Real-Time Imaging **4**, 181–191 (1998)

A Recursive Digital Filter Implementation for Noisy and Blurred Images

This paper presents the design and the implementation of an optimized Canny-Deriche edge detector. After a brief reminder of the filter's equations, we define different techniques to speed up the sampling rate of the IIR filter. In particular, improving the throughput rate of the IIR filter, we present a look-ahead with decomposition technique. This method leads us to design a first chip, which performs at a sampling rate of over 20 MHz 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 30 ns with a silicon area of 30 mm^2. These two approaches are compared. This work leads us to an ASIC which was designed in a CMOS 1 μm technology and successfully tested.

L. Torres[1], **E. Bourennane**[2], **M. Robert**[1] **and M. Paindavoine**[2]

(GDR ISIS members)

[1]*Laboratoire LIRMM, UMR 9928 CNRS, Université de Montpellier II 161 rue Ada – 34392 Montpellier Cedex 5, France*

[2]*Université de Bourgogne, Laboratoire LE2I, 6, bd Gabriel, 21000 Dijon, France*

Introduction

A part of low level image processing consists of the detection (by filtering) of sharp level variations of luminance in the image to obtain the edge segments, extremities and angles. To select the geometrical information related to the structure of the objects present in an image, a key step consists of extracting the edges of the considered objects. Several accurate numerical algorithms have been proposed in this field: Roberts [1], Prewitt [2], Canny [3], and Deriche [4] have already been implemented in software or with dedicated hardwares for several applications [1,5,6].

One of the main differences between these algorithms is the calculation complexity of differential operators between neighbor pixels: from 2×2 to more than 9×9 per kernel. The choice of an edge detection algorithm depends mainly on the signal to noise ratio (robustness), the localization, and the quality of detection and the implementation efficiency. One problem is the necessary CPU time required to perform the edge extraction.

We present here a very high speed edge detector chip using the optimized CannyDeriche IIR filter in a single chip.

This paper is organized as follows. The following section is devoted to the edge model and the optimized Canny-Deriche filter. Then different digital filter integration examples are shown and different techniques are defined to accelerate calculations in filtering. The calculation is

1077-2014/98/030181 + 11 $25.00/ri970094

detailed by using a look-ahead with decomposition technique well adapted to the acceleration of recursive filters, and this technique is applied to the optimized Canny-Deriche filter [7].

In the penultimate section the integration part of the filter obtained by look-ahead calculation in an ASIC is shown, and in the last section the direct implementation of the filter without look-ahead technique is shown, but with a local retiming of internal registers contained in the recursive part of the filter.

The Filter

The edge model presented here is the best approximation for a wide variety of edges seen in real-time image processing [7]. Even if the edge at the input of a camera is a step, it is transformed at the output in an edge which we assume to follow this model (exponential edge model):

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

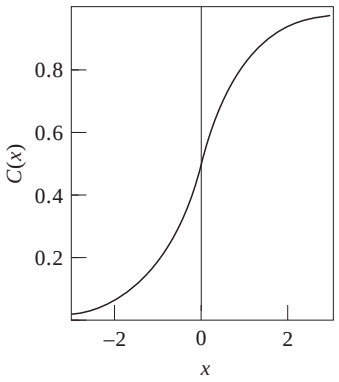

Figure 1. Exponential edge model with $s = 0.5$.

Chosen criteria for this algorithm are the following:

(i) signal/noise ratio: where the noise is considered white, additional, gaussian and centered in zero.
(ii) localization: the average gap between edge detected with noise position and its real position.
(iii) quality of the detection (spurious response): with noise, the signal detected can present several maxima, and thus parasitic responses must be as far as possible from the real response.

These three criteria are maximized using a mathematical approach.

The optimal blurred edge detector impulse response is thus given by (optimal edge detector impulse response) [7]:

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

This leads to a recursive realization with two stable third recursive filters moving in opposite directions. It can be extended to the 2D case by knowing that the slope of a surface in any direction can be determined exactly from its slope in two directions:

$$\begin{aligned} y^{+}(i) &= a_1 x(i-1) + a_2 x(i-2) \\ &\quad a_3 y^{+}(i-1) - a_4 y^{+}(1-2) + a_5 y^{+}(i-3) \\ y^{-}(i) &= a_1 x(i-1) + a_2 x(i+2) \\ &\quad + a_3 y^{-}(i-1) - a_4 y^{-}(i+2) + a_5 y^{-}(i+3) \end{aligned} \tag{3}$$

with:

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

We have shown [7] that for a good choice of the parameters *m* and *w*, and in the case of blurred and noisy images, our operator allows a notable increase of the signal to noise ratio in comparison with the Deriche filter (for example with a signal/noise ratio equal to 2, we increase the performance by 20%).

The recursive filter implementation allows us to avoid the truncation of the filter impulse response, and the use of this filter in the case of multiscale application is easier because the number of operations per pixel is independent of the scale parameters.

Filter's Implementations

We describe, in the following sections, the filter's implementations in an ASIC.

Direct implementation of the optimized Canny–Deriche filter

A solution to achieve real-time image processing consists of integrating the filter in a specific dedicated processor (ASIC).

The Z transform of the recursive filter given by equations (2) are the following:

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

The corresponding architecture is given in Figure 2.

The integration of such an architecture creates some problems. Observe that the critical path is composed by one multiplier and four adders and constitutes a complex combinational chain. Morever, the number of glitches increases when signals through each adder.

The integration of this filter with the help of the CAD tool OPUS (Cadence Design Framework II) in a CMOS 1 μm technology has shown its limitations. The frequency of processing is in the order of 25 Mhz (for a silicon area 5 mm^2). If our main goal is the integration of an edge detector operator in real time for images larger than 512 × 512 pixels, this architecture will not work correctly.

For fast real-time throughput applications, the various methods that allow accelerated calculations become more impotant.

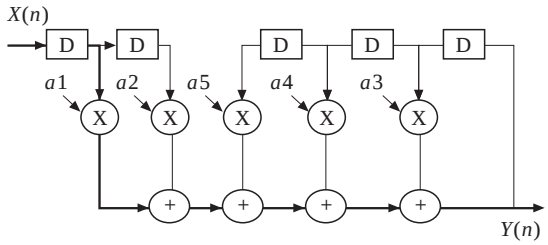

Figure 2. TZ$^+$ direct implementation.
(—) Critical path. D Register;
X Multiplier; + adder.

Filter implementation with calculus acceleration

Digital filtering occupies an important place in the field of the digital signal and image processing. To improve the performances of these filters for real-time applications. the traditional approach consists of increasing the hardware component speed or to program a dedicated processor such as DSP (Digital Signal Processor). The parallel system implementation in dedicated circuits (ASIC), allows higher frequencies to be reached. However, the class of applications of such circuits remains limited.

In the following subsection a solution is given that accelerates the calculations of the filter $f(x)$.

Sampling period definition

A digital filter is characterized by its minimal sampling period imposed by its implementation (Figure 3):

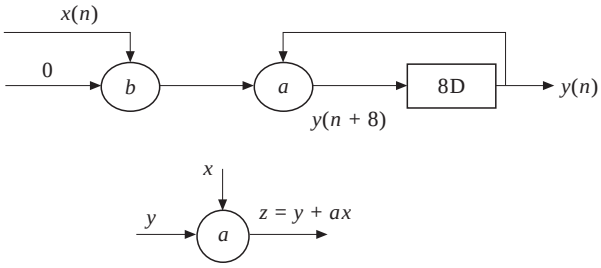

Figure 3. Pipelined IIR filter.

The sampling period of an IIR filter is limited by its recursive part. It is given by:

$$T_{\infty} = \frac{1}{L} \mathrm{MAX}\left[\frac{D_1}{M_1}\right], \quad 1 \in S_1 \quad (5)$$

The *loop l* element of Sl that verifies this equation is called the *critical loop*. *Sl* = the feedback loop in the graph (in this graph, there is a single feedback loop); Dl = time of calculation associated with each loop (here $T_a + T_m$, where T_a is the time of an addition operation, and T_m is the time of a multiplication operation); M_l represents the total number of delays (latches) in the loop l (here $l = 1$ and $M_l = 8$); L represents the ratio between the frequency input samples and the internal registers to the graph. Here, $L = 1$.

The iteration period of the graph above-mentioned is:

$$T_{\infty} = \left(T_a + T_m\right)/8$$

Scattered look-ahead with power-of-two-decomposition technique

Given the recursive part of an Nth order filter with k pipeline stages in its feedback loop,

$$H(z)=\frac{1}{1-\sum_{i=1}^{N} q_i(k)z^{-ik}} \tag{6}$$

The original transfer function corresponds to $k = 1$, where $q_i(1) = a_i$. The equivalent filter with $2k$ pipeline stages can be obtained by multiplying the numerator and the denominator by:

$$\left(1-\sum_{i=1}^{N}(-1)^i q_i(k)z^{-ik}\right) \tag{7}$$

The pipelined filter is described by:

$$H(z)=\frac{1-\sum_{i=0}^{N}(-1)^i q_i(k)z^{-ik}}{\left(1-\sum_{i=1}^{N} q_i(2k)z^{-2ik}\right)} \tag{8}$$

where the sequence $q_i(2k)$ is derived in terms of the sequences $q_i(k)$ (depending on the Nth order filter being even or odd) by relationships given in annex.

The application of this transformation a second time gives a pipelined filter four times, (then eight times, and so on). From the original function, it is necessary to apply log2 (M) transformations to have M pipeline levels. Each transformation increases the speed by a factor of 2 and brings N supplementary multipliers to the filter structure.

In general, the transfer function of an M level pipeline is given by:

$$H(z)=\frac{\left(\sum_{i=0}^{N} b_i z^{-i}\right)\prod_{k=0}^{\log_2(M)-1}\left[1-\sum_{i=1}^{N}(-1)^i q_1\left(2^k\right)z^{-i2^k}\right]}{\left(1-\sum_{i=1}^{N} q_i(M)z^{-iM}\right)} \tag{9}$$

This transfer function can be implemented with ($2N$ + Nlog2(M) + 1) multiplications.

Application to the f(x) *filter*

The described acceleration method mentioned above is valid for any order of the filter used whose Z-transform is:

$$\begin{aligned} TZ^{+} &= \frac{Y^{+}(z)}{X(z)} = \frac{a_1 z^{-1}+a_2 z^{-2}}{1-a_3 z^{-1}+a_4 z^{-2}-a_5 z^{-3}} \\ TZ^{-} &= \frac{Y^{-}(z)}{X(z)} = -\frac{a_1 z+a_2 z^{2}}{1-a_3 z+a_4 z^{2}-a_5 z^{3}} \end{aligned} \tag{10}$$

with $X(z) = TZ(x)$.

The transfer function of this filter TZ^{+} (TZ^{-}) is transformed by four pipeline levels (limited by the capacity of integration in an ASIC design). The filter resulting from the transformation is a 12th order filter whose transfer function H (z) is given by Eqn 11.

This transfer function can be seen as a product of a four transfer functions. These transfer functions are implemented in a cascade form, as described in Figure 4.

Figure 4. $H(z)$ cascade implementation.

$$H(z)=\frac{\left(a_1 z^{-1}+a_2 z^{-2}\right)\left(1+a_3 z^{-1}+a_4 z^{-2}+a_5 z^{-3}\right)\left[1+\left(a_3^2-2a_4\right)z^{-2}-\left(-a_4^2+2a_3a_5\right)z^{-4}+a_5^2 z^{-6}\right]}{1-\left[\left(a_3^2-2a_4\right)^2+2\left(-a_4^2+2a_3a_5\right)z^{-4}+\left(-\left(-a_4^2+2a_3a_5\right)^2+2a_5^2\left(a_3^2-2a_4\right)\right)z^{-8}+a_5^4 z^{-12}\right]} \tag{11}$$

with:

$$\begin{aligned} H_1(z) &= a_1 z^{-1}+a_2 z^{-2} \\ H_2(z) &= 1+a_3 z^{-1}+a_4 z^{-2}+a_5 z^{-3} \\ H_3(z) &= 1+\left(a_3^2-2a_4\right)z^{-2}-\left(-a_4^2+2a_3a_5\right)z^{-4}+a_5^2 z^{-6} \\ H_4(z) &= \frac{1}{1-\left[\left(\left(a_3^2-2a_4\right)^2+2\left(-a_4^2+2a_3a_5\right)\right)z^{-4}+\left(-\left(-a_4^2+2a_3a_5\right)^2+2a_5^2\left(a_3^2-2a_4\right)\right)z^{-8}+a_5^4 z^{-12}\right]} \end{aligned} \tag{12}$$

The corresponding graph filter is the following:

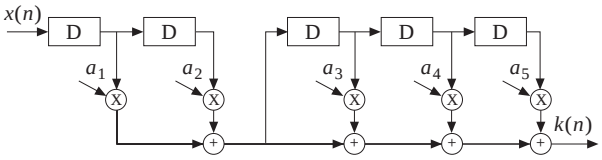

Figure 5. $H_1(z)$ and $H_2(z)$ cascade implementation.

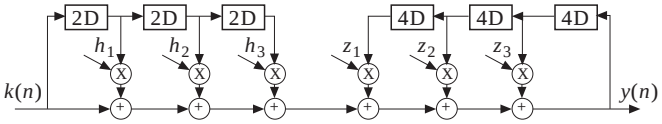

Figure 6. $H_3(z)$ and $H_4(z)$ cascade implementation. Coefficients a_i (i = 1,..., 5), h_j and z_j (j = 1,2,3) are directly deduced from $H(z)$.

Global register retiming

Supplementary registers obtained from the look-ahead calculation have an interest only if they are suitably exploited. Retiming [8,9], is a technique which allows delays to be moved in a data flow graph and exploited better. By using this principle, registers of the $H(z)$ filter feedback-loop are retimed as shown in Figure 7(a). In Figure 7(b), $e(n)$ is the output of the non-recursive part of the look ahead filter.

The registers are moved inside each multiplier to obtain a pipeline level. This operation also allows a pipeline structure for adders between them. The non-recursive part can use these multipliers without introducing modification to the transfer function, but will introduce a timing latency of the output $y(n)$.

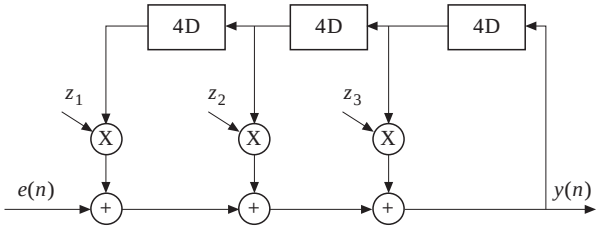

Figure 7(a). $H(z)$ feedback loop.

After showing the different techniques to accelerate recursive filters, the look-ahead acceleration is a method well adapted in view of filter integration in an ASIC implementation.

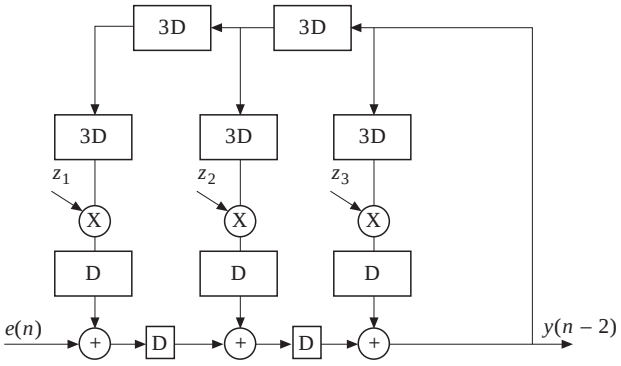

Figure 7(b). Retiming registers in the $H(z)$ feedback loop.

Data coding

The use of fixed arithmetic takes into account the coefficient and internal data coding. To limit the bit coding an empirical approach is chosen. In [10], we show that the bit coding is highly dependent on the 's' parameter. In this way, for $s > 1$, a 12 bit coding for coefficients and internal data is a good tradeoff between filter performances (signal to noise ratio and localization) and silicon area. For $s < 1$, 16 bit coding is necessary.

In Figures 10, 11 and 12 we show the results obtained for this image (it is a circle drowned in a Gaussian noise with a signal to noise ratio of 17 dB), for $s = 2$ and with a 12 bit data coding and with real calculus (on 64 bits).

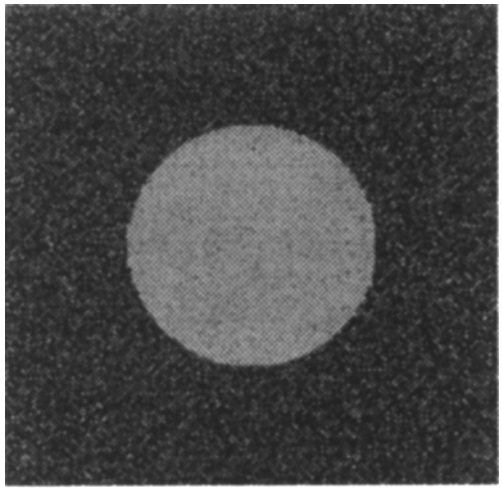

Figure 8. Test image with noise.

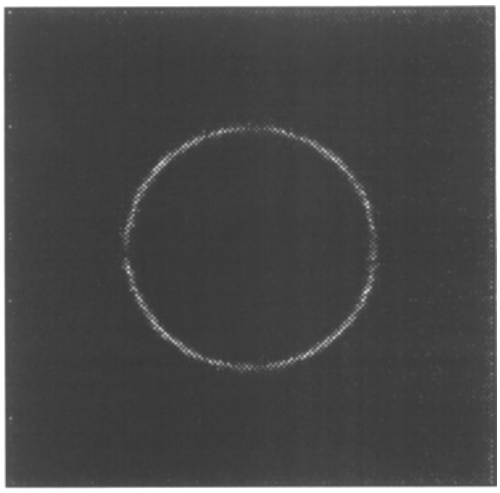

Figure 9. Edge detected with $s = 2$ and 12 bits data coding.

ASIC Implementation of the Look-ahead Filter

In this section the architecture and the realization of dedicated circuit (ASIC) for a ramp edge detection is presented. The look-ahead calculations are obtained with the method developed in the previous section. The design is made such that four identical circuits are used to obtain the horizontal and the vertical gradient. We describe from an initial bloc diagram how we have been able to reduce the internal memory and to synchronize processings of left to right and right to left for lines, top to bottom and bottom to top for columns.

X+ are the results obtained by the filtering in sweeping lines from left to right, X− in sweeping them right to left.

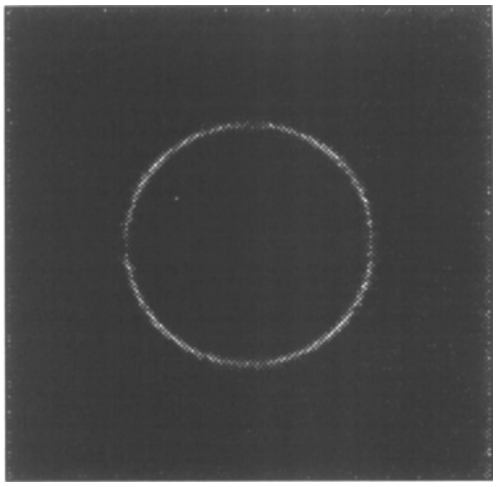

Figure 10. Edge detected with $s = 2$ and with real calculus.

Y+ are the resuIts obtained from the filtering in sweeping columns top to bottom, Y− in sweeping columns bottom to top These results are calculated in parallel with the help of four identical circuits. They process the same image in parallel and synchronous manner. The edge extraction operator is given in Figure 11.

The general idea (simplified) of the working of such a circuit is:

(i) loading coefficients.
(ii) loading an image line (column) into a 512×8 bits of RAM.
(iii) processing the line (column) following X^+ and X^- (there Y^+ and Y^-). The filter is managed by a special control unit.
(iv) results of the processing of each filter are loaded in an output RAM (512×12 bits).
(v) $X^+[i]$ and $X^-[i]$ addition (there $Y^+[i]$ and $Y^-[i]$) is done in a very rapid manner compared to the iteration frequency of the filter. This process is started as soon as the two necessary terms for the addition are available.

One RAM architecture

The objective is realized as a single circuit, with the two choice modes of running as X^+ or X^-.

In order to develop an architecture based on muxes (Figure 11) four identical circuits are necessary to obtain the vertical and horizontal gradient.

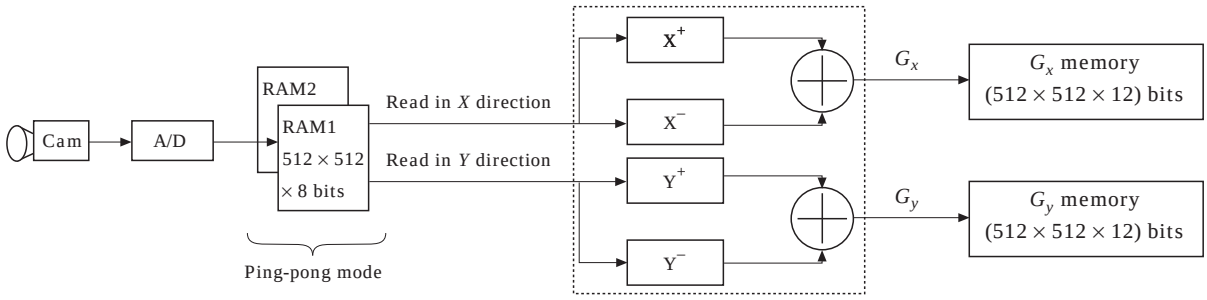

Figure 11. General board architecture.

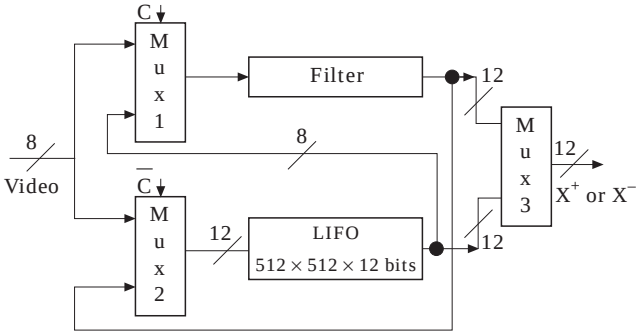

Figure 12. Circuit mode X^+ or X^-, following the value of "c".

If "c" is equal to zero, the filter realizes the X^+ mode the input video (8 bits). Data is filtered, then loaded by the LIFO.

If "c" is equal to one the input video is then directed to the LIFO, then filtered so as to obtain the processing following X^-.

ASIC bIocks

This subsection is dedicated to the functions of each of the ASIC blocks.

Figure 12 shows the main functional blocks that form the heart of the circuit as well as the different signals of input/output and command signals.

The different blocks are: the filter, the control unit, coefficients registers, the input muxes (MUX1 and MUX2) and output (MUX3), the LIFO memory, and the address counter.

All the filter coefficients are on 12 bits and are represented in two's complement notation. Coefficients are loaded from the video data bus.

As the bus of data is on a byte, to load 12 bits two clock cycles are necessary (loading the first 8 bits in parallel manner, and on the clock edge following loading the last four bits). The pipelined filter (Eqn 11) has 11 coefficients, and thus programming requires 22 clock cycles. The control unit manages the coefficients.

We have seen (Eqn 11) that the filter implementation is a 12th order recursive filter. Its equation contains 11 coefficients. Each coefficient ends at one of the two inputs of the associate multiplier. The filter has 11 multipliers. Three of them are in the recursive part and eight in the non-recursive part. The structure of these multipliers is based on the Baugh-Wooley algorithm.

This multiplier architecture was chosen because of its ability to be pipelined [11,12]. For the purpose of reducing times of partial products additions, Wallace [13] and Dadda [14] have proposed techniques to manage the different adders undertaking the partial product addition.

The Baugh-Wooley [12] multiplier allows the generation of partial products for mixed products (two's complement (C2), no signed, no signed X C2) in a very regular structure.

The main blocks of the filter are: memories element; multipliers; control blocks for the initialization; and adders.

ASIC features

The circuit has been realized with the CAD tool Cadence, using the ES2 standard cell library in a CMOS 1.2 μm technology. The edge operator features are: frequency (post-layout), 20 MHZ (the multipliers until 50 MHz); it processes 512 × 512 pixels images; area: 60 mm^2; pixels coding, 8 bits; all the internal arithmetic on 12 bits.

The results presented here confirm the complexity of realizing a recursive filter (processing a pixel to a frequency greater

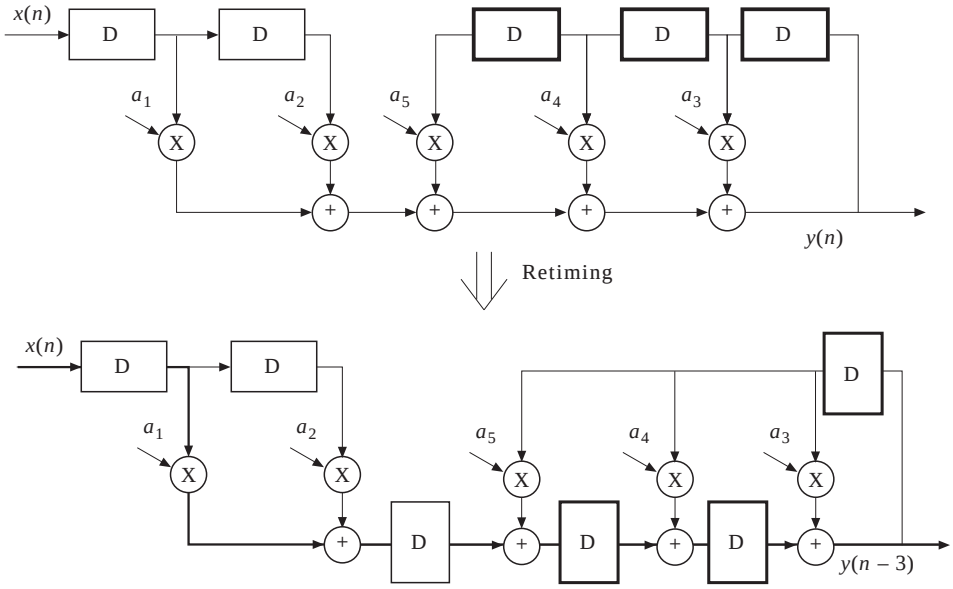

Figure 13. Local retiming method. (—) Critical path.

than 10 MHz). The gradient ($G_x = X^+ + X^-$) can, in parallel, only be obtained by the use of two circuits of this type.

Thus we propose another alternative to improve the area/speed trade-off. This method consists of implementing the filter directly without look-ahead calculation, but moving the registers in the recursive part of the filter [5].

Improvement of the Direct Filter Implementation

This section proposes a new approach based on an optimized clocked circuit by relocating registers in order to reduce combinational rippling. Unlike pipelining, direct retiming does not increase circuit latency. As shown in Figure 13, only the recursive part remains. Note that the two architectures use the same functional elements connected in the same manner, except for the locations of registers.

The frequency of this architecture is given by the time through one multiplier and one adder. With this method three adders are eliminated in the critical path, allowing a speed increase of about 20% for the same area.

The choice for the final idea of the dedicated processor has focused on this architecture.

Architecture

Due to the necessary memory space for the calculation of the edge operator. an implementation of the architecture, described in Figure 13, in a single programmable component (FPGA) is not suitable. The structural implementation of the ASIC has been done using a cell-based approach in a CMOS 1 μm technology, with the CAD tool OPUS (Cadence Design Framework II). The proposed architecture is presented in Figure 14.

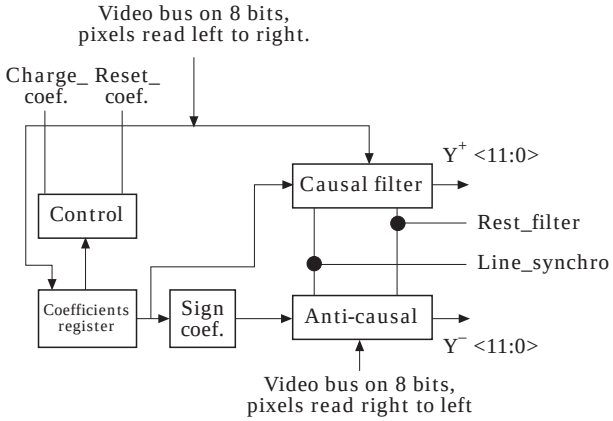

Figure 14. ASIC architecture.

The different blocks are: the causal filter, the anti-causal filter, the latches for loading coefficients, and one block to change the sign of coefficients a1 and a2 (anti-causal filter). The circuit is controlled by the following signals: charge_coef, reset_coef, reset_filter, line_synchro, scan1 and scan2 (test of filters), clock. The output of filters are given by Y^+ and Y^-. The gradient operation is (G_x), calculated on the exterior of the processor.

Filters

This circuit integrates a causal and an anti-causal filters. Each of these filters is composed of five 12 × 12 bit multipliers, 12-bit four adders and six 12-bit registers (Figure 4). Only the signs of the coefficients a1 and a2 are opposite between the causal filter and anti-causal, which justifies the necessity of inserting a supplementary logic to change the sign of these coefficients (bloc «coeff-sign» on Figure 14.

The cell-based approach seems sufficient for the realization in ASIC. However, it is necessary to determine the multiplier/adder which allows the best speed/area trade-off. The multiplier thus generated has suitable performances (less than 20 ns for multiplication 12 × 12 bits, for a silicon area lower than 1 mm^2). The choice of the adder is focused on an architecture of the type "carry-select", offering the best trade-off between speed and area (two addition words of 12 bits in 5 ns and a silicon area lower than 0.1 mm^2). In a cell-based approach the multiplication and addition operation is achievable in less than 25 ns.

Coefficients

All five coefficients are in a two's complement notation, and are serially loaded in 10 clock cycles into the latch by the video bus (8 bits). Once the coefficients are loaded, they are available at the input of the corresponding multiplier. The coefficients can be initialized by setting the reset signal to zero.

Results

Prototypes realized with this approach, in a CMOS 1 μm technology, have been tested successfully. Indeed, the obtained dedicated processor (Figure 15) is able to process a pixel in 30 ns, with variable size images (64 × 64 – 1024 × 1024 pixels). The complexity of the circuit is about 72 000 transistors for a silicon area of 29 mm^2 (5.6 × 5.3 mm^2).

Note that we have not integrated the LIFOS in order to economize the silicon area.

Note also that the sensitive improvement of the speed of functioning of the circuit, and the idea of using two identical circuits, the vertical and horizontal gradient, are calculated in parallel.

Comparison of the different methods

We have proposed different solutions for the integration of the third order recursive Canny-Deriche filter. In previous sections we have described the direct implementation of this filter, developed an adequate method to the acceleration of this type of filter, and in the previous section we have seen how a simple modification of the filter architecture leads to a sensitive increase of performances. Table 1 compares performances of these different implementations. To realize an objective comparison we have integrated a CMOS 1 μm technology for all of these solutions.

According to the aimed application, this study brings different integration solutions. In all cases, it is conceivable to process images of size larger than 512 × 512 pixel images.

Table 1. Comparison between the different methods of the optimized Canny-Deriche filter implementations (Technology CMOS 1 μm ES2)

Implementation	Estimated performances		Real performances	
	Area (mm^2)	F (MHz) (mm^2)	Area	F (MHz)
Direct	5.2	25	–	–
Global retiming	15.3	71	24.5	66.5
Local retiming	5.3	40	7.5	37.5

Estimated: Pre-layout results take into account an estimation of interconnect capacitances.

Real: Post-layout results are obtained after the place and route phase and the extraction of capacitances.

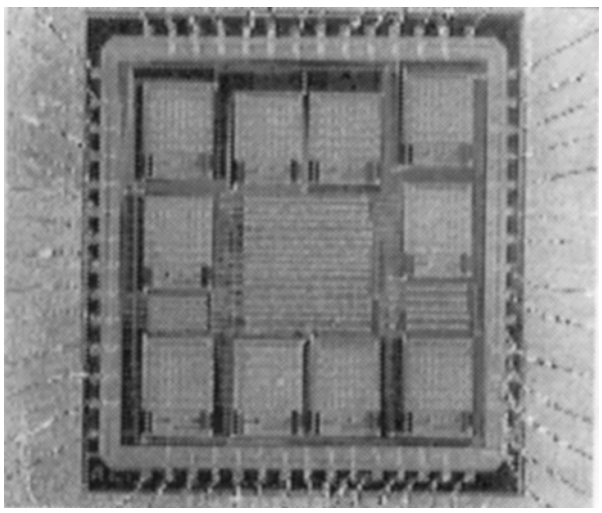

Figure 15. Chip picture. Real time edge detection operator (optimized Canny–Deriche algorithm) with a clock rate of 33 MHz for a silicon area of 29 mm^2 (72 000 transistors), internal precision 12 bits.

The solution of the pipelined filter with look-ahead technique (global retiming) allows data flow in the order of 800 Megabits/s (66.5 MHz by pixel), three times more than a direct filter implementation, but with a silicon area cost increase of three times that the original.

The local retiming method proposed represents a good area/speed trade-off. Indeed, the cost in area is minimal (area identical with the direct implementation), with a data flow in the order of 400 Megabits/s (37.5 MHz by pixel) that allows us to process images of size 1024×1024 pixels in real time.

Conclusion

The digital recursive filter acceleration is a necessary step for the implementation of algorithms in real time.

This algorithm-architecture-silicon adequation work for a third order recursive filter has allowed us to review the different recursive filter acceleration techniques.

We have thus demonstrated two techniques of filter integration. Firstly, the scattered look-ahead technique, with power-of-two decomposition technique calculation, was demonstrated on the first dedicated processor. Secondly, the more direct method allowed us to improve appreciably the area/speed compromise, which is an important factor during the design of the circuit

This study has been based on a high data flow dedicated processor. The satisfactory results obtained have allowed us to develop an image processing board around this processor.

Acknowledgements

The authors wish to thank G. Cathebras and R. Lorival (University of Montpellier II-Laboratory LIRMM) and C. Milan and J.P. Zimmer (University of Bourgogne-Laboratory LIESIB) for their help.

References

1. Roberts, L.G. (1965) Machine perception of three dimensionals solids. Optical and Electro-optical information processing.
2. Prewitt, J.M.S. (1970) Object enhancement and extraction. In: B.S. Rochester, Lokim and A. Rosenfeld, Picture processing and psychopictoris. Academic Press, pp. 75–149.
3. Canny, J. (1986) A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell*., **PAMI-8**: 679–698.
4. Deriche, R. (1987) Using criteria to derive a recursively optimal edge detector. *Int. Journal of Comp. Vision*: 167–187.
5. Zarka, N. (1992) Conception d'un circuit intégré de détection optimale de contours. Phd thesis, University of PARIS.
6. Bourennane, E. (1994) Conception et implantation d'un détecteur de contours optimisé sous forme d'un circuit ASIC. Phd thesis, University of Burgundy.
7. Bourennane, E., Paindavoine, M. & Truchetet, F. (1995) Amelioration du filtre de Canny Deriche pour la détection de contours sous forme de rampe. *Traitement du signal*, **10**: 297–310.
8. Leiserson, C., Rose, F. & Saxe, J. (1983) Optimizing synchronous circuitry by retiming. *Proc. 3rd Caltech conf VLSI*, Pasadena, CA.
9. Fettweis. (1987) Realizability of digital filter networks. Second Edition. Addison-Wesley Publishing Company.
10. Torres, L., Bourennane, E., Robert, M. & Paindavoine. Implantation du détecteur de contours Canny-Deriche optimise sous forme d'un circuit spécifique. *Traitement du signal*, (submitted).
11. Weste, N.H.E. & Keshraghiain. Principles of CMO VLSI design, second edition. Addison-Wesley Publishing Company.
12. Baugh, R. & Wooley, B.A. (1973) A two's complement parallel array multiplication algorithm. *IEEE Transactions on Computers*, **C-22**: 1973.
13. Wallace, C.S. (1964) A suggestion for a fast multiplier. *IEEE Transactions on Electronics Computer*.
14. Dadda, L. (1987) Some schemes for parallel multipliers. *Alta Freguenza*, 34: 167–187.

Appendix

Calculation of $q_i(k)$

The recursive part of an *N*th order filter with *k* pipeline latches in its fee-back loop is described by:

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

The original transfer function corresponds to $k = 1$ and $q_i(1) = a_i$. We obtain an equivalent $2k$ pipeline level by multiplying the numerator and denominator by:

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

where

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

With the relationship between $q_i(2k)$ and

$$\begin{aligned} & q_i(k) \quad : \\ & q_1(2k) = q_1^2(k) + 2q_2(k) \\ & q_N(2k) = (-1)^{N+1} q_N^2(k) \end{aligned}$$

For an Nth-order odd filter:

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

For an Nth-order even filter:

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