Two ASIC for Low and Middle Levels of Real Time Image Processing

P. Lamaty*, B. Mazar*, D.Demigny**, L. Kessal**, M. Karabernou**
** Aérospatiale-Matra-Missiles, Département électronique, 18 rue Le Brix, 18000 Bourges cedex, philippe.lamaty@aeromatra.com*

*** ETIS, UPRESA CNRS 8051, ENSEA et Université de Cergy Pontoise, ENSEA, 6 av. du Ponceau, 95014 Cergy Pontoise cedex, demigny@ensea.fr*

Abstract: This paper presents the system architecture (with main details on each algorithm) of two ASIC for real time image processing designed with Aérospatiale-Matra industry. The first chip: OREC is dedicated to low level processing (edge detection) and includes a large 2D convolution filter 12x12, gradient computation, extraction of the local maxima of the gradient and thresholding. The second chip: OPNI is dedicated to intermediate level image processing. Processes or IP for edge thinning, region labeling, edge chaining (line segment extraction, line segment chaining, polygonal approximation and little chains elimination) are included in OPNI as well as a DSP core and a mathematical coprocessor based on Cordic method for trigonometric computations. Both VLSI chips have been successfully tested. They are used in a European project : obstacle detection for vehicule. The maximum frame rate reaches 25 images per second for 1024x1024 image size, and more than 110 images per second for 233x256 image size.

Key words: asic, image processing, edge detection, region labelling, polygonal approximation, edge chaining

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35597-9_40

M. Robert et al. (eds.), *SOC Design Methodologies*

1. INTRODUCTION

1.1 Purpose

Low and intermediate real time image processing is still a challenge for embedded system. Common multi-DSP programmed architectures are ineffective considering size and consumption criteria. If the most powerful DSP satisfy the real time constraint for one algorithm, they are not enough efficient to support the whole processes required by an application. Beyond the computation architecture, real time image processing requires high memory bandwidth. Wired architectures associated with dedicated memory organization remain the best solution. The global function of the wired architecture is the reduction of the data bandwidth which are transferred to the high level processing stage commonly build with programmable architectures.

From the collaboration between the vision and electronic departments of Aérospatiale-Matra industry and the ETIS research lab, it results the design of two VLSI chips which offer a complete solution for low and intermediate image processing.

The first chip: OREC (see section 2) is dedicated to low level processing (edge detection). It includes a large 2D convolution filter (12x12), gradient computation, extraction of the local maxima of the gradient, and thresholding of the gradient maxima.

The second circuit: OPNI (see section 3) is dedicated to intermediate level image processing. Processes or IP for edge thinning, edge closing, region labeling, edge chaining (line segment extraction, line segment chaining, polygonal approximation, elimination of little chains), corner detection, are included as well as a DSP core and a mathematical coprocessor based on Cordic method for trigonometric computations.

All the functional block have been designed as IP. This allows an easy reuse of each part of the design.

1.2 Constraints

Behind the high data rate inherent to real time image processing, two main difficulties arise in this project.

- In the application planned by Aérospatiale, the image processing flow is used to control actuators which modify the camera position and then the visual scene (feedback loop). Then the latency of the computation has a strong effect on the system stability. This induces that no frame buffer can be used in front of and inside the treatments. Data flow computations must be done on the fly, when pixels are extracted (at the video rate)

from the camera. A second involvement is that the latency constraints reduce the choice of the available algorithms that can be used on each step as it will be discussed in the rest of the paper.

- The second difficulty is that all the algorithms and IP architectures must be consistent to lead to an efficient system (data format, exchange, interfaces, timing).

1.3 Design methodology

The major aspect is that architect and hardware specialists have been associated to the image processing team in the early steps of the project. So they could contribute to the algorithms choice in the way of a feasible solution and they could help the discussion about how to preserve the quality for the lowest complexity. The algorithms have been first written in C language with bit true computation. Then VHDL was used for digital design. Real images were used for the simulation tests. Results in C and VHDL implementation have been compared to ensure the functional quality of the design.

2. THE OREC CHIP

2.1 Main functions

The OREC chip is dedicated to edge detection. The figure 1 shows the different blocks and the main data flows of this chip.

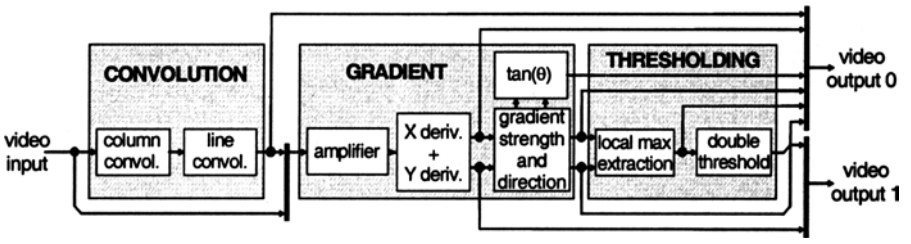

Figure 1. OREC functionality. Two output video buses allow to extract the results of different blocks (ex. edges and gradient orientation tan(θ) for a fast Hough transform)

Edge are detected has the local maxima of the gradient of the intensity pixel value. In a recent work [4], we show that gradient computation can be made with a separable 2D smoothing filter (which can be computed with two 1D filters) associated to a simple 2x2 Roberts local derivating filter (for the first derivative computation). This scheme reduces the computation cost by a

factor two compared to more classical approaches. An other advantage is that the resolution of the detection is only fixed by the choice of the smoothing coefficients. The gradient and threshold parts remain unchanged when the resolution varies.

The video input are 8 bits width and the two pixel synchronized video output are 16 bits width. All the memory required for the treatment are included in the chip. Two synchronization standards (Imaging and DataCube) are allowed. All the parameters (synchronization, filter and amplification coefficients, threshold levels, output sources) can be changed on the fly. The maximum image size is 1024x1024, and the maximum pixel rate is more than 20Mhz with military constraints. The chip was realized in a CMOS 0.6 μm technology process. It uses 100K equivalent gates and 160 Kbits of memory and includes boundary scan, full scan et bistram tests.

2.2 Smoothing filter

Recursive filters as the Deriche filter [2] are often used because they offer the best noise reduction for a given computation cost. Unfortunately vertical recursivity in image processing induces a frame delay which is not compatible with our latency constraint (a few lines delay). Then, we choose to use a large kernel FIR 2D separable smoothing filter computed with two 1D convolvers. Each of them implements a symmetrical FIR filter with 12 coefficients which requires 6 multipliers and 11 adders. To avoid transient responses near the image boundaries, a mirror effect is generated on the "shadow pixels" near each horizontal or vertical image frontiers. This is controlled with a simple state machine. In order to limit the number and the size of the line buffers (remember that no frame buffer exists between the camera and the chip), the vertical convolver is placed in front of the horizontal one. To preserve the detection quality, the smoothing result is conserved without data truncation.

2.3 Gradient computation

The amplifier is needed to ensure that the gradient strength will be independent on the resolution. So the same threshold can be used for all the resolutions. Small size Roberts derivative filters are used to compute the x and y components of the gradient. The following equation gives the expression of the horizontal gradient component G_x where $s_{i,j}$ is the smoothed output (column i, line j).

$$G_x = (s_{i+1,j} + s_{i+1,j+1}) - (s_{i,j} + s_{i,j+1}) \quad (1)$$

Inverting *i* and *j* leads to the vertical component G_y expression. Instead of using expensive square root and square computation, the gradient strength is approximated with:

$$G = \max\left[|G_x|, |G_y|, \frac{3}{4}(|G_x| + |G_y|)\right] \quad (2)$$

which has a 6 % precision and only requires a Max computation, two incrementers and two adders. This is sufficient because the local maxima detection uses relative gradient weight and because the sensibility of the threshold is not too high. The gradient direction is computed with a 22.5° precision for the local maxima extraction. Furthermore a divider is used to compute $\tan(\theta)=G_y/G_x$ or its inverse with a 8 bit precision. This allows the implementation of a fast Hough transform. The gradient orientation is used to select the location of the vote in the Hough space and considerably reduces the memory bandwidth compared to the classical implementation. Only one line buffer is needed for gradient computation.

2.4 Thresholding

The local maxima extraction operator selects the highest gradient value in the gradient direction. the gradient direction is used to select involved pixels in a 3x3 neighborhood. Balancing (function of the gradient direction) of the gradient strength of involved pixels is made to improve the detection. Comparisons are then computed with the gradient of the centered pixel. Then, a double thresholding of the gradient of the retained pixels is applied. A high level threshold selects the pixels which are surely edges. A low level threshold selects doubtful pixels which can help to close the contour lines on the OPNI chip.

3. THE OPNI CHIP

3.1 Main functions

The OPNI chip (see figure 2) extracts features from an input video with dedicated hardware operators localized on the higher part of the figure 2.

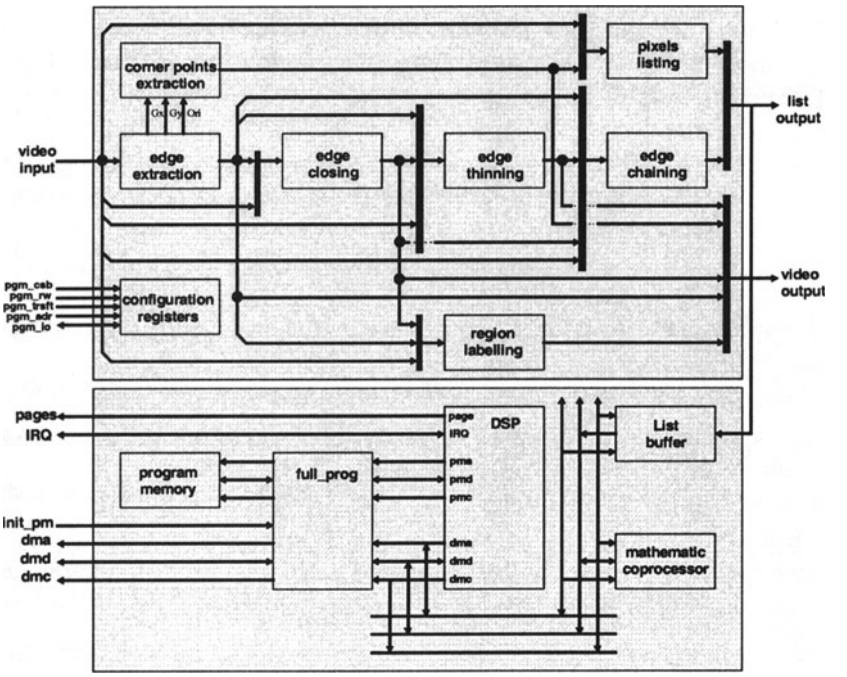

Figure 2. OPNI fonctionnality. Two buses one for video frame an the other for data list extracted the results of the different IP. The lower part of the figure shows the DSP and its mathematical coprocessor which can be used for features processing

These features are corner points, vector extremity of polygonalized contour lines, and region labels. The two first ones are reorganized to form list of pixels which allows a more compress form than the video flow where a lot of pixels are useless. The second part is based on a DSP core associated to a mathematical coprocessor. They are used to implement high level image processing algorithms which operate on feature lists. The OPNI can be used without preprocessing when images are sufficiently noiseless. In this case, the edge extractor is used on the video input. The video output is 10 bits wide. The list output is 32 bits wide which allows to extract in parallel the pixel coordinates and their characteristics. The maximum number of detected segments is 2^{18} and the maximum number of retained labelled regions: 2^{10}. As for the OREC chip, two synchronization standards (Imaging and DataCube) are allowed. All the parameters can be changed on the fly. The maximum image size is 1024x1024, and the maximum pixel rate is more than 20Mhz with military constraints. The chip was realized in a CMOS 0.35 μm technology process. It uses 240K equivalent gates (DSP core not included) and 200 K bits of memory. It includes boundary scan, full scan et bistram tests. Its size is 71 mm^2. Edge extraction is based on the same architecture as in the OREC chip, so it is not more discussed here.

3.2 Corner points extraction

The gradient operator (stength and orientation) is applied again to the gradient of the intensity image. Then, the local maxima of the second derivatives are extracted. A threshold operation retains the edge pixels which have a high curvature value. Finally a list of the selected pixel is built. The same hardware architecture as for edge detection is used in this block.

3.3 Edge closing

Edge closing is based on a cellular automaton (as the game of life)[1]. 64 elementary processors implement a four state cellular automaton. Initially, each pixel can be background, strong edge (gradient is more than high threshold), weak edge (gradient is between high and low thresholds) or extremity of a strong edge. So pixels are stored with 2 bits.

The automaton rules are simple and operate on a 3x3 neighborhood:

- each weak pixels which has at least two not background neighbors and which has an extremity neighbord is change to an extremity pixel,
- each weak or extremity pixels which has less than two not background neighbors is change to a background pixel.

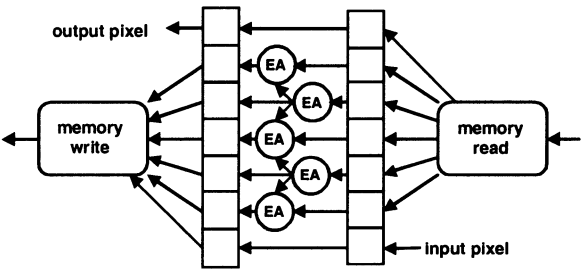

Figure 3. Organization of the closing edge operator. AE are automaton elements

The iteration of the automaton rules propagates along the weak edges a closing wave starting from the extremities. The waves which reach a strong edge are finally conserved. At the end, extremity pixels are changed to strong edge pixels, weak edge pixels are changed to background. To applied the automaton rules 32 times on each pixel, 33 lines (2 bits wide) are stored in FIFO. The figure 3 shows and example with 5 iterations and 6 stored lines. This operator scans the image in a classical video flow. At each cycle, a column of 32 pixels is read from the line memory, the current pixel of the same column is input, 32 pixels can change their state and will be stored in the line memory, one pixel is definitely output. This dataflow edge closing

operator can be implement very simply and easily pipelined. Full implementation details can be found on the reference [1].

3.4 Edge thinning

Edge chaining requires that edges will be one pixel wide. Because various edge detectors (as morphological operators) can be used in front of the OPNI chip, we have implemented an edge thinning operator. A simple filter which operates on a 3x3 neighborhood. Edge pixels are retained if and only if they are useful for edge connectivity. Results for each possible configuration of the neighborhood is stored in a look up table (see figure 3). The main advantage of this scheme is that it is a simple dataflow algorithm. The major drawback is that resulting edges are pushed in the scan direction. However, The misslocation is reduced when the edge thickness is small.

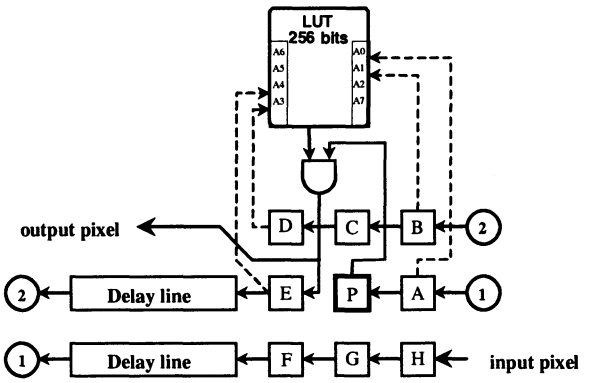

Figure 4. Details of the edge thinning operator. P is the processed pixel, its value is changed as a function of its neighborhood. The change is immediatly taken into account in the context of the following pixel

3.5 Edge chaining and polygonalization

This part is composed of four successive blocks: segment extraction, segment chaining, polygonal approximation, little chains elimination.

Segment extraction is based on a simple automaton which works on a 3x3 pixel neighborhood during video scanning. The informations about each segment detected are stored in a queue memory. The segment chaining links the connected segments and build a list of segments stored in memory with dynamic allocation. Each segment is then described by the coordinates of its extremities. This part is based on a state machine with 24 states which trigges memory pointers modification.

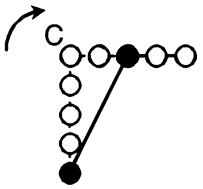

Figure 5. An Example of a bad approximation when corner points are not taken into account

The polygonal approximation [3] is made by an iterative approximation between corner pixels. This is needded in order to conserve the global shape of the features. The figure 5 shows the bad approximation result when the corner point C is not detected. So, corner pixels are detected has the vertex of two segments which strongly differ in direction. To do that, the distance (vertically and horizontally) between two successive vertex is computed. Successive distance results are combined and compared to a threshold in order to retain high curvature vertex.

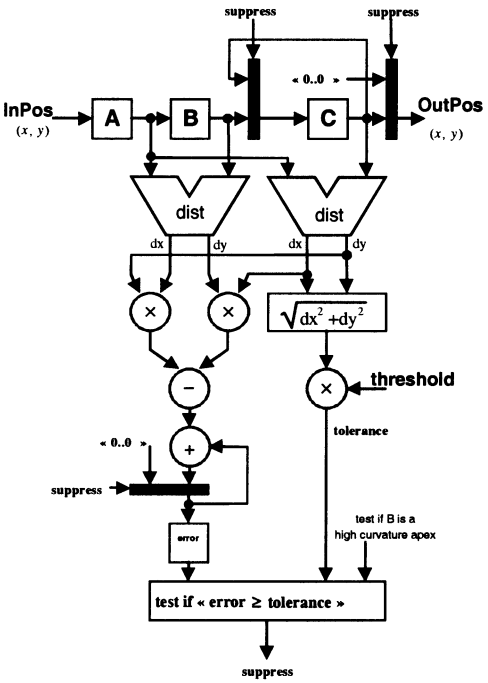

Figure 6. Iterative approximation for the edge polygonalization

The iterative approximation starts from a corner point or an extremity and includes (step by step) successive vertex. At each step, the approximation error is refreshed.

When the error becomes greater than a given tolerance, the approximation process of the present vector stops. Details of the iterative approximation are given in figure 6. Here, we try to suppress the point stored in register B. C is the memorized starting point of the approximation vector. The output of the substractor is the area generated when B is suppressed. The following adder accumulate successive area errors. To take the decision, the error is relatively ponderated by a fraction (threshold) of the vector length computed with $(dx^2 + dy^2)^{1/2}$. B is finally suppressed if it is not a high curvature vertex and if the global area error is less than the tolerance. Little chain elimination is simply realized by a new distance computation between vertex. An other particularity of our implementation is the use of Freeman code in order to reduce the memory size.

3.6 Region labelling

This algorithm is based on the original algorithm from Rosenfeld [5]. It involves not only "pixel computation" but also global computation to solve equivalence between identical regions which first appeared as different because of the video scanning order. Most of the computation cost occurs when a edge is encountered but no work has to be done for "background" pixels. So, it is not a good idea to synchronize the algorithm on the pixel rate, because in this case, the computation cost cannot be distirbuted with regularity. To avoid this problem, we have introduced RLE compression and uncompression before and after the region labelling. Algorithms and architecture have been modified consequently.

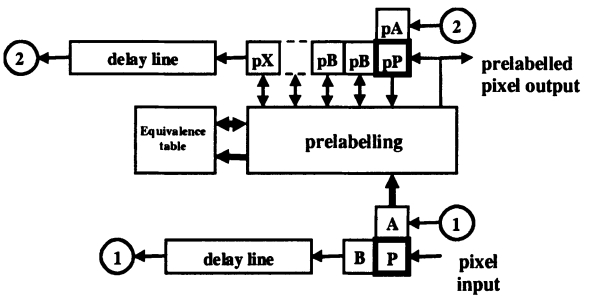

Figure 7. Architecture for the region labelling

The architecture is based on two parts. The first part scans the edge image and build a fictitious prelabelled image. We store only the equivalence between region which have been initially supposed different

during the scan and the edges image. At the end of the image scan, a scan from low to high labels of the equivalence table computes a new table which contains final labels. Then the second part of the architecture is used to scan again the edge images but now, when a new region arises, the final label is find in the equivalence table. Compare to more classical scheme, this architecture does not need to store prelabelled image but only the edge image. The figure 7 gives the organization of the prelabelling scheme (first part). Here, the pixel P is computed. Two neighborhoods are used:

- at the bottom of the figure from the edge image, the pixel B at the left side of P and the pixel A at the up side of P,
- at the top of the figure, prelabelled values pA, pB, pP of pixels A, P, B and of some of the predecessors of B.

Configuration (edge or background) of the pixels of the neighborhood decided that pP is or is not a new region. Usually, only the pixels connected to P (A,B) are used. Here, we temporary store the label results of the last pixels (in the left of the top neighborhood). After a fixed delay the final decision is made for all the stored pixels. This scheme reduces by a factor of four the equivalence memory size.

The same architecture is used for the second part (final labelling) except that the prelabelled neighborhood is reduced to (A,B,P) and that the equivalence table is just read (not written).

4. CONCLUSION

Both circuits have been successfully tested and are used in different systems. The maximum frame rate reaches 25 images per second for 512x512 image size in a prototype designed by Aérospatiale-Matra. A PC card was also build to designed a prototype used in an european project (obstacle detection in front of cars). More than 110 images per second for 233x256 image size are computed by this PC card. More recently, all the IP of the OREC and OPNI have been included in a Xilinx Virtex II 3M gates FPGA. Sixty images of size 1280x1024 per second are computed by the FPGA. This implementation outperforms all the designs based on DSP architectures for speed, compactness and consumption criteria and prove the reusability of the IP designed in this project.

5. REFERENCES

[1] J. Devars, D. Demigny, and J.F. Quesne. Boundary closing with asynchronous cellular automata. IEEE Conf. on Computer Architecture for Machine Perception, pages 81 – 88, 1991.

[2] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. Int. J. Computer Vision, pages 167 – 187, 1987.

[3] P. Garnesson, and G. Giraudon. Chaînage efficace de contours. Rapport de recherche INRIA, n° 1621, 1992.

[4] F. Garcia Lorca, L. Kessal, and D. Demigny. Efficient ASIC and FPGA implementations of IIR filters for real time edge detection. Proc. International Conference on Image Processing, n° 2, pages 406 – 409, Santa Barbara, october 1997.

[5] A. Rosenfeld, and J.L. Pflatz. Sequential operations in digital picture processing. Journal of ACM, volume 13, pages 471 – 494, 1966.