaluation de la transformation de Fourier d'une séquence complexe (Tabl. II), l'algorithme FFCT conduit exactement au même nombre de multiplications que l'algorithme classique de Rader-Brenner, avec une réduction sensible sur le nombre d'additions. Finalement, l'algorithme usuel de calcul de la transformation en cosinus discrète [19] est comparé à l'algorithme de FFCT dans le tableau III. Tout en gardant un nombre comparable d'additions, l'algorithme de FFCT réduit notablement le nombre de multiplications.

Notons que les algorithmes pris en considération dans les comparaisons étaient les algorithmes usuels à l'époque du développement de l'algorithme de FFCT. Entre-temps, plusieurs algorithmes nouveaux ont été publiés [9, 11], et ils atteignent la même complexité que celui présenté ci-dessus, sans toutefois lui être identiques.

Avec l'apparition de processeurs de traitement du signal dans lesquels les temps d'exécution des additions et des multiplications sont sensiblement les mêmes, il est très important de disposer aujourd'hui d'algorithmes ayant une structure régulière et qui minimisent le nombre total d'opérations (multiplications, additions, transferts de mémoire) plutôt que les seules multiplications. Dans ce contexte, les approches de type FFCT semblent très attrayantes bien qu'elles conduisent à un nombre de multiplications supérieur à celui auquel on aboutit avec l'algorithme de Winograd ou l'algorithme PFA.

III. APPLICATIONS AUX SIGNAUX AVEC SYMÉTRIES ET A LA TRANSFORMATION DE FOURIER IMPAIRE

L'application de l'algorithme de FFCT au cas de séquences possédant certaines symétries présente un intérêt à la fois théorique et pratique.

D'un point de vue théorique, s'il est possible de montrer que l'existence d'une symétrie du signal réduit la complexité de moitié (exactement pour le nombre de multiplications, approximativement pour les additions), on établit que l'algorithme est cohérent. En effet, si l'amélioration était plus grande, on pourrait décomposer un signal général en composantes symétriques, donc améliorer la complexité (multiplicative) du cas général. Si d'autre part l'amélioration était moindre, l'algorithme ne tirerait pas entièrement parti de la réduction de complexité. En résumé, un algorithme cohérent utilisant M multiplications pour la transformation d'une séquence complexe en utilisera $M/2$ pour une séquence réelle ou une séquence complexe symétrique/antisymétrique et $M/4$ pour une séquence réelle symétrique/antisymétrique. Pour les additions, l'ordre de grandeur sera le même, quoique des termes de corrections peuvent apparaître. Notons que ces relations ne sont pas respectées pour bon nombre d'algorithmes proposés pour le cas réel [18] pour le cas symétrique [20].

En ce qui concerne les aspects pratiques du calcul de transformées de signaux symétriques, il existe de nombreuses applications de ce type, l'exemple connu étant la transformation de Fourier inverse d'un signal hermitien (qui est la représentation fréquentielle d'un signal réel).

Nous considérerons d'abord ici le calcul de la transformée de Fourier d'un signal symétrique. La partie imaginaire de la transformée étant nulle, il suffit d'évaluer une cos-DFT de longueur N. La relation (5) devient donc :

$$(16)\quad \text{cos-DFT}(k, N, \mathbf{x}_S) = \sum_{n=0}^{N/2-1} x(2n) \cos\left(\frac{4\pi nk}{N}\right) + 2 \sum_{n=0}^{N/4-1} x(2n+1) \cos\left(\frac{2\pi(2n+1)k}{N}\right),$$

où $\mathbf{x}_S$ est une séquence telle que :

$$(17)\quad x_S(n) = x_S(N-n), \quad n = 1, \ldots, N/2-1,$$
$$x_S(0),\ x(N/2) \text{ arbitraires.}$$

En négligeant le facteur 2 qui correspond à un simple décalage, la charge du calcul requise pour évaluer (16) devient :

$$(18)\quad O_M[\text{DFT}(N, \mathbf{x}_S)] = (N/4)(\log_2(N) - 3) + 1,$$
$$O_A[\text{DFT}(N, \mathbf{x}_S)] = (N/4)(3\log_2(N) - 7) + \log_2(N) + 3.$$

Lorsque la séquence est antisymétrique, on a :

$$(19)\quad x_A(n) = -x_A(N-k), \quad n = 1, \ldots, N/2 - 1,$$
$$x_A(0) = 0, \; x_A(N/2) = 0.$$

Dans ce cas, la transformation de Fourier se réduit à une sin-DFT de longueur N, avec :

(20)

$$\text{sin-DFT}(k, N, \mathbf{x}_A) = -\mathbf{j}\sum_{n=0}^{N/2-1} x(2n)\sin\left(\frac{4\pi nk}{N}\right) - 2\mathbf{j}\sum_{n=0}^{N/4-1} x(2n+1)\sin\left(\frac{2\pi(2n+1)k}{N}\right).$$

En négligeant de nouveau le facteur 2, le nombre de multiplications est le même que dans (18), et le nombre d'additions est donné par :

$$(21)\quad O_A[\text{DFT}(N, \mathbf{x}_A)] = (N/4)(3\log_2(N) - 7) - \log_2(N) + 3.$$

Si la séquence $\mathbf{x}_H(k)$ est la transformée de Fourier d'un signal réel, elle obéit à la relation suivante :

$$(22)\quad \mathbf{x}_H(k) = \mathbf{x}_H^*(N-k) \quad * \text{ complexe conjugué,}$$

avec $\mathbf{x}_H(0)$ et $\mathbf{x}_H(N/2)$ purement réels. Le calcul de la transformée de Fourier inverse d'une séquence définie en (22), c'est-à-dire d'une séquence hermitienne, se réduit à évaluer une cos-DFT de la partie réelle (qui est symétrique) et une sin-DFT de la partie imaginaire (qui est antisymétrique) puis de recombiner la sortie au moyen de $N - 2$ additions (la sin-DFT ayant $N - 2$ sorties non nulles). Ceci conduit à la charge de calcul suivante pour la transformation de Fourier inverse d'un signal hermitien de longueur N :

$$(23)\quad O_M[\text{DFT}(N, \mathbf{x}_H)] = N/2(\log_2(N) - 3) + 2,$$
$$O_A[\text{DFT}(N, \mathbf{x}_H)] = N/2(3\log_2(N) - 5) + 4.$$

L'algorithme précédent peut être utilisé pour évaluer une convolution circulaire réelle par transformation. La transformée avant est calculée avec l'algorithme de FFCT. Ensuite, la convolution est évaluée dans le domaine transformé, avec une multiplication réelle pour chacun des points 0 et $N/2$, et une multiplication complexe (3 multiplications/additions) pour chacun des points 1, ..., $N/2 - 1$, ce qui donne une charge totale de 3 $N/2 - 1$ multiplications et 3 $N/2 - 3$ additions. Enfin, la transformée inverse de la séquence hermitienne résultante est calculée comme ci-dessus. Vu l'optimalité de chacune des étapes utilisées, ceci donne un algorithme de convolution circulaire réelle avec un nombre minimal connu d'opérations pour une structure régulière.

Considérons maintenant le cas de la transformation de Fourier impaire, qui est une transformation de Fourier où seuls les éléments d'indice impair de la séquence de sortie sont calculés [17]. Cette transformation est définie par :

$$(24)\quad \text{DFT}_o(k, N, \mathbf{x}) := \sum_{n=0}^{N-1} x(n)\, W_{2N}^{(2k+1)n}, \; W_{2N} = e^{-\mathbf{j}2\pi/2N}, \quad k = 0, \ldots, N-1.$$

Nous définissons également, de façon similaire à (2), (3) et (4), les autres transformations impaires :

— la transformation de Fourier-cosinus impaire (cos-DFT$_o$) s'écrit :

$$(25)\quad \text{cos-DFT}_o(k, N, \mathbf{x}) := \sum_{n=0}^{N-1} x(n)\cos\left(\frac{2\pi(2k+1)n}{2N}\right), \quad k = 0, \ldots, N-1,$$

— la transformation de Fourier-sinus impaire (sin-DFT$_o$) :

$$(26)\quad \text{sin-DFT}_o(k, N, \mathbf{x}) := \sum_{n=0}^{N-1} x(n)\sin\left(\frac{2\pi(2k+1)n}{2N}\right). \quad k = 0, \ldots, N-1,$$

— la transformation en cosinus impaire (DCT$_o$) :

$$(27)\quad \text{DCT}_o(k, N, \mathbf{x}) := \sum_{n=0}^{N-1} x(n)\cos\left(\frac{2\pi(2k+1)(2n+1)}{8N}\right), \quad k = 0, \ldots, N-1.$$

L'évaluation de (24) revient évidemment à celle de (25) et (26). En traitant séparément les éléments d'indices pairs ou impairs de la séquence d'entrée, la relation (25) devient :

$$(28)\quad \text{cos-DFT}_o(k, N, \mathbf{x}) = \sum_{n=0}^{N/2-1} x(2n)\cos\left(\frac{2\pi(2k+1)n}{N}\right) + \sum_{n=0}^{N/4-1} (x(2n+1) - x(N-2n-1)) \times \cos\left(\frac{2\pi(2k+1)(2n+1)}{2N}\right),$$

La transformation de Fourier-cosinus impaire se décompose donc en une transformation de Fourier-cosinus impaire de longueur moitié et en une DCT impaire de longueur un quart. D'une façon similaire, la transformation de Fourier-sinus impaire se décompose en une transformation de Fourier-sinus de dimension moitié et en une DCT impaire de longueur un quart, avec :

$$(29)\quad \text{sin-DFT}_o(k, N, \mathbf{x}) = \sum_{n=0}^{N/2-1} x(2n)\sin\left(\frac{2\pi(2k+1)n}{N}\right) + \sum_{n=0}^{N/4-1} (-1)^n (x(2n+1) + x(N-2n-1)) \times \cos\left(\frac{2\pi(N/2-(2k+1))(2n+1)}{2N}\right).$$

Le calcul de la transformée en cosinus impaire peut être traité d'une façon similaire à celui de la DCT, avec un réarrangement de la séquence d'entrée :

$$(30)\quad y(n) = x(2n), \quad y(N-n-1) = -x(2n+1),$$

grâce à laquelle la DCT_o se récrit ainsi :

$$(31)\quad \mathrm{DCT}_o(k, N, \boldsymbol{x}) = \sum_{n=0}^{N-1} y(n) \cos\left(\frac{2\pi(4k+1)(4n+1)}{8N}\right), \quad k = 0, \ldots, N-1.$$

Cette dernière expression peut être évaluée d'une façon similaire à (12).

L'approche ainsi décrite permet d'évaluer une transformation de Fourier discrète impaire de longueur N avec un nombre d'opérations donné par :

$$(32)\quad \mathrm{O_M}[\mathrm{DFT}_o(N, \boldsymbol{x}_R)] = (N/2)(\log_2 N - 1),$$
$$\mathrm{O_A}[\mathrm{DFT}_o(N, \boldsymbol{x}_R)] = (3N/2)(\log_2 N - 1).$$

Remarquons que si la transformation est complexe, et que l'on applique l'algorithme à la partie réelle et complexe en recombinant à la fin, on obtient un nombre de multiplications égal à celui de l'approche Rader-Brenner [17] tout en économisant un nombre substantiel d'additions.

Le développement ci-dessus permet également de voir que l'idée sous-jacente de l'algorithme de FFCT (division inégale en moitié et quart) peut être utilisée dans d'autres cas comportant des transformations trigonométriques.

IV. EXTENSION AUX TRANSFORMATIONS BIDIMENSIONNELLES

Nous présenterons ici brièvement l'extension des méthodes introduites précédemment aux transformations de Fourier et en cosinus discrète bidimensionnelles, car ces dernières présentent un intérêt pratique en traitement d'image. Les algorithmes présentés suivent des approches connues (transformations polynomiales, permutations), mais nous montrerons qu'il est possible de diminuer sensiblement les charges de calcul en les combinant avec les méthodes développées dans les paragraphes II et III.

La transformation de Fourier bidimensionnelle est définie par :

$$(33)\quad X(k_1, k_2) := \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2)\, W^{n_1 k_1}\, W^{n_2 k_2}, \quad W = \mathrm{e}^{-\mathrm{j}2\pi/N},$$

avec $X(k_1, k_2) = [X(N - k_1, N - k_2)]^*$ lorsque $x(n_1, n_2)$ est réel. Lorsque k_2 est impair, (33) peut être représenté par [17] :

$$(34)\quad X(k_1, k_2) \equiv \mathrm{X}^1_{k_1}(z) \quad [\mathrm{mod}(z - W^{k_2})], \quad k_2 \text{ impair},$$

où :

$$(35)\quad X^1_{k_1}(z) \equiv \sum_{n_1=0}^{N-1} X^1_{n_1}(z)\, W^{n_1 k_1} \quad [\mathrm{mod}(z^{N/2} + 1)],$$

et

$$(36)\quad X^1_{n_1}(z) = \sum_{n_2=0}^{N/2-1} [x(n_1, n_2) - x(n_1, n_2 + N/2)]\, z^{n_2},$$

puisque $(z - W^{k_2})$ est un facteur de $(z^{N/2} + 1)$, donc de $(z^N - 1)$. Lorsque $x(n_1, n_2)$ est réel, l'évaluation de (36) requiert $N^2/2$ additions. On peut ensuite remplacer k_1 par $k_1 k_2$ (ce qui est une permutation valide puisque k_2 et N sont premiers entre eux). Comme $z \equiv W^{k_2}$, il est possible de remplacer W^{k_2} par z dans (35) :

$$(37)\quad X^1_{k_1 k_2}(z) \equiv \sum_{n_1=0}^{N-1} X^1_{n_1}(z)\, z^{n_1 k_1} \quad [\mathrm{mod}(z^{N/2} + 1)].$$

La relation (37) définit une transformation polynomiale de longueur N et de base z définie modulo $(z^{N/2} + 1)$. Cette transformation polynomiale peut être évaluée avec un algorithme du type base-2 FFT [17] et utilise $(N^2/2)\log_2 N$ additions lorsque $X^1_{k_1}$ a des coefficients réels. Puisque $X^2_{k_1 k_2}(z)$ est le degré $N/2 - 1$ et k_2 est impair (et peut donc être écrit comme $(2u + 1)$), l'équation (35) peut se récrire comme :

$$(38)\quad X(k_1(2u+1), (2u+1)) = \sum_{l=0}^{N/2-1} y(k_1, l)\, W^l\, W^{2ul},$$

où $y(k_1, l)$ est le l-ième coefficient du polynôme $\mathrm{X}^1_{k_1 k_2}(z)$. L'équation (38) correspond à l'évaluation de N transformées de Fourier impaires de longueur $N/2$. Celles-ci peuvent être évaluées selon la méthode présentée dans le paragraphe précédent.

Si k_1 et k_2 sont pairs, l'équation (33) se réduit à une transformation de dimension $N/2 \times N/2$ sur un signal réel défini par :

$$(39)\quad \mathrm{x}(n_1, n_2) + \mathrm{x}(n_1, n_2 + N/2) + \mathrm{x}(n_1 + N/2, n_2) + \mathrm{x}(n_1 + N/2, n_2 + N/2).$$

Finalement, si k_1 est impair et k_2 pair, l'équation (9) peut être récrite de façon similaire à (34-35), avec :

$$(40)\quad \mathrm{X}(k_1, 2uk_1) \equiv X^1_{2uk_1}(z) \quad [\mathrm{mod}(z - W^{k_1})],$$

où :

$$(41)\quad X^1_{2uk_1}(z) \equiv \sum_{n_2=0}^{N-1} X^1_{n_2}(z)\, z^{2un_2} \quad [\mathrm{mod}(z^{N/2} + 1)], \quad u = 0, \ldots, N/2 - 1,$$

et

$$(42)\quad X^1_{n_2}(z) = \sum_{n_1=0}^{N/2-1} [x(n_1, n_2) + x(n_1, n_2 + N/2) - x(n_1 + N/2, n_2) - x(n_1 + N/2, n_2 + N/2)]\, z^{n_1}.$$

Notons que (41) est une transformation polynomiale avec une base z^2 définie modulo $(z^{N/2} + 1)$ qui peut être évaluée, de façon identique à (37), à l'aide de $N^2/2$ $(\log_2 N - 1)$ additions. Finalement, (40) peut être récrit comme (38), donc comme $N/2$ transformations de Fourier impaires de longueur $N/2$.

Afin d'évaluer la complexité de la méthode décrite, remarquons que (39) et (42) utilisent N^2 additions supplémentaires.

Donc, l'évaluation d'une transformation de Fourier discrète de $N \times N$ sur des données réelles a été réduite au problème de l'évaluation de 3 $N/2$ transformations de Fourier impaires de longueur $N/2$ sur des séquences

réelles ainsi que d'une transformation de dimension $N/2 \times N/2$, et ceci au prix de $3\,N^2/4 \log_2 N^2 + 5\,N^2/4$ additions. On obtient ainsi la complexité de calcul suivante pour une transformation de Fourier réelle de dimension $N \times N$, avec $N = 2^M$:

(43)

$$O_M[\text{DFT}(N \times N, \boldsymbol{x}_R)] = (N^2/2)\log_2 N - (7/6)\,N^2 + 4,$$

$$O_A[\text{DFT}(N \times N, \boldsymbol{x}_R)] = (5N^2/2)\log_2 N - (13/6)\,N^2 + 56/3.$$

Considérons maintenant le cas de la transformation en cosinus bidimensionnelle définie par :

$$(44)\quad \text{DCT}(k_1, k_2, N, \boldsymbol{x}) := \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} x(n_1, n_2) \times \cos\left(\frac{2\pi(2n_1+1)k_1}{4N}\right)\cos\left(\frac{2\pi(2n_2+1)k_2}{4N}\right).$$

La séquence d'entrée peut être réarrangée d'une façon similaire à (10), avec :

$$(45)\quad y(n_1, n_2) = x(2n_1, 2n_2),\quad n_1 = 0, \ldots, N/2-1,\ n_2 = 0, \ldots, N/2-1,$$

$$y(n_1, n_2) = x(2n_1, 2N-2n_2-1),\quad n_1 = 0, \ldots, N/2-1,\ n_2 = N/2, \ldots, N-1,$$

$$y(n_1, n_2) = x(2N-2n_1-1, 2n_2),\quad n_1 = N/2, \ldots, N-1,\ n_2 = 0, \ldots, N/2-1,$$

$$y(n_1, n_2) = x(2N-2n_1-1, 2N-2n_2-1),\quad n_1 = N/2, \ldots, N-1,\ n_2 = N/2, \ldots, N-1.$$

La relation (44) devient alors :

$$(46)\quad \text{DCT}(k_1, k_2, N, \boldsymbol{x}) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1, n_2) \times \cos\left(\frac{2\pi(4n_1+1)k_1}{4N}\right)\cos\left(\frac{2\pi(4n_2+1)k_2}{4N}\right).$$

En utilisant des relations trigonométriques simples, on peut récrire (46) comme :

$$(47)\quad \text{DCT}(k_1, k_2, N, \boldsymbol{x}) = 1/2\left[\cos\left(\frac{2\pi(k_1+k_2)}{4N}\right)\times \text{cos-DFT}(k_1, k_2, N, \boldsymbol{x}) - \sin\left(\frac{2\pi(k_1+k_2)}{4N}\right)\text{sin-DFT}(k_1, k_2, N, \boldsymbol{x}) + \cos\left(\frac{2\pi(k_1-k_2)}{4N}\right)\text{cos-DFT}(k_1, N-k_2, N, \boldsymbol{x}) - \sin\left(\frac{2\pi(k_1-k_2)}{4N}\right)\text{sin-DFT}(k_1, N-k_2, N, \boldsymbol{x})\right],$$

où cos-DFT$(k_1, k_2, N, \boldsymbol{x})$ et sin-DFT$(k_1, k_2, N, \boldsymbol{x})$ sont les parties réelles et imaginaires d'une transformée de Fourier discrète à 2 dimensions qui s'écrivent comme suit :

$$(48)\quad \text{cos-DFT}(k_1, k_2, N, \boldsymbol{x}) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1, n_2) \times \cos\left(\frac{2\pi(n_1k_1 + n_2k_2)}{N}\right),$$

$$\text{sin-DFT}(k_1, k_2, N, \boldsymbol{x}) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} y(n_1, n_2) \times \sin\left(\frac{2\pi(n_1k_1 + n_2k_2)}{N}\right).$$

Les transformations en cosinus DCT$(k_1, k_2, N, \boldsymbol{x})$, DCT$(N-k_1, k_2, N, \boldsymbol{x})$, DCT$(k_1, N-k_2, N, \boldsymbol{x})$ et DCT$(N-k_1, N-k_2, N, \boldsymbol{x})$ peuvent être dérivées des cos-DFT et sin-DFT correspondantes à l'aide de 2 rotations et de quelques additions, avec :

(49)

$$\begin{bmatrix}\text{DCT}(k_1, k_2, N, \boldsymbol{x})\\ \text{DCT}(N-k_1, k_2, N, \boldsymbol{x})\\ \text{DCT}(k_1, N-k_2, N, \boldsymbol{x})\\ \text{DCT}(N-k_1, N-k_2, N, \boldsymbol{x})\end{bmatrix} = \begin{bmatrix}1&0&1&0\\0&1&0&1\\0&1&0&-1\\-0&0&1&0\end{bmatrix} \times \begin{bmatrix}R(k_1, k_2)\,O_2 \\ O_2\,R(k_1, -k_2)\end{bmatrix}\begin{bmatrix}\text{cos-DFT}(k_1, k_2, N, \boldsymbol{x})\\ \text{sin-DFT}(k_1, k_2, N, \boldsymbol{x})\\ \text{cos-DFT}(k_1, N-k_2, N, \boldsymbol{x})\\ \text{sin-DFT}(k_1, N-k_2, N, \boldsymbol{x})\end{bmatrix},$$

où :

$$(50)\quad O_2 = \begin{bmatrix}0&0\\0&0\end{bmatrix},$$

$$R(k_1, k_2) = \begin{bmatrix}\cos\left(\frac{2\pi(k_1+k_2)}{4N}\right) & -\sin\left(\frac{2\pi(k_1+k_2)}{4N}\right)\\ \sin\left(\frac{2\pi(k_1+k_2)}{4N}\right) & \cos\left(\frac{2\pi(k_1+k_2)}{4N}\right)\end{bmatrix}.$$

La relation (49) doit être évaluée pour :

$$k_1, k_2 = 0, \ldots, N/2-1,$$

et si l'on tient compte de toutes les simplifications (par exemple $k_1 = k_2$), la charge de calcul pour obtenir la transformée en cosinus à partir d'une transformée de Fourier d'une séquence réelle bidimensionnelle s'élève à $3\,N^2/2 - 2\,N$ multiplications et $5\,N^2/2 - 6\,N + 2$ additions.

La méthode correspondant à (49) peut paraître lourde, mais il se trouve que les post-multiplications dominent la charge de calcul lorsque la transformation est petite (comme 8×8 ou 16×16, des cas fréquents en codage d'image).

En utilisant les méthodes précédentes pour l'évaluation de la DFT et des post-multiplications, on obtient la charge de calcul totale ci-dessous pour une transformation en cosinus discrète bidimensionnelle de dimension $N \times N$, avec $N = 2^M$:

$$(51)\quad O_M[\text{DCT}(N \times N, \boldsymbol{x})] = (N^2/2)\log_2 N - 2\,N + N^2/3 + 8/3,$$

$$O_A[\text{DCT}(N \times N, \boldsymbol{x})] = (5\,N^2/2)\log_2 N - 6\,N + N^2/3 + 62/3.$$

Dans le tableau IV, cette complexité est comparée avec celles obtenues par d'autres méthodes, c'est-à-dire :

a) l'approche de Chen *et al.* [19] en ligne/colonne,

b) l'approche polynomiale de [21],

c) l'algorithme de FFCT en ligne/colonne,

d) la méthode précédente.

TABL. IV. — Comparaison de divers algorithmes de DCT bidimensionnels. *a*) méthode [19], *b*) méthode [21], *c*) méthode [10], *d*) méthode proposée.

Comparison of various two-dimensional DCT *algorithms.* a) *method* [19], b) *method* [21], c) *method* [10], d) *proposed method.*

Algorithme	Multiplications	Additions
a)	$2N^2\log_2 N - 3N^2 + 8N$	$3N^2\log_2 N - 3N^2 + 4N$
b)	$2N^2\log_2 N$	$5N^2\log_2 N - 2N^2$
c)	$N^2\log_2 N$	$3N^2\log_2 N - 2N^2 + 2N$
d)	$\frac{N^2}{2}\log_2 N + \frac{N^2}{3} - 2N + \frac{8}{3}$	$\frac{5N^2}{2}\log_2 + \frac{N^2}{3} - 6N + \frac{62}{3}$

Asymptotiquement, le gain est de 4 pour les multiplications et de 1/6 pour les additions si l'on compare aux méthodes actuelles (*a* et *b*).

V. MISE EN ŒUVRE

Afin de déterminer le domaine possible d'application des algorithmes FFCT, deux mises en œuvre expérimentales ont été réalisées, l'une sous forme d'un programme destiné à un ordinateur à usage général, et l'autre sous forme d'un circuit intégré spécialisé.

Le programme FFCT a été généré sous forme de code linéaire produit automatiquement à partir d'une formulation récursive de l'algorithme en langage de haut niveau (Pascal). Avec une telle approche, l'ordre des opérations a une influence déterminante sur la vitesse d'exécution, et il n'a pas été possible de déterminer l'ordre optimal, compte tenu du nombre considérable des possibilités dû à la nature combinatoire du problème.

L'influence du compilateur (le code linéaire étant en FORTRAN) sur l'efficacité est également apparue

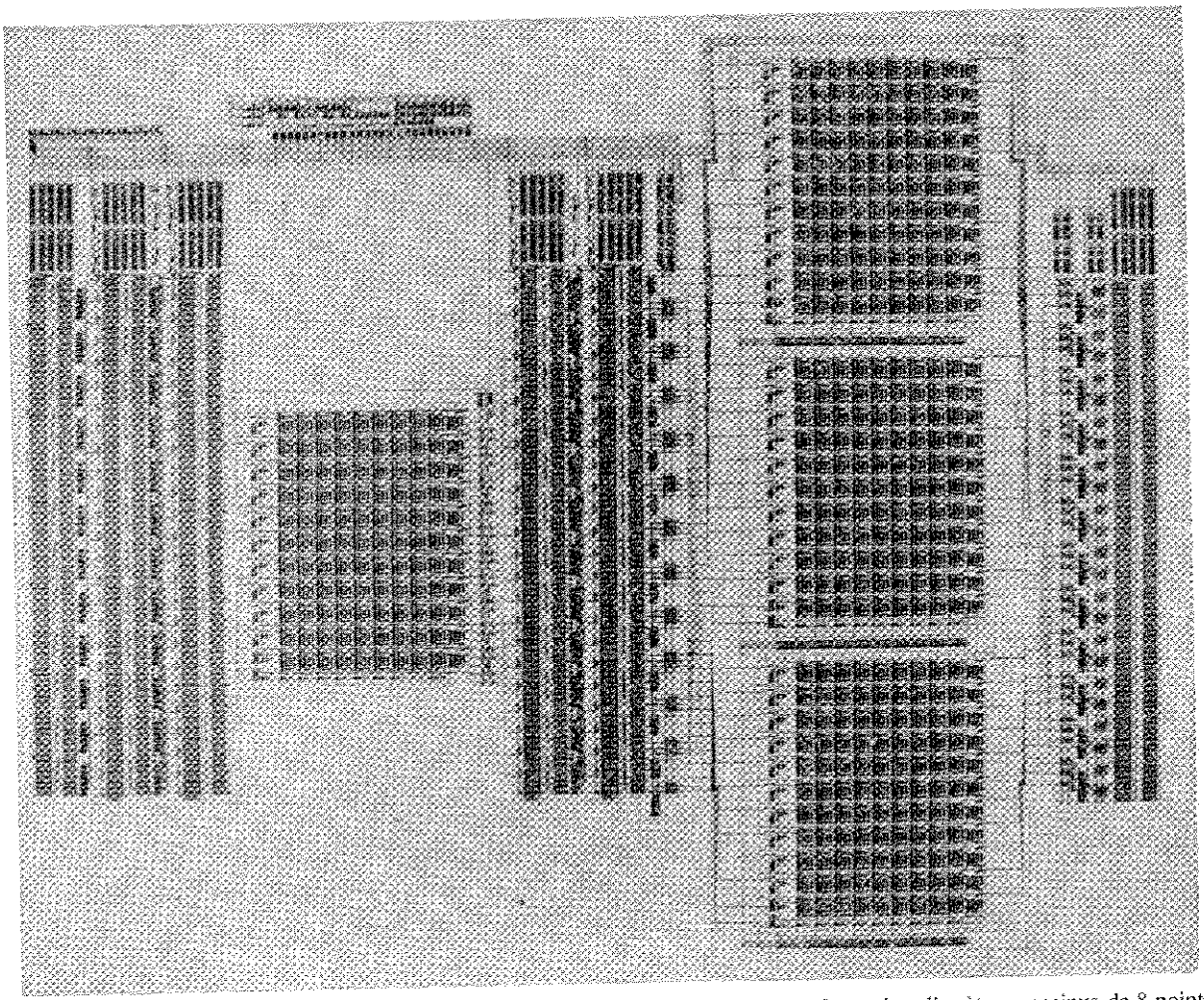

FIG. 4. — Schéma symbolique d'un circuit CMOS de 35 000 transistors réalisant une transformation discrète en cosinus de 8 points par 100 ns. Les données de 12 bits entrent à gauche puis sont traitées dans 6 étages de calcul mis en « pipeline » avant de sortir à droite. Les grandes surfaces carrées sont des multiplicateurs parallèles, et on peut reconnaître l'étage de rotation aux 3 multiplicateurs superposés.

Symbolic layout of a 35 000 transistor CMOS *circuit realizing an 8-point* DCT *every 100 ns. The 12 bit data enters on the left, goes through 6 pipelined stages of computations and leaves on the right. The large square surfaces are parallel multipliers, and one can recognize the rotation stage at its 3 superposed multipliers.*

lors de l'inspection du code machine. Idéalement, ce dernier devrait être tel que les valeurs les plus fréquemment utilisées soient stockées dans les registres du processeur.

L'algorithme de FFCT pour des séquences complexes a été comparé à l'algorithme de FFT à base 2 et à base 4 (à défaut d'un algorithme pour séquences réelles efficace). Du code linéaire pour une transformation de longueur 256 a été généré pour chacun des algorithmes considérés. L'algorithme de FFCT ayant moins d'opérations (— 8% de multiplications, — 2% d'additions) que la FFT à base 4, avait un temps d'exécution plus grand (près de 10% sur VAX-780). L'explication la plus vraisemblable est que la FFT à base 4, ayant une structure plus régulière, permet un adressage plus simple et une optimisation meilleure du code est obtenue lors de la compilation. Ceci tend à indiquer que l'algorithme FFCT est probablement mieux adapté au calcul de transformées en cosinus et de transformées de Fourier de séquences réelles qu'à l'évaluation de transformations de Fourier de séquences complexes

Notons que les algorithmes peuvent être écrits *sur place*, mais que cette propriété n'a pas été utilisée dans le code linéaire qui a été testé. La complexité de l'écriture du code n'en a pas été affectée, puisque le code est généré automatiquement.

Pour évaluer l'application des algorithmes FFCT dans le cadre d'une implantation matérielle, la réalisation d'un circuit intégré à grande échelle capable de calculer une transformée en cosinus de 8 points à 120 Mbit/s (un échantillon de 12 bits toutes les 100 ns) a été entreprise. Cette étude a fait apparaître qu'il était possible dans un tel contexte de mieux assortir l'architecture à l'algorithme et donc de tirer un meilleur parti de la complexité de calcul réduite de l'algorithme FFCT. De plus, la complexité multiplicative domine ici la complexité totale, puisque la surface d'un multiplicateur est égale à N fois la surface d'un additionneur, N étant la largeur de bits des mots. Comme l'algorithme FFCT utilise 25% de multiplications en moins que l'algorithme de Chen *et al.*, on aboutit ainsi à une réduction substantielle de surface avec l'approche proposée.

L'architecture choisie [14] utilise une arithmétique parallèle à 12 bits et permet d'atteindre un rendement élevé grâce à un chaînage des opérations (« pipeline »). Le circuit comporte 35 000 transistors et sera intégré en technologie CMOS 2,5 μm. La figure 4 montre le schéma symbolique du circuit qui a été conçu.

VI. CONCLUSION

Des algorithmes efficaces pour la transformation de Fourier et cosinus discrète de longueur $N = 2^M$ ont été présentés. Ceux-ci ont été particularisés au cas de séquences symétriques et antisymétriques, ainsi qu'à celui de la transformée impaire. La généralisation au cas bi-dimensionnel a également été présentée. Finalement, quelques expériences avec une implantation logicielle (code linéaire) et matérielle (une puce réalisant une transformée en cosinus) ont été présentées.

Manuscrit reçu le 25 *avril* 1985,
accepté le 25 *juin* 1985.

BIBLIOGRAPHIE

[1] COOLEY (J. W.), TUKEY (J. W.). An algorithm for the machine calculation of complex Fourier series. *Math. of Comput.*, USA (avr. 1965), **19**, pp. 297-301.

[2] YAVNE (R.). An economical method for calculating the discrete Fourier transform. *AFIPS Proc.*, Fall Joint Comput. Conf., Washington (1968), **33**, pp. 115-125.

[3] RADER (C. M.), BRENNER (N. M.). A new principle for fast Fourier transformation. *IEEE Trans. ASSP*, USA (juin 1976), **24**, pp. 264-265.

[4] BRUUN (G.). z-Transform DFT filters and FFT's. *IEEE Trans. ASSP*, USA (fév. 1978), **26**, pp. 56-63.

[5] SINGLETON (R.). An algorithm for computing the mixed radix fast Fourier transform. *IEEE Trans. AU*, USA (juin 1969), **17**, pp. 93-103.

[6] WINOGRAD (S.). On computing the discrete Fourier transform. *Proc. Nat. Acad. Sci.*, USA (avr. 1976), **73**, pp. 1005-1006.

[7] KOLBA (D. P.), PARKS (T. W.). A prime factor FFT algorithm using high-speed convolution. *IEEE Trans. ASSP*, USA (août 1977), **25**, pp. 281-294.

[8] MORRIS (L. R.). A comparative study of time efficient FFT and WFTA programs for general purpose computers. *IEEE Trans. ASSP*, USA (avr. 1979), **26**, pp. 141-150.

[9] DUHAMEL (P.), HOLLMANN (H.). Split radix FFT algorithm. *Electronics Lett.*, GB (jan. 1984), **20**, n° 1.

[10] VETTERLI (M.), NUSSBAUMER (H. J.). Simple FFT and DCT algorithms with reduced number of operations. *Signal Processing*, NL (juin 1984), **6**, n° 4, pp. 267-278.

[11] MARTENS (J. B.). Recursive cyclotomic factorisation. A new algorithm for calculating the discrete Fourier transform. *IEEE Trans. ASSP*, USA (août 1984), **32**, n° 4, pp. 750-761.

[12] JAIN (A. K.). Image data compression : a review. *Proc. IEEE*, USA (mars 1981), **69**, n° 3, pp. 349-389.

[13] MORRIS (L. R.). Automatic generation of time efficient digital signal processing software. *IEEE Trans. ASSP*, USA (fév. 1977), **25**, pp. 74-78.

[14] VETTERLI (M.), LIGTENBERG (A.). A discrete Fourier-cosine transform chip. To be published in *IEEE Trans. COM*, USA.

[15] AHMED (N.), NATARAJAN (T.), RAO (K. R.). Discrete cosine transform. *IEEE Trans. C.*, USA (jan. 1974), **23**, pp. 88-93.

[16] NARASIMHA (M. J.), PETERSON (A. M.). On the computation of the discrete cosine transform. *IEEE Trans. COM.*, USA (juin 1978), **26**, pp. 934-936.

[17] NUSSBAUMER (H. J.). Fast Fourier transform and convolution algorithms. *Springer Verlag*, Berlin (1982).

[18] BRIGHAM (E. O.). The fast Fourier transform. *Prentice-Hall*, Inc., Englewood Cliffs, NJ (1974).

[19] CHEN (W. H.), SMITH (C. H.), FRALICK (S. C.). A fast computational algorithm for the discrete cosine transform. *IEEE Trans. COM*, USA (sep. 1977), **25**, pp. 1004-1009.

[20] SIEGMAN (A. E.). How to compute two complex even Fourier transforms with one transform step. *Proc. IEEE*, USA (mars 1975), p. 544.

[21] PEI (S. C.), HUANG (E. F.). Improved 2 D discrete cosine transforms using generalized polynomial transforms and DFT's. *Proc. IEEE.* Int. Conf. on Communic., Amsterdam (1984), pp. 242-244.