Implantation du détecteur de contours Canny-Deriche optimisé sous forme d'un circuit spécifique

ASIC Implementation of the Optimized Canny-Deriche Edge Detector

par Lionel TORRES*, El-Bay BOURENNANE, Michel ROBERT, Michel PAINDAVOINE**

Université de Bourgogne, Laboratoire LE2I 6, boulevard Gabriel
F-21000 Dijon
* ATMEL-ES2, ZI Rousset
F-13106 Rousset
** Laboratoire LIRMM, UMR 9928 CNRS,
Université de Montpellier II
161, rue Ada F-34392 Montpellier cedex 5

résumé et mots clés

Nous présentons dans cet article l'implantation d'un processeur dédié intégrant le détecteur de contours de Canny–Deriche optimisé.
Après un bref rappel des équations du filtre, nous exposons différentes techniques d'accélération des filtres récursifs et notamment une technique d'accélération de calcul par anticipation. Cette méthode nous a amené à la conception d'un premier circuit dont la fréquence de fonctionnement est de l'ordre de 20 Mhz pour une surface de silicium de 60 mm^2. En utilisant une méthode de redistribution locale des registres, nous avons réalisé un second circuit, capable de traiter un pixel à une fréquence de 33 MHz pour une surface en silicium inférieure à 30 mm^2. Les deux approches sont alors comparées. Cette étude a conduit à la fabrication d'un processeur dédié réalisé en technologie CMOS 1 μm, et testé avec succès.

Traitement d'images, Détection de contours, Circuits intégrés, Filtres numériques.

abstract and key words

We present in this paper the design and the implementation of the optimized Canny–Deriche edge detector.
After a brief reminder of the filter's equations, we expose different techniques to speed up the sampling rate of the IIR filter. In particular, to improve throughput rate of the IIR filter, we present a look–ahead with a decomposition technique. This method leads us to design a first chip, which performs over 20 Mhz sampling rate with a silicon area of 60 mm^2. Using a local register retiming method, we have designed a second circuit, which is able to process a pixel in 33 MHz with a silicon area of 30 mm^2. These two approaches are compared. This work leads us to an ASIC designed in a CMOS 1 μm technology and succesfully tested.

Image processing, Edge detector, Integrated circuits (chip), Digital filters.

1. introduction

Les systèmes de vision artificielle en temps réel deviennent de plus en plus exigeants et imposent des contraintes temporelles de plus en plus élevées. La vidéo rapide impose des temps de traitement inférieurs à 40 ms par image. Parallélisme algorithmique et architectural sont souvent utilisés pour accélérer les calculs. En début de conception deux questions se posent :

– Quelle architecture adopter?

– L'algorithme et l'architecture sont-ils optimisés du point de vue temps d'exécution/espace occupé (nombre de composants, surface de silicium)?

La réponse à la première question exige la connaissance des différents types d'architectures et de machines parallèles. La

réponse à la deuxième question nécessite l'appréhension des techniques de parallélisation d'algorithmes et d'optimisation en vue d'une adéquation machine parallèle — algorithme parallèle.

Pour répondre à des contraintes de type « temps réel », il convient donc de répartir les tâches en sous tâches (parallélisme des fonctions) ou bien de partager les données en sous ensembles (parallélisme des données) afin de les traiter d'une façon parallèle.

Généralement, quel que soit le domaine d'application de la vision artificielle, l'image initiale subit un traitement qui selon son type peut être qualifié d'un traitement bas, moyen ou haut niveau. Dans cet article, nous nous intéressons uniquement au traitement bas niveau.

Une partie du traitement bas niveau repose sur la détection (par filtrage) des variations brusques des niveaux de luminance dans l'image en vue de l'obtention des segments de contours, des extrémités et des angles.

Nous nous intéressons plus particulièrement à une approche du type filtrage numérique, en étudiant les méthodes permettant l'accélération des calculs dans les filtres récursifs.

Plusieurs solutions ont été proposées pour l'accélération des calculs depuis une dizaine d'années. Cependant, si ces solutions sont adaptées à des implantations du type multiprocesseurs, elles demeurent difficilement intégrables sur un circuit intégré spécifique (ASIC).

Pour cette raison, nous détaillons une technique de calcul par anticipation très adaptée à l'accélération des filtres récursifs et à l'intégration du filtre anticipé sous forme d'un ASIC et nous appliquons cette technique de restructuration d'algorithmes au filtre de Canny-Deriche optimisé [1].

Le présent article est organisé comme suit :

La partie 2 est consacrée au modèle du contour et à la présentation du filtre (détecteur de contours) de Canny–Deriche optimisé.

Dans la partie 3 nous développons les méthodes d'implantations du filtre.

En partie 4 nous traitons la précision des calculs dans le filtre.

En partie 5 nous traitons l'intégration du filtre obtenu par calcul par anticipation sous forme d'un circuit spécifique.

En partie 6 nous montrons l'implantation directe du filtre sans anticipation mais avec une redistribution locale des registres internes au filtre.

2. le filtre

La détection de contours est l'une des opérations de base de tout système de vision artificielle (reconnaissance de caractères, détection de défauts sur des pièces industrielles...). Les contours caractérisent une variation brusque et large des niveaux de luminance dans une image. La détection de ces variations de luminance et la localisation de ces variations ont fait l'objet de nombreuses publications.

Nous avons proposé [2] un filtre récursif d'ordre 3 qui améliore sensiblement les performances de détection (au sens des critères de Canny) par rapport à un filtre de type Deriche (optimisé correctement pour le contour choisi). Nous présentons, dans ce paragraphe, notre modèle de contour ainsi que le filtre obtenu par optimisation.

L'acquisition d'une scène réelle, en relief, par une caméra implique la présence d'un certain flou que nous avons approximé par la fonction contour suivante :

$$\begin{cases} c(x) = 1 - \dfrac{e^{-sx}}{2} & \text{pour } x \geq 0 \\ & \text{avec } s > 0 \\ c(x) = \dfrac{e^{sx}}{2} & \text{pour } x < 0 \end{cases} \tag{1}$$

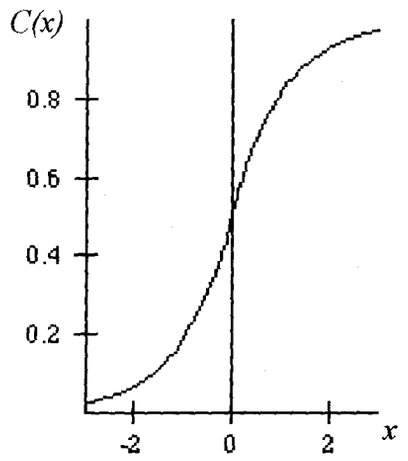


Figure 1. – Contour sous forme exponentielle avec $s = 0,5$.

Le filtre optimum [2] qui répond aux trois critères (selon Canny : rapport signal sur bruit, localisation et suppression des réponses multiples) est donné par la fonction suivante :

$$\begin{aligned} f^-(x) &= A\left(ke^{\alpha x}\sin(\omega x) + e^{\alpha x}\cos(\omega x) - e^{sx}\right) \\ &\quad \text{pour } x \leq 0 \\ f^+(x) &= A\left(ke^{-\alpha x}\sin(\omega x) - e^{-\alpha x}\cos(\omega x) + e^{-sx}\right) \\ &\quad \text{pour } x > 0 \end{aligned} \tag{2}$$

Cette fonction peut être implantée sous forme d'un filtre récursif, stable d'ordre 3 suivant :

$$\begin{aligned} y^+(i) = a_1x(i-1) + a_2x(i-2) + a_3y^+(i-1) \\ - a_4y^+(i-2) + a_5y^+(i-3) \end{aligned}$$

$$y^-(i) = -a_1 x(i+1) - a_2 x(i+2) + a_3 y^-(i+1) - a_4 y^-(i+2) + a_5 y^-(i+3) \quad (3)$$

avec :

$$a_1 = A\left(ke^{-\alpha}\sin(\omega) + e^{-s} - e^{-\alpha}\cos(\omega)\right)$$
$$a_2 = -A\left(ke^{-(s+\alpha)}\sin(\omega) + e^{-(s+\alpha)}\cos(\omega) - e^{-2\alpha}\right)$$
$$a_3 = 2e^{-\alpha}\cos(\omega) + e^{-s}$$
$$a_4 = e^{-2\alpha} + 2e^{-(s+\alpha)}\cos(\omega)$$
$$a_5 = e^{-(s+2\alpha)}$$

Cet opérateur correspond à un filtre récursif d'ordre 3 et ses performances peuvent être facilement exprimées en fonction de m et ω ($m = \alpha \cdot s$ détermine la largeur du filtre $f(x)$ et ω le nombre de passages par zéro du filtre $f(x)$ au voisinage de l'origine).

Dans [2] nous avons discuté plus en détail des performances de cet algorithme. Nous avons montré que pour un choix correct de m et ω, et dans le cas d'images floues et bruitées (s compris entre 0.1 et 2) notre opérateur permet d'obtenir des performances supérieures à celles obtenues par Deriche.

L'implantation sous forme récursive de notre filtre nous a permis d'éviter la troncature de la réponse impulsionnelle de ce filtre, facilitant ainsi l'utilisation de ce filtre dans le cas d'une application multi-échelles.

3. méthodes d'implantations du filtre

3.1. implantation sur des circuits reconfigurables (FPGA)

Dans le cadre de notre étude, il est intéressant de citer l'approche proposée par [3]. Elle consiste en la réalisation du filtre de Deriche avec une réponse impulsionnelle finie (RIF). Une fenêtre rectangulaire d'une taille 17 centrée en zéro est utilisée pour déterminer les termes exacts de la réponse impulsionnelle finie. De plus l'intégration matérielle s'en trouve plus aisée. En effet cette étude a conduit à la conception de l'opérateur de détection de contours de Deriche en temps réel pour des images de 512 × 512 pixels. L'implantation a été réalisée avec des circuits programmables du type Xilinx XC 4010, une carte d'acquisition vidéo, plus une mémoire permettant de charger 17 lignes consécutives d'une image. Il est à noter cependant que cette implantation sur des composants programmables est concevable uniquement dans le cas où les coefficients des multiplieurs sont constants.

A titre d'exemple, la complexité mise en place pour la réalisation du dérivateur vertical et horizontal, ainsi que le calcul de la norme s'élève à : 146 entrées/sorties, 707 bascules de type D, 785 générateurs de fonctions (composants principaux des blocs logiques des FPGAs Xilinx).

Il est évident que l'intérêt d'une telle approche provient du fait qu'il est possible à tout instant de changer la configuration de ces composants, permettant ainsi de modifier aisément l'architecture des filtres.

3.2. implantation directe du filtre de Canny-Deriche optimisé

Une autre solution consiste à intégrer ces architectures sur des processeurs dédiés spécifiques (ASIC). Nous allons analyser l'implantation directe du filtre de Canny–Deriche optimisé sur de tels composants.

Les transformées en Z des filtres récursifs données par les équations (2) sont les suivantes :

$$TZ^+ = \frac{Y^+(z)}{X^+(z)} = \frac{a_1 z^{-1} + a_2 z^{-2}}{1 - a_3 z^{-1} + a_4 z^{-2} - a_5 z^{-3}}$$
$$TZ^- = \frac{Y^-(z)}{X^-(z)} = -\frac{a_1 z + a_2 z^2}{1 - a_3 z + a_4 z^2 - a_5 z^3} \quad (4)$$

L'architecture typique correspondante est donnée en Figure 2. L'intégration d'une telle architecture pose un certain nombre de problèmes. En effet nous pouvons constater que le chemin critique est composé d'un multiplieur et de quatre additionneurs et constitue une chaine combinatoire complexe. Celle–ci favorise la création de pics parasites (« glitchs ») dus à l'asynchronisme entre les opérateurs arithmétiques, affectant sensiblement les performances électriques.

L'intégration de ce filtre à l'aide de l'outil de CAO OPUS (Cadence Design Framework II) en technologie CMOS 1μm a montré ses limites. En effet la fréquence de traitement est de l'ordre de 22 Mhz (pour une surface de silicium de 7.4 mm2), si notre objectif est la réalisation d'un opérateur de détection de contours en temps réel pour des images de dimension supérieure à

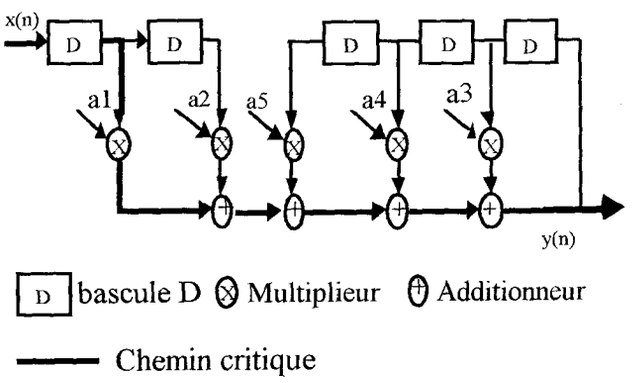


Figure 2. – Exemple : implantation directe de TZ^+.

512×512 pixels (1024×1024) cela est difficilement envisageable avec cette architecture.

Dans le but de répondre à des applications temps réel haut débit, nous nous intéressons aux diverses méthodes permettant d'accélérer les calculs.

3.3. implantation du filtre avec accélération des calculs

Le filtrage numérique occupe une place importante dans les domaines du traitement du signal et des images. Pour augmenter les performances de ces filtres pour des applications temps réel, l'approche traditionnelle consiste à augmenter la vitesse des composants matériels ou à programmer des matrices de processeurs spécialisés tels que les DSP (Digital Signal Processor). L'implantation des systèmes parallèles dans des circuits dédiés (ASICs), permet d'atteindre des fréquences élevées. La classe d'application de tels circuits reste cependant limitée. Dans ce qui suit nous donnons une solution permettant d'accélérer les calculs du filtre $f(x)$.

Avant de commencer cela, nous définissons d'abord la période d'échantillonnage minimale liée à une structure donnée du filtre à implanter.

3.3.1. définition de la période d'échantillonnage

Tout filtre numérique possède une période d'échantillonnage minimale imposée par son implantation. Afin de mieux introduire cette notion, nous donnons l'exemple suivant :

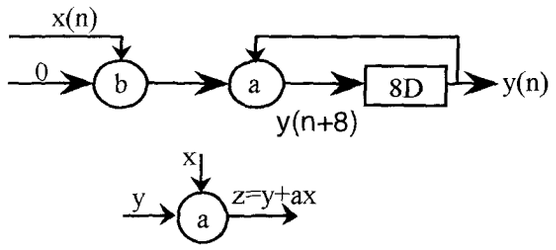


Figure 3. – FiltreIIR entrelacé.

La période d'échantillonnage du filtre IIR est limitée par sa partie récursive. Elle est donnée par :

$$T_\infty = \frac{1}{L} \max \left[\frac{D_l}{M_l} \right], \; l \in S_l \tag{5}$$

La boucle l, élément de S_l qui vérifie cette équation est appelée : *boucle critique*.

S_l : Ensemble des boucles de retour dans le graphe de fluence (dans ce graphe, il y a une seule boucle de retour).

D_l : Temps de calcul associé à chaque boucle (ici $T_a + T_m$). T_a est le temps d'une opération d'addition et T_m est le temps d'une opération de multiplication.

M_l : représente le nombre total de délais dans la boucle l (ici $l = 1$ et $M_l = 8$).

L : représente le rapport entre la fréquence des échantillons d'entrée et celle des registres internes au graphe (cas des implantations sous forme de blocs).

Ici, nous prenons $L = 1$.

La période d'itération du graphe ci-dessus est alors :

$$T_\infty = \frac{(T_a + T_m)}{8} \tag{6}$$

3.3.2. anticipation par ajout de pôles dispersés

Soit la fonction de transfert du filtre $h(n)$ suivante :

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 - \sum_{k=0}^{N} a_i z^{-k}} \tag{7}$$

La fonction de transfert $H(z)$ de $h(n)$ est transformée de telle façon que son dénominateur ne contienne que les N termes suivants :

$$z^{-M}, z^{-2M}, z^{-3M}, z^{-4M}, \ldots, z^{-NM}.$$

La sortie du filitre $y(n)$ est fonction uniquement des termes $y(n-M), y(n-2M), \ldots,$ et $y(n-NM)$.

Dans la fonction de transfert originale $H(z)$, pour chacun de ses pôles, nous introduisons $(M-1)$ pôles et zéros qui s'éliminent, et qui ont un module égal au pôle original ce qui garantit la stabilité du filtre modifié au même titre que celle du filtre initial. Si la fonction de départ a un pôle $z = p$, nous ajoutons $(M-1)$ pôles et zéros aux points $z = p \exp\left(\frac{j2\pi k}{M}\right)$ pour $k = 1, 2, \ldots, (M-1)$ afin d'avoir dans la boucle de retour critique M étages de pipeline.

Exemple :

Soit le filtre IIR du premier ordre suivant :

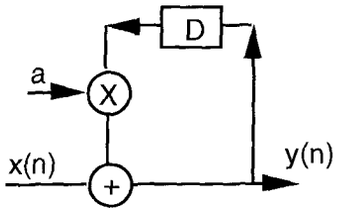


Figure 4. – Filtre IIR d'ordre 1.

Le module du pôle « a » est inférieur à l'unité. $H(z)$ a un pôle à la position $z = a$. En pipelinant avec 3 étages, on obtient un filtre stable :

$$H'(z) = \frac{1 + az^{-1} + a^2 z^{-2}}{1 - a^3 z^{-3}}$$

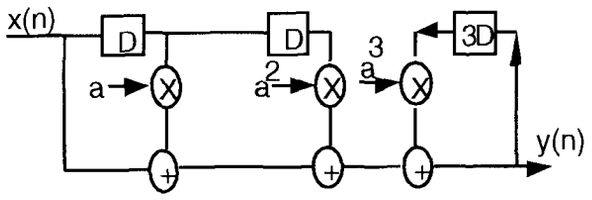


Figure 5. – Filtre équivalent obtenu par anticipation ($M = 3$).

Si le filtre initial est stable alors le filtre transformé $H'(z)$ est aussi stable car les distances par rapport à l'origine des pôles du filtre transformé sont identiques à celles du filtre original.
La partie non récursive nécessite $(NM + 1)$ multiplications et la partie récursive nécessite N multiplications. La complexité est linéaire par rapport à M. Au total nous avons besoin de $(NM + N + 1)$ multiplications pipelinées pour accélérer un filtre d'ordre N d'un facteur M. Chaque multiplieur nécessite M registres pour le pipeliner, soit une complexité proportionnelle au carré de M pour implanter le filtre complet. Cette complexité linéaire impose une limitation à l'ordre du filtre à implanter. Nous présentons dans ce qui suit une méthode qui permet de réduire cette complexité linéaire à une complexité logarithmique.

3.3.3. *anticipation par ajout de pôles dispersés avec une décomposition en puissance de 2*

Soit la partie récursive d'un filtre d'ordre N avec k registres de pipeline dans sa boucle de retour décrite par :

$$H(z) = \frac{1}{1 - \sum_{i=1}^{N} q_i(k)\, z^{-ik}} \tag{8}$$

La fonction de transfert originale correspond à $k = 1$ et $q_i(1) = a_i$. Le filtre équivalent à $2k$ étages de pipeline, s'obtient en multipliant le numérateur et le dénominateur par :

$$\left(1 - \sum_{i=1}^{N} (-1)^i q_i(k)\, z^{-ik}\right) \tag{9}$$

D'où :

$$H(z) = \frac{1 - \sum_{i=0}^{N} (-1)^i\, q_i(k)\, z^{-ik}}{\left(1 - \sum_{i=1}^{N} q_i(2k)\, z^{-2ik}\right)} \tag{10}$$

Avec les $q_i(2k)$ reliés aux séquences $q_i(k)$ (suivant que l'ordre N du filtre est pair ou impair) par les relations données en annexe.
L'application de cette transformation une deuxième fois permet d'avoir un filtre pipeliné 4 fois, puis 8 fois et ainsi de suite. A partir de la fonction originale, il faut appliquer $\log_2(M)$ transformations pour avoir M étages de pipeline. Chaque transformation permet une augmentation de la vitesse d'un facteur de 2 et apporte N multiplieurs supplémentaires à la structure du filtre.

En général, la fonction de transfert à M étages de pipeline est donnée par :

$$H(z) = \frac{\left(\sum_{i=0}^{N} b_i(z^{-i})\right)^{\log_2 (M)-1} \prod_{k=0} \left[1 - \sum_{i=1}^{N} (-1)^i q_i(2^k) z^{-i2^k}\right]}{\left(1 - \sum_{i=1}^{N} q_i(M) z^{-iM}\right)} \tag{11}$$

Cette fonction de transfert peut être implantée avec $(2N + N \log_2(M) + 1)$ multiplications.

3.3.4. *application de l'anticipation au filtre* $f(x)$

La méthode d'accélération décrite ci-dessus est valable quelque soit l'ordre du filtre utilisé. Nous allons l'appliquer au filtre récursif $f(x)$ d'ordre 3 : (voir équation (3) section 2).
La transformée en Z de la réponse impulsionnelle de ce filtre est :

$$\begin{aligned} TZ^+ &= \frac{Y^+(z)}{X^+(z)} = \frac{a_1 z^{-1} + a_2 z^{-2}}{1 - a_3 z^{-1} + a_4 z^{-2} - a_5 z^{-3}} \\ TZ^- &= \frac{Y^-(z)}{X^-(z)} = \frac{a_1 z + a_2 z^2}{1 - a_3 z + a_4 z^2 - a_5 z^3} \end{aligned} \tag{12}$$

avec : $X(z) = TZ(x)$.
La fonction de transfert de ce filtre TZ^+ (TZ^-) est transformée par anticipation à 4 étages de pipeline (limité par la capacité d'intégration dans un ASIC). Le filtre résultant de la transformation est un filtre d'ordre 12 dont l'équation de transfert $H(z)$ est donnée par (13).
Cette fonction de transfert peut être vue comme étant un produit de 4 fonctions de transfert. Celles-ci sont implantées en cascade comme cela est explicité en figure 6.

$$\begin{aligned} H(z) = {} & \frac{(a_1 z^{-1} + a_2 z^{-2})(1 + a_3 z^{-1} + a_4 z^{-2} + a_5 z^{-3})}{1 - [(a_3^2 - 2a_4)^2 + 2(-a_4^2 + 2a_3 a_5) z^{-4}} \\ & \frac{\left[1 + (a_3^2 - 2a_4) z^{-2} - (-a_4^2 + 2a_3 a_5) z^{-4} + a_5^2 z^{-6}\right]}{+ \left(-(-a_4^2 + 2a_3 a_5)^2 + 2a_5^2 (a_3^2 - 2a_4)\right) z^{-8} + a_5^4 z^{-12}]} \end{aligned} \tag{13}$$

Figure 6. – Implantation en cascade de $H(z)$.

Avec :

$$\begin{aligned} H_1(z) &= a_1 z^{-1} + a_2 z^{-2} \\ H_2(z) &= 1 + a_3 z^{-1} + a_4 z^{-2} + a_5 z^{-3} \\ H_3(z) &= 1 + (a_3^2 - 2a_4) z^{-2} - (-a_4^2 + 2a_3 a_5) z^{-4} + a_5 z^{-6} \\ H_4(z) &= \frac{1}{1 - [((a_3^2 - 2a_4)^2 + 2(-a_4^2 + 2a_3 a_5))\, z^{-4} + \\ & \frac{1}{\left(-(-a_4^2 + 2a_3 a_5)^2 + 2a_5^2 (a_3^2 - 2a_4)\right) z^{-8} + a_5^4 z^{-12}]} \end{aligned} \tag{14}$$

Le graphe de fluence correspondant est donné ci-après : figure 7.

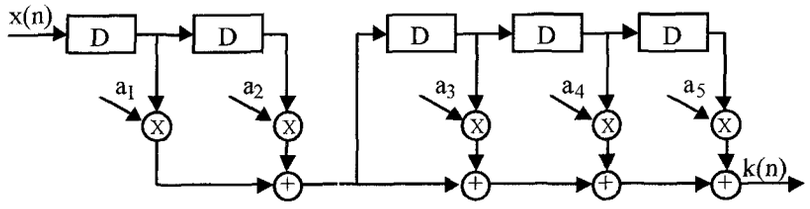


Figure 7. – Implantation en cascade de $H_1(z)$ et de $H_2(z)$.

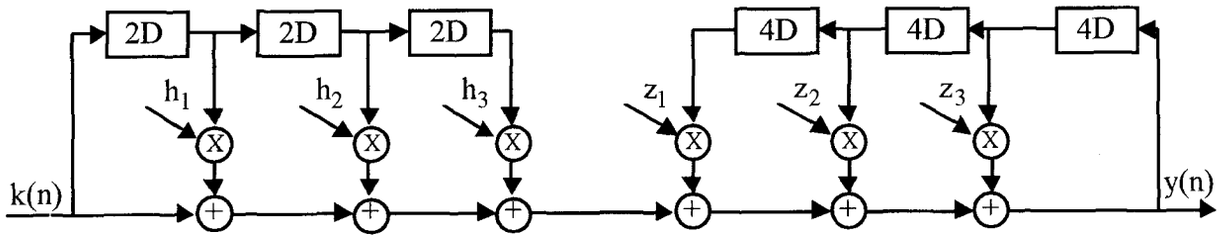


Figure 8. – Implantation en cascade de $H_3(z)$ et de $H_4(z)$. Les coefficients $a_i (i = 1, \ldots, 5)$, h_j et $z_j (j = 1, 2, 3)$ sont déduits directement de $H(z)$.

3.3.5. *redistribution globale des registres (retiming)*

Les registres supplémentaires obtenus à partir de l'anticipation de calcul n'ont un intérêt que si ils sont exploités convenablement. La redistribution des registres (« retiming ») [4], [5] est une technique permettant de déplacer les délais (retards) dans un graphe type flot de données et de mieux exploiter ces retards.

En exploitant ce principe, nous avons redistribué les registres de la boucle de retour du filtre $H(z)$ (figure 9a.) et le résultat est montré sur la Figure 9b. où $e(n)$ est la sortie de la partie non récursive du filtre anticipé.

Les registres 3D sont redistribués à l'intérieur de chaque multiplieur afin d'obtenir les étages de pipeline. Cette redistribution permet aussi de pipeliner les additionneurs entre eux. La partie non récursive peut utiliser de tels multiplieurs sans introduire aucune modification de la fonction de transfert, mais seulement une latence de la sortie $y(n)$.

Après avoir exposé les différentes techniques d'accélération des filtres récursifs, nous avons montré que l'accélération des calculs par anticipation est une méthode très adaptée en vue d'une intégration d'un filtre dans un ASIC.

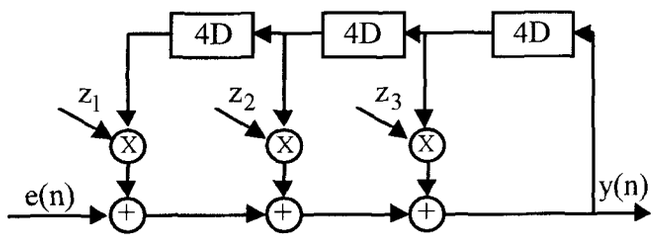


Figure 9a. – Boucle de retour $H(z)$.

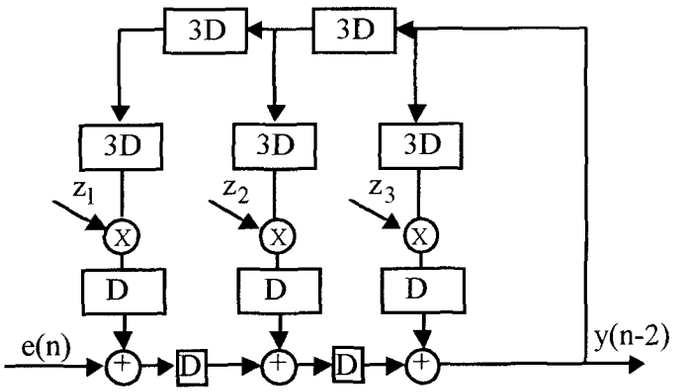


Figure 9b. – Redistribution des registres dans la boucle de retour de $H(z)$.

Avant d'aborder l'application de ces méthodes à la réalisation de notre circuit, nous allons étudier l'influence du codage des données et des coefficients sur la précision des calculs.

4. la précision des calculs dans le filtre numérique de Canny-Deriche optimisé

La description du filtre de Canny-Deriche optimisé[1] que nous avons résumé dans le paragraphe 2, n'a pas tenu compte du format de codage des coefficients et des données de calcul.

L'arithmétique flottante sur 64 bits permet de limiter le problème d'arrondi ou de débordement en cours de calcul mais présente l'inconvénient de l'emploi d'un nombre important de bits de codage.
Aussi l'arithmétique fixe, avec un nombre de bits de codage plus réduit, est plus souvent utilisée dans le cas de l'implantation des filtres numériques pour le traitement d'images. Dans ce cas, on adapte l'architecture (nombre de bits) à la nature du signal et à la précision souhaitée.

Différentes approches théoriques sont envisageables[6], [7] pour étudier le nombre de bits de codage et font appel à des méthodes qui consistent à calculer l'erreur quadratique entre la réponse du filtre en arithmétique « réelle » et la réponse du filtre en arithmétique « fixe ». Le nombre de bits de codage correspond aux configurations qui minimisent cette erreur quadratique.

Ces approches ont été testées par différents auteurs [8], [9]. Zarka [8] a montré que pour le filtre de Deriche [10], un nombre de 8 bits de codage pour les coefficients et un nombre de 16 bits pour les données de calculs sont suffisants pour minimiser l'erreur quadratique. De même Sariffudin [9] a montré que pour le filtre de Canny-Deriche optimisé [2], un nombre de 12 bits pour les coefficients et un nombre de 12 bits pour les calculs sont suffisants pour minimiser l'erreur quadratique.

Une autre approche originale a été proposée par Kamle [3] pour la réalisation du filtre de Canny–Deriche avec une réponse impulsionnelle finie. En se plaçant dans quelques cas particuliers de a, il a pu montré que les coefficients peuvent se coder sur 8 bits et que les données de calculs peuvent se coder sur 12 bits pour maintenir une réponse correcte en sortie du filtre avec un niveau maximum égal à 255.

Pour confirmer les résultats précédents obtenus pour les implantations récursives[8], [9], nous avons choisi une approche expérimentale. Nous avons positionné en entrée du filtre une fenêtre rectangulaire d'amplitude 255 et centrée de manière à éliminer les effets de bord (figure10).

Nous présentons, dans le tableau 1, les erreurs quadratiques moyennes suivant les paramètres « s » et les différents codages des coefficients et des calculs intermédiaires.

Pour mesurer l'écart entre la réponse impulsionnelle sans troncature (donnée par les calculs sur une machine SUN) et celle dont nous tronquons les coefficients et les calculs internes, nous définissons l'erreur quadratique comme :

$$\sigma^2 = \left(\frac{1}{N}\right)\left(\sum_{i=0}^{i=255}(y(i) - \tilde{y}(i) - \overline{y})\right)^2$$

avec

$$\overline{y} = \sum_{i=0}^{i=255}(y(i) - \tilde{y}(i)) \text{ et } N = 256 \qquad (15)$$

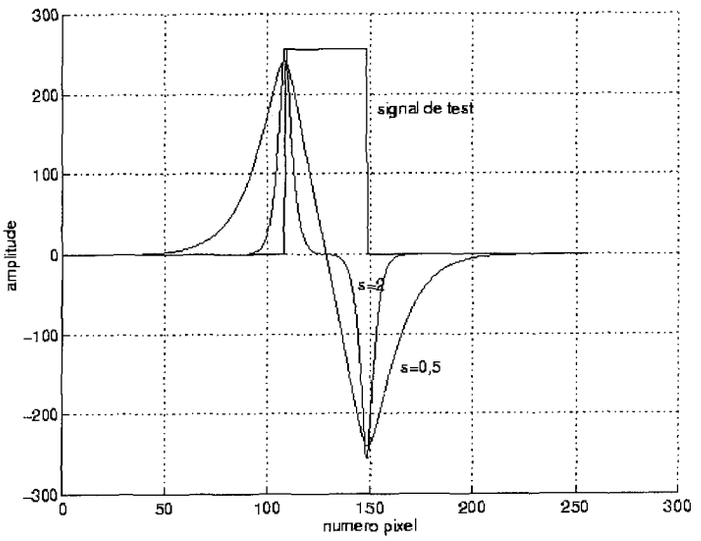


Figure 10. – Echelon d'entrée et réponse du fitre pour $s = 2$ et $s = 0,5$.

$y(i)$ réponse réelle, $\tilde{y}$ réponse avec coefficients et calculs internes tronqués.

Ce tableau montre que pour $s \leq 0,5$, les erreurs sont très importantes ceci étant dû principalement au mauvais choix de s (vue la forme du contour) plutôt que du codage.
Pour $s \geq 1$, ce tableau montre qu'à partir d'un codage de 12 bits, pour les coefficients et les données, l'erreur quadratique moyenne est de l'ordre de l'unité (voire inférieure) ce qui garantit donc une déviation moyenne inférieure à un niveau de gris (pour une échelle de 256 niveaux de gris). La première colonne de ce tableau représente le nombre de bits de codage des coefficients et le nombre de bits de codage des données (calculs internes).

Tableau 1 – Erreurs quadratiques moyennes

σ^2	$s = 0,5$	$s = 1$	$s = 2$	$s = 4$
10–12 bits	8000	32,7	16,2	0,29
12–12 bits	1150	1,5	0,75	0,052
16–16 bits	1,8	0,0095	0,0079	0,00085

Nous avons donc pu confirmer le choix du codage sur 12 bits et à titre d'illustration nous montrons sur les figures 11, 12 et 13 les résultats obtenus pour un disque bruité de rapport signal sur bruit égal à 17 db.

D'une manière identique, nous avons pu tirer la même conclusion concernant la précision des calculs pour une implantation directe du filtre (voir le paragraphe 6).

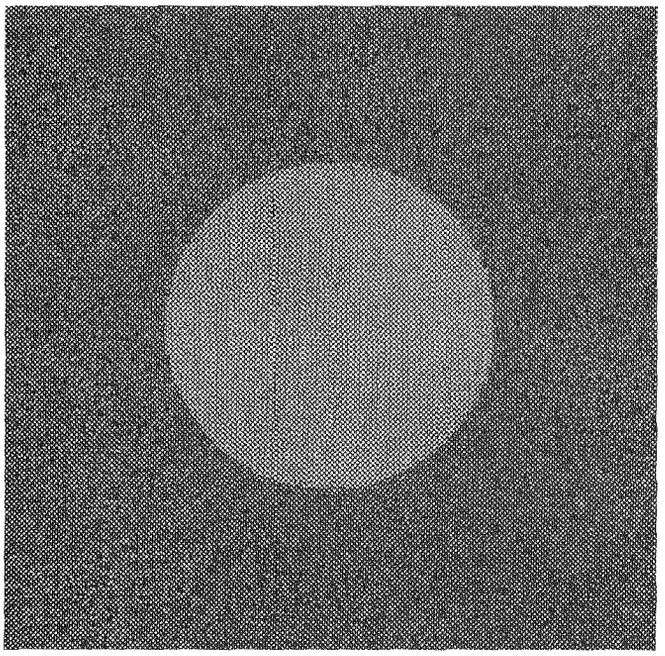


Figure 11. – Image du disque bruité.

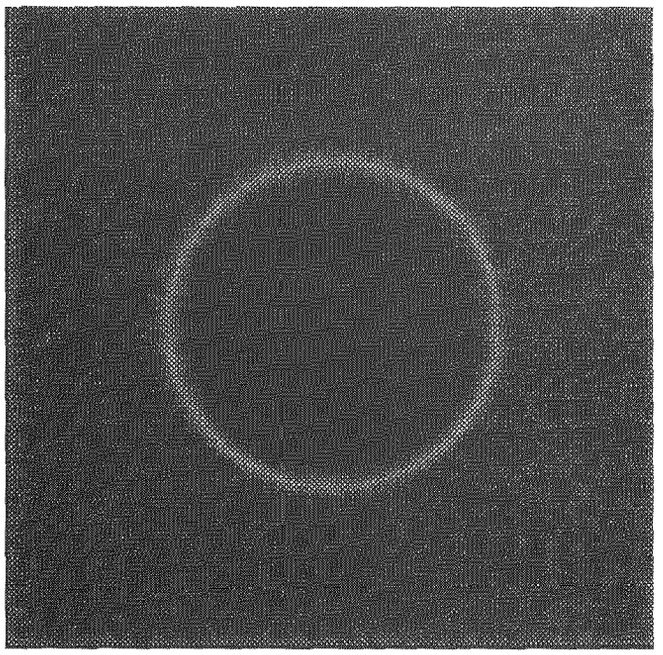


Figure 12. – Contours détectés avec un codage des coefficients et des calculs internes sur 12 bits et $s = 2$.

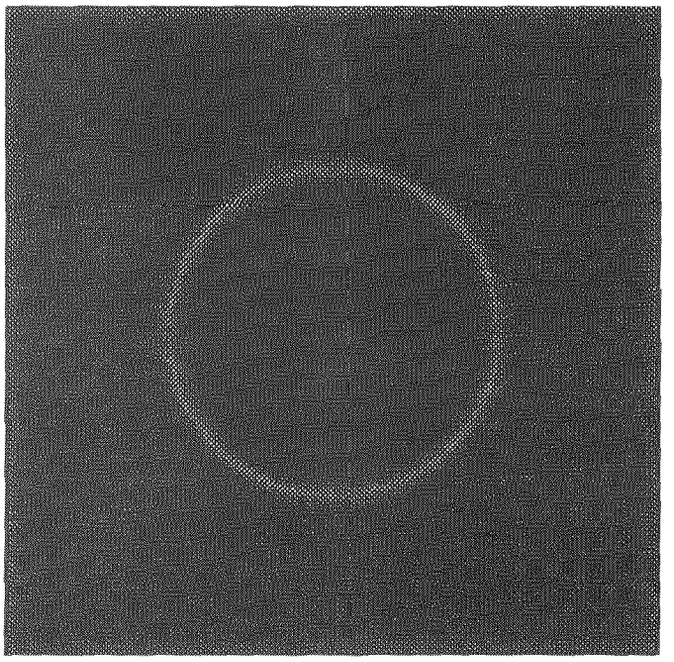


Figure 13. – Contours détectés avec $s = 2$ et les calculs en réel sur 64 bits.

5. implantation du filtre obtenu par anticipation sous forme d'un ASIC

Nous présentons dans cette partie l'architecture et la réalisation d'un circuit intégré spécifique (ASIC) pour la détection de contours sous forme de rampe. L'accélération des calculs est obtenue grâce à l'anticipation des calculs (voir section 3). La conception est faite de telle sorte que quatre circuits identiques permettent d'obtenir le gradient horizontal et le gradient vertical.
Nous décrivons à partir d'un schéma bloc initial, comment nous avons pu réduire la mémoire interne et synchroniser les traitements de gauche à droite et de droite à gauche pour les lignes, du haut vers le bas et du bas vers le haut pour les colonnes.
Soient X^+ les résultats obtenus par le filtrage en balayant les lignes de gauche à droite, X^- en les balayant de droite à gauche. Y^+ les résultats obtenus à l'issue du filtrage en balayant les colonnes de haut en bas, Y^- en balayant les colonnes de bas en haut. Ces résultats sont calculés en parallèle à l'aide de quatre circuits identiques. Ils effectuent les traitements sur une même image de manière parallèle et synchrone. Le synoptique général de l'opérateur d'extraction de contours est donné en figure 14.

L'idée générale (simplifiée) de fonctionnement d'un tel circuit est :

1 – Chargement des coefficients.
2 – Chargement d'une ligne (colonne) de l'image dans la RAM 512 × 8 bits.
3 – Traitement de la ligne (colonne) suivant X^+ et X^- (Y^+ et Y^-). Le filtre est commandé par une unité de contrôle.
4 – Les résultats du traitement de chaque filtre sont stockés dans une RAM de sortie (512 × 12 bits).

5 – L'addition de $X^+[i]$ et $X^-[i]$ ($Y^+[i]$ et $Y^-[i]$) est effectuée d'une façon très rapide (par rapport à la fréquence d'itération du filtre) et ceci dès que les deux termes nécessaires à l'addition sont disponibles.

Le synoptique présenté en figure 14 montre que l'on peut garantir le temps réel vidéo mais avec une latence de une image ce qui est illustré par la présence des RAMs « Image » en entrée.

5.1. architecture à une seule RAM

Notre objectif est donc de réaliser un seul circuit avec le choix de deux modes de fonctionnement : X^+ ou X^-.

Pour cela nous avons développé une architecture à base de multiplexeurs (voir figure 16). Quatre circuits identiques sont nécessaires pour l'obtention du gradient vertical et horizontal. Si « c » est égal à zéro, le filtre fonctionne en X^+, l'entrée vidéo (codée sur 8 bits) est alors sélectionnée. Les données sont filtrées puis mémorisées par la FIFO. Si « c » est égal à un, l'entrée vidéo est alors aiguillée vers la LIFO, puis filtrée afin d'obtenir le traitement suivant X^-.

5.2. schéma bloc de l'ASIC

La figure 15 montre les principaux blocs fonctionnels qui forment le cœur du circuit ainsi que les différents signaux d'entrée/sortie et les signaux de commande indispensables au fonctionnement du circuit. Les différents blocs sont :

– Le filtre.

– L'unité de contrôle.

– Les registres des coefficients.

– Les multiplexeurs d'entrée (MUX1 et MUX2) et de sortie (MUX3).

– La mémoire LIFO.

– Le compteur d'adresses.

Tous les coefficients du filtre sont de 12 bits et sont représentés en complément à deux. Les coefficients sont chargés de l'extérieur par l'intermédiaire du bus de données « Entrée-vidéo ».

Comme le bus de données est sur un octet, pour charger 12 bits deux cycles d'horloge sont nécessaires (nous chargeons en premier les 8 bits de poids forts d'une façon parallèle et au coup d'horloge suivant nous chargeons les quatre bits de poids faibles). Le filtre anticipé (équ.13) possède 11 coefficients, leur programmation requiert donc 22 cycles d'horloge. L'unité de contrôle gère l'arrivée et le stockage des coefficients.

Nous avons vu (équ.13) que le filtre à implanter est récursif d'ordre 12. Son équation aux différences renferme 11 coefficients. Chaque coefficient aboutit à l'une des deux entrées du multiplieur associé. Au total, le graphe de fluence du filtre possède 11 multiplieurs. Trois d'entre eux sont dans la partie récursive et huit dans la partie non récursive. La structure de ces multiplieurs est basée sur l'algorithme de Baugh-Wooley[11].

Le choix de ce multiplieur est justifié par le fait que nous cherchons une architecture pouvant être pipelinée. Celle-ci est de type à sauvegarde de retenue CS (Carry-Save) [12], [13].

Dans le but de réduire les temps d'addition des produits partiels Wallace [14] et Dadda [5] ont proposé des techniques de réarrangement des différents additionneurs effectuant l'addition des produits partiels.

Le multiplieur de Baugh-Wooley [11] permet de générer, dans une structure très régulière, des produits partiels pour des produits en complément à deux (C2) ou pour des produits mixtes (non signés multipliés par C2).

Les principaux blocs du filtre sont :

– Les éléments mémoires.

– Les multiplieurs.

– Les blocs de contrôle pour l'initialisation.

– Les additionneurs.

Le bloc filtre reçoit, en plus des données à traiter, divers signaux de commande :

– Horloge de fonctionnement du filtre.

– Les signaux d'initialisation.

– Les 11 coefficients.

5.3. conception des masques

Le circuit a été conçu avec le logiciel de CAO Cadence, en utilisant la bibliothèque de cellules pré-caractérisées CMOS 1.2 μm de la société ES2. Les caractéristiques du détecteur de contours par filtrage récursif d'ordre 3 anticipé sont les suivantes :

– Fréquence de fonctionnement (post-layout) : 20 MHz.(les multiplieurs fonctionnent jusqu'à 50 MHz).

– Il traite des images de 512 × 512 pixels.

– Surface : 60 mm^2.

– Technologie CMOS 1.2 μm.

– Boîtier : 40 broches.

– Codage des pixels : 8 bits

– Toute l'arithmétique interne est sur 12 bits.

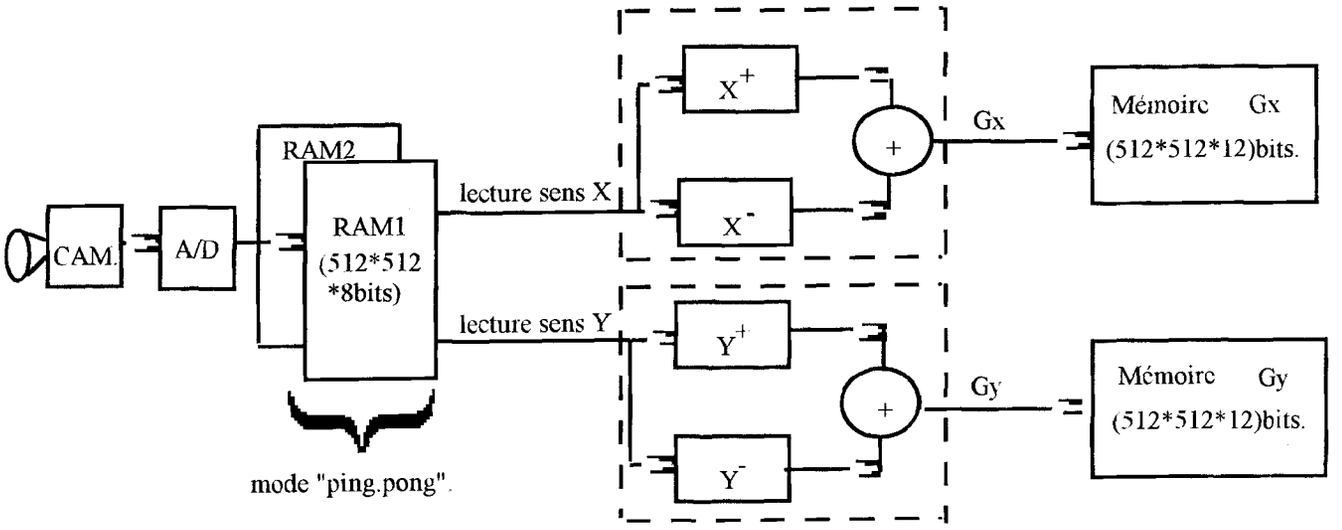
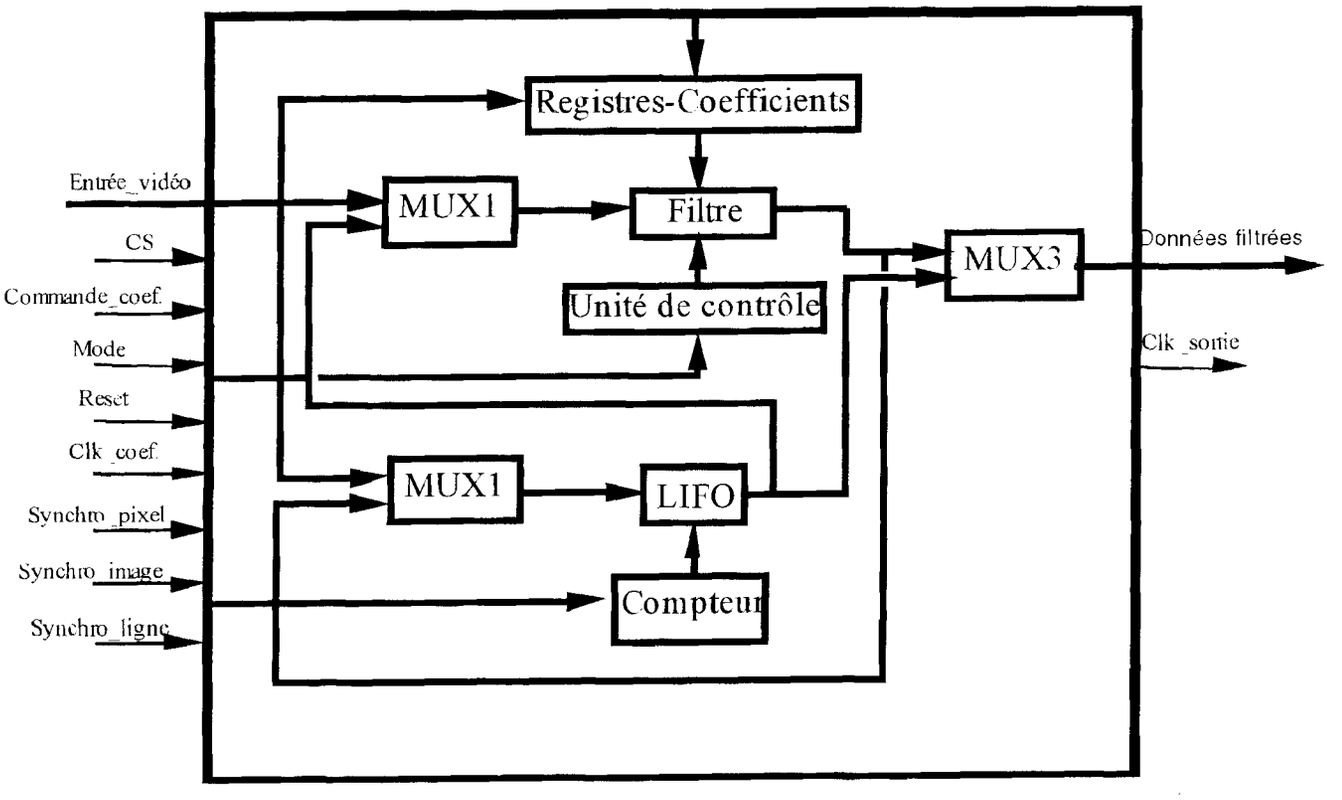


Figure 14. – Schéma synoptique de la carte complète.



Le circuit est piloté par les différents signaux : Entrée-vidéo (Le bus d'entrée), Synchro_ pixels (Horloge pixel), Synchro_ligne (indique l'arrivée d'une nouvelle ligne), Synchro_image (indique l'arrivée d'une nouvelles image), Clk_coef. (Horloge de changement des coefficients), Charge_coef : permet le chargement des coefficients à travers le bus d'entrée, mode : Selon la valeur de cette entrée le circuit fonctionne en X^+ (Mode = 0) ou X^- (Mode = 1), Reset : initialisation du circuit, CS : Permet la sélection du "chip".

Figure 15. – Schéma fonctionnel du circuit.

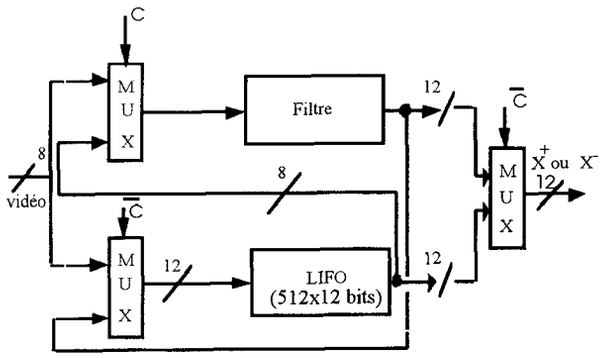


Figure 16. – Contour sous forme exponentielle avec $s = 0,5$.

Les résultats présentés ici confirment bien la complexité de conception pour réaliser un filtre récursif traitant un pixel à une fréquence supérieure à 10 Mhz. Le gradient ($Gx = X^+ + X^-$) ne peut être obtenu que par la mise en parallèle de deux circuits de ce type. De ce fait, nous proposons une autre alternative pour améliorer le compromis temps/surface. Cette méthode consiste à implanter le filtre directement sans anticipation de calcul, mais en redistribuant les registres dans la partie récursive du filtre (réseau de convolution récursive d'efficacité 1) [8].

6. amélioration de l'implantation directe du filtre

Cette approche consiste à implanter le filtre $f(x)$ directement, en réalisant une redistribution locale des registres sur la partie récursive, comme cela est représenté sur la figure 17 (b).
La fréquence de fonctionnement est alors donnée par le temps de calcul d'une multiplication et d'une addition. Le gain en temps correspond à l'élimination de trois additionneurs 12 bits, soit un gain de l'ordre de 30%. Le choix pour la conception finale du processeur dédié s'est porté sur cette architecture.

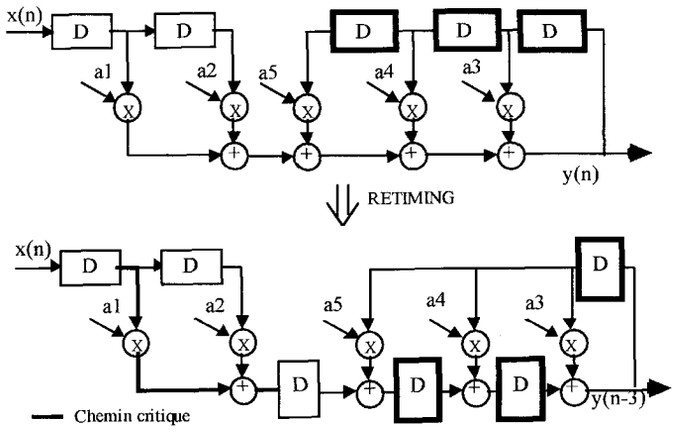


Figure 17a et 17b. – Redistribution locale des registres.

6.1. architecture

L'implantation de cet ASIC a été réalisée à l'aide du logiciel OPUS (Cadence Design Framework II) avec une technologie CMOS 1 μm.

L'architecture proposée est présentée en figure 18. Les principaux blocs utilisés sont : le filtre causal et anti-causal, un micro-contrôleur permettant de gérer les coefficients des filtres, et une partie de mémorisation des coefficients. A chaque filtre nous avons attribué un bus vidéo permettant ainsi de les synchroniser (X^+ et X^-). La sortie des filtres est donnée par Y^+ et Y^-. L'opération du gradient (Gx) se faisant à l'extérieur du processeur.

Les Filtres

Ce circuit intègre le filtre causal et anti-causal. Chacun de ces filtres est composé de cinq multiplieurs 12 × 12 bits, de quatre additionneurs 12 bits, ainsi que de six registres 12 bits (voir figure 17(b)). Seuls les signes des coefficients a1 et a2 sont opposés entre le filtre causal et anti-causal, ce qui justifie la nécessité d'insérer une logique supplémentaire permettant de réaliser le changement de signe de ces coefficients (bloc « signe-coeff » sur la figure 18).

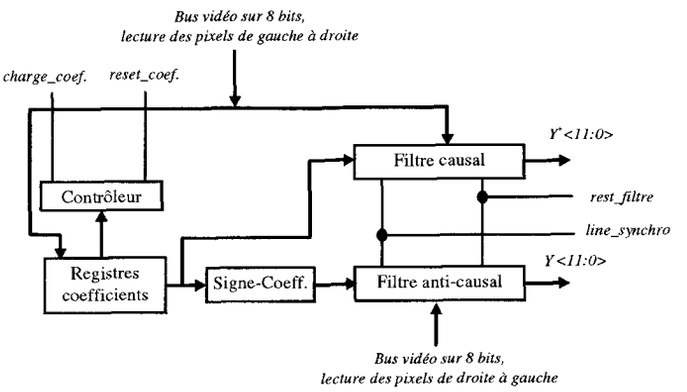


Figure 18. – Architecture de l'ASIC.

La conception en cellules pré-caractérisées nous semble suffisante pour la réalisation de cet ASIC. Cependant il est indispensable de déterminer le couple multiplieur additionneur permettant d'obtenir le meilleur compromis vitesse/surface. Nous avons choisi comme algorithme de multiplication celui de Booth [16].

Le multiplieur ainsi généré a des performances convenables (moins de 20 ns pour une multiplication 12 × 12 bits pour une surface inférieure à 1 mm^2). Le choix de l'additionneur quant à lui s'est porté sur une architecture du type « carry-select » offrant le meilleur compromis vitesse/surface (une addition de deux mots de 12 bits en 5 ns et une surface inférieure à 0.1 mm^2). En cellules pré-caractérisées l'opération de multiplication et d'addition est donc réalisable en moins de 25 ns.

Les Coefficients

Tous les coefficients sont codés sur 12 bits en complément à deux, et sont chargés en série par le bus vidéo en 10 cycles d'horloge dans des registres statiques de 12 bits. Le canal vidéo X^+ est utilisé pour cette manipulation. Le canal vidéo étant sur 8 bits, un coefficient sera donc mémorisé en deux cycles d'horloges (un premier mot de 8 bits correspondant aux poids forts puis un mot de 4 bits correspondant aux poids faibles). Une fois ces coefficients chargés, ils sont alors disponibles à l'entrée du multiplieur correspondant.

Le Test

Le mode test consiste à sortir en série, à un instant donné toutes les valeurs des registres appartenant aux filtres. Lorsque le circuit est en mode test celui-ci configure les regitres de manière à former un chaînage des toutes les bascules des filtres.

6.2. résultats

Les prototypes réalisés avec cette approche, dans une technologie CMOS 1 μm, ont été testés avec succès. En effet le processeur dédié ainsi obtenu (figure 19) est capable de traiter un pixel en 30 ns, avec des images de taille variable (64 $\times$ 64 à 1024 $\times$ 1024 pixels). La complexité du circuit est de 72 000 transistors pour une surface de 29 mm^2 (5.6 $\times$ 5.3 mm^2).
Il est nécessaire de noter que, nous n'avons pas ici intégré les LIFOs, celles–ci étant à l'extérieur, pour des raisons d'économie de surface de silicium. Cependant il faut noter, d'une part l'amélioration sensible de la vitesse de fonctionnement du circuit, et d'autre part avec deux circuits identiques nous avons la possibilité d'obtenir le gradient vertical et horizontal en parallèle.

6.3. comparaison des différentes méthodes exposées

Nous avons proposé différentes solutions pour l'intégration du filtre récursif d'ordre 3 de Canny-Deriche optimisé. Dans la section 6 nous avons décrit l'implantation directe de ce filtre. Dans la section 3, nous avons développé une méthode adéquate à l'accélération de ce type de filtre. Dans la section précédente nous avons vu comment une modification simple de l'architecture du filtre entraîne une augmentation sensible des performances. Le tableau 2 compare les performances de ces différentes implantations. Pour réaliser une comparaison objective nous avons intégré en technologie CMOS l'ensemble de ces solutions.
Selon l'application visée, cette étude apporte différentes solutions d'intégration. Dans tous les cas il est envisageable de traiter des images de dimension supérieure à 512 $\times$ 512 pixels.

Tableau 2. – Comparaisons entre les différentes méthodes d'implantations du filtre de Canny-Deriche optimisé (technologie CMOS 1μ m ES2)

Implantations	Performances estimées avant routage		Performances réelles après routage	
	Surface (mm^2)	F (Mhz)	Surface (mm^2)	F (Mhz)
Directe	5.2	25	7.4	22
Redistribution globale des registres	15.3	71	24.5	66.5
Redistribution locale des registres	5.3	40	7.5	37.5

Performances estimées : simulation réalisée en estimant les capacités d'interconnexion.

Performances réelles (« post layout ») : simulations réalisées après extraction électriques en tenant compte des capacités d'interconnexion.

La solution du filtre anticipé (redistribution globale des registres) permet d'atteindre des fréquences pixels de 66.5 Mhz, soit trois fois plus que pour l'implantation du filtre direct, avec un coût en surface de silicium non négligeable puisque celui-ci est de l'ordre de trois. La méthode de redistribution locale des registres proposée représente un bon compromis temps/surface. En effet le coût en surface est minime (surface quasiment identique avec l'implantation du filtre direct), avec une fréquence pixel de 37.5 Mhz ce qui nous permet d'envisager de traiter des images de dimension 1024 $\times$ 1024 pixels en temps réel.

7. conclusion

L'accélération des filtres numériques récursifs est une étape nécessaire pour l'implantation d'algorithmes en temps réel.

L'étude de l'adéquation algorithme architecture silicium pour un filtre récursif d'ordre 3, nous a permis de passer en revue différentes techniques d'accélération de filtres récursifs.

Nous avons ainsi mis en évidence deux techniques de conception de filtres récursifs.

Le calcul par anticipation et ajout de pôles dispersés a débouché sur un premier processeur dédié. La seconde méthode plus directe nous a permis d'améliorer sensiblement le compromis temps/surface, qui est un facteur important lors de la conception.

Cette étude a débouché sur un processeur dédié à haut débit. Les résultats satisfaisants obtenus nous permettent de développer actuellement une carte de traitement d'images autour de ce processeur.

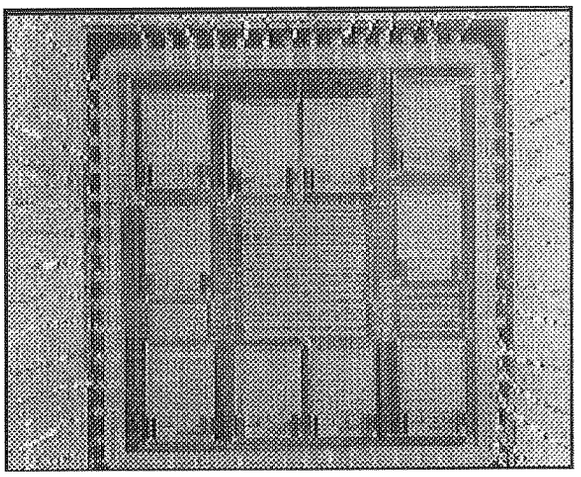


Figure 19. – Photographie du processeur dédié. Opérateur de détection de contours en temps réel (à partir de l'algorithme de Canny-Deriche optimisé) à une fréquence de 33 Mhz pour une surface de 29 mm^2 (72 000 transistors), précision interne 12 bits.

Remerciements

Nous tenons à remercier G. Cathebras, R. Lorival (Université de Montpellier II - Laboratoire LIRMM), C. Milan et J.P. Zimmer (Université de Bourgogne - Laboratoire LIESIB) pour leurs participations à ce projet.

ANNEXE

Calcul des séquences $q_i(k)$

Soit la partie récursive d'un filtre d'ordre N avec k latches de pipeline dans sa boucle de retour décrite par :

$$H(Z) = \frac{1}{1 - \sum_{i=1}^{N} q_i(k)\, Z^{-ik}}$$

La fonction de transfert originale correspond à $k = 1$ et $q_i(1) = a_i$. On peut obtenir un filtre équivalent à $2k$ étages de pipeline en multipliant le numérateur et le dénominateur par

$$\left(1 - \sum_{i=1}^{N} (-1)^i\, q_i(k)\, z^{-ik}\right)$$

D'où

$$H(Z) = \frac{1 - \sum_{i=0}^{N} (-1)^i\, q_i(k)\, z^{-ik}}{1 - \sum_{i=1}^{N} q_i(2k)\, Z^{-2ik}}$$

Avec les $q_i(2k)$ reliés aux séquences $q_i(k)$ par les relations suivantes :

$$q_1(2k) = q_1^2(k) + 2q_2(k)$$
$$q_N(2k) = (-1)^{N+1} q_N^2(k)$$

Pour un filtre d'ordre N pair :

$$q_i(2k) = \begin{cases} 2q_{2i}(k) + (-1)^{i+1} q_i^2(k) + 2\sum_{j=1}^{i-1} (-1)^{j+1} q_j(k) q_{2i-j}(k), \\ \qquad i = 2, \ldots, \frac{N}{2}. \\ (-1)^{i+1} q_i^2(k) + 2\sum_{j=i+1}^{N} (-1)^{j+1} q_j(k) q_{2i-j}(k), \\ \qquad i = \frac{N}{2} + 1, \ldots, N-1. \end{cases}$$

Pour un filtre d'ordre N impair :

$$q_i(2k) = \begin{cases} 2q_{2i}(k) + (-1)^{i+1} q_i^2(k) + 2\sum_{j=1}^{i-1} (-1)^{j+1} q_j(k) q_{2i-j}(k), \\ \qquad i = 2, \ldots, \frac{N-1}{2}. \\ (-1)^{i+1} q_i^2(k) + 2\sum_{j=1}^{i-1} (-1)^{j+1} q_j(k) q_{2i-j}(k), \\ \qquad i = \frac{N+1}{2} \\ (-1)^{i+1} q_i^2(k) + 2\sum_{j=i+1}^{N} (-1)^{j+1} q_j(k) q_{2i-j}(k), \\ \qquad i = \frac{N+3}{2}, \ldots, N-1. \end{cases}$$

BIBLIOGRAPHIE

[1] E. Bourennane, « Conception et implantation d'un détecteur de contours optimisé sous forme d'un circuit ASIC », *Thèse de doctorat*, Université de Dijon, Février 1994.

[2] E.Bourennane, M. Paindavoine et F. Truchetet, « Amélioration du filtre de Canny Deriche pour la détection de contours sous forme de rampe », *Revue Traitement du signal*, Volume 10, N°4, 1993.

[3] T. Kamle, « Implantation d'algorithmes de traitement de signaux bidimensionnels en flots de données sur ASICS et circuits reconfigurables », *Thèse de Doctorat*, Décembre 1994, Université de Paris Sud centre d'Orsay.

[4] C. Leiserson, F. Rose, and J. Saxe, « Optimizing synchronous circuitry by retiming », in *Proc. 3 rd Caltech conf VLSI*, Pasadena, CA, MAR,1983.

[5] A. Fettweis, *Realizability og digital filter networks*, Second Edition, Addison–Wesley Publishing Company, 1987.

[6] M. Bellanger, *Traitement Numérique du Signal*, collection CNET-ENST, Ed. Masson,1987.

[7] R. Boite, H. Leich, *Les filtres numériques-Analyse et synthèse des filtres unidimensionnels*, collection CNET-ENST, Ed. Masson, 1990.

[8] N. Zarka, « Conception d'un circuit intégré de détection optimale de contours », *Thèse de doctorat de l'université de PARIS 6*, Déc 1992.

[9] Sarifuddin, « Implantation sous forme d'un circuit spécialisé d'un algorithme de détection de contours multi–échelles », *Thèse de doctorat*, Université de Dijon, Juillet 1995.

[10] R. Deriche, « Using criteria to derive a recursively optimal edge detector », *Int. Journal of Comp. Vision*, 1987, pp. 167–187.

[11] C.R. Baugh, B.A. Wooley, « A two's complement parallel array multiplication algorithm », *IEEE Transactions on Computers*, Vol. C–22, N°12, Dec 1973.

[12] N.H.E Weste, Keshraghian, *Principles of CMOS VLSI design*, Second Edition, Addison–Wesley Publshing Company.

[13] J. Chinal, *Circuits magiques de traitement numérique de l'information*, Cepadues Editions.

[14] C.S. Wallace, « A suggestion for a fast multiplier », *IEEE Transactions on Electronics Computer*, Feb 1964.

[15] L. Dadda, « Some schemes for parallel multipliers », *Alta frequenza*, vol. 34, N°5, May 1987, pp. 167-187.

[16] A.D. Booth, « A signed binary multiplication technique », *Quart. Journ. Mech and Applied Math*, vol. IV Pt 2, 1951.

Manuscrit reçu le 22 Novembre 1995.

LES AUTEURS

Lionel TORRES

Lionel Torres, 26 ans, a obtenu le diplôme de docteur en micro–électronique en 1996, à l'Université de Montpellier 2, au sein du LIRMM. Ses domaines de recherche concernent l'intégration d'architectures reconfigurables dédiées au traitement du signal et de l'image. Il est actuellement en charge du développement et de la mise en place de méthodologies de conception de cœurs de micro-processeur au sein de la société ATMEL-ES2.

El-Bay BOURENNANE

El-Bay Bourennane, 30 ans, a obtenu le diplôme de docteur en automatique et traitement d'images en 1994, à l'Université de Bourgogne, au sein du laboratoire GERE. Il est actuellement Maître de Conférences à l'I.U.T. de Dijon. Ses travaux de recherche portent essentiellement sur l'Adéquation-Algorithmes-Architecture en traitement d'images en vue d'implantation sous forme d'ASICs.

Michel ROBERT

Michel Robert, 39 ans, Membre de l'Institut Universitaire de France, est Professeur à l'Université Montpellier 2. Il enseigne la CAO des circuits intégrés à l'ISIM (Institut des Sciences de l'Ingénieur). Il effectue ses travaux de recherche au LIRMM (Laboratoire d'Informatique, de Robotique et de Micro–électronique) dans le domaine de la synthèse électrique et topologique des circuits intégrés, l'analyse des performances des circuits CMOS, et l'Adéquation Algorithmes–Architectures–Silicium en traitement d'images temps réel.

Michel PAINDAVOINE

Michel Paindavoine, 40 ans, est Professeur à l'Université de Bourgogne. Il enseigne le traitement du Signal et des Images à l'Ecole d'Ingénieurs FIRST et à l'IUP – Génie Industriel (Electronique et Image). Il effectue ses travaux de recherche au LE2I (Laboratoire d'Electronique, d'Informatique et d'Image) dans le domaine de l'Adéquation Algorithmes-Architectures en traitement d'images.