FPGA implementation of a real time multi-resolution edge detection video filter

Jeremie Hamon[1], Vincent Fristot[1] and Robin Rolland[2]

[1] PHELMA - Grenoble Institute of Technology - jeremie.hamon@grenoble-inp.fr
[1] PHELMA - Grenoble Institute of Technology - vincent.fristot@grenoble-inp.fr
[2] CIME-Nanotech, Grenoble - robin.rolland@grenoble-inp.fr

***Abstract*—This paper presents digital video filters labs for final year engineering students. The project deals with the implementation of Canny Deriche optimal edge detectors on a FPGA plateform. The target of these labs is to illustrate the design of integrated electronic systems and to introduce the concept of architecture/algorithm adequacy.**

***Index Terms*—Engineering student labs, embedded filters, edge detectors, real-time video processing, FPGA, SoC**

I. Introduction

This paper presents an advanced digital design project using FPGA for final year engineering students. The aim of this project is to study and to implement a real time edge detection video filter on a DE2 Altera Cyclone II Development Board [1]. This kind of project presents several pedagogic interests. Firstly, it is an opportunity for students to put into practice signal and image processing theory on a concrete example. Secondly, as shown in this paper, the relative complexity of this design project leads students to explore different architectures to choose the one which presents the better performance/complexity tradeoff. In the same line, students have to propose an efficient validation methodology suited to the project complexity. Finally, this design project covers the entire FPGA design flow, i.e. from the HDL specification to the hardware implementation.

This paper is outlined as follows : the second section presents the required background theory on the edge detection filters. Then, the hardware implementation upon which the project is based is described in the third section. The fourth section presents the design environment, and especially, the DE2 Altera Cyclone II Development board [1]. The fifth section presents an illustrative example based on the lab results, and, finally, the pedagogic interests of such design project are discussed in the sixth section.

II. Theoretical background

The principle of an edge detection filter is to locate the sharp changes in the image brightness. Two methods are commonly used: the *Laplacian*-based ones and the *Gradient*-based ones [2]. The second ones, that interest us in this paper, consist in computing the gradient magnitude and direction in order to detect local maxima of the magnitude in the direction of the gradient. These maxima correspond to the image edges. Usually, a smoothing stage is applied before the actual edge detection stage to enhance the detection quality (noise reduction). The figure 1 shows the generic architecture of such kind of edge detectors.

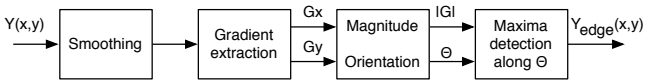


Fig. 1. Generic architecture of a Gradient-based edge detector

A. Deriche edge detection filters

Canny [3] defines three criteria of optimal edge filters: *good detection*, i.e. a low probability of failing to detect a real edge point and a low probability of falsely marking non-edge points, *good localization*, i.e. detected edges close as possible to the real edges, and a *single response* to one edge. By applying these criteria on a noisy step edge model, he demonstrates that an optimal edge detection filter can be implemented by a Gaussian filter to smooth the image followed by its derivative to extract the gradient components Gx and Gy. The size of the Gaussian filter is an important parameter as it defines the resolution/sensitivity tradeoff: smaller filters allow the detection of thin lines, while larger filters are more robust to the noise effect.

Deriche improves Canny's filter in case of digital images. He proposes to substitute the Gaussian filter with recursive filtering structure (Infinite Impulse Response filters) to reduce the computational effort required for smoothing [4], [5]. By doing so, Deriche's filter presents a fixed complexity that does not depend on the smoother coefficient. That allows multi-resolution edge detection processing.

B. Garcia-Lorca implementation

Finally, Garcia-Lorca proposes an efficient implementation of the Deriche's filter suited to FPGA architectures [6], [7]. He demonstrates that the Deriche's filter can be decomposed by four second order IIR filters to smooth the image $Y(x, y)$ (two for the horizontal smoothing and two for the vertical one), followed with two FIR Roberts filters to extract the

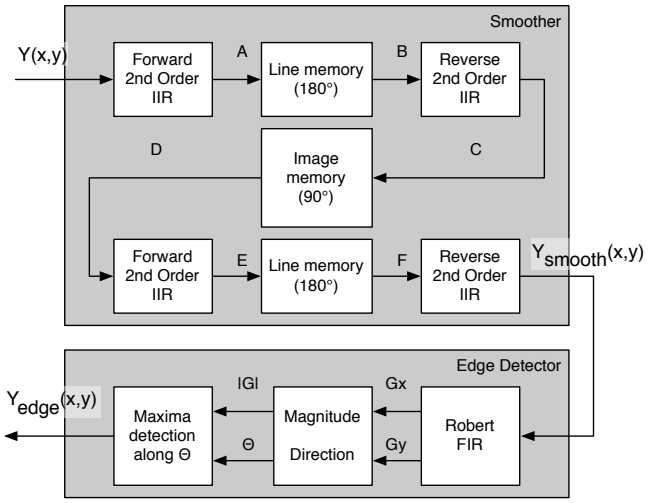


Fig. 2. Architecture of the Garcia-Lorca edge detector

gradient components (Gx and Gy). While the design project proposed in this paper relies on this implementation, we propose to detail it in the next section.

III. Edge detector architecture

The figure 2 shows the Garcia-Lorca edge detector architecture. The smoother stage is composed of four identical IIR filters, of two line memories, and of one image memory. The equation (1) represents the z transfer function of the IIR filters with γ corresponds to the smoother coefficient.

$$S_{GL}(z) = \frac{(1-\gamma)^2}{(1-\gamma z^{-1})^2} \tag{1}$$

After the first IIR filter, a line memory is used to perform the horizontal rotation of the image. The principle is to write the incoming pixels in the memory in one direction (ex. increasing addresses), and to read them in the opposite direction (ex. decreasing addresses). By doing so, the image is rotated of 180°. However, we note that this principle implies that two memory accesses are performed at each pixel cycle (one to read the pixel stored in the memory, and one to write the new incoming pixel). After this horizontal rotation, the same IIR filter can be applied to complete the horizontal smoothing. Then, the image is turned of 90° to carry out the vertical smoothing along exactly the same principle. Again, this new rotation is performed thanks to a special management of the image memory: while a pixels column previously stored is extracted from the memory, the incoming line is stored at the same memory addresses. Once again, this principle implies two memory accesses at each pixel cycle.

Possibly, additional line and image memories can be inserted between the smoother stage and the edge detector stage in order to rectify the image orientation. These optional memories are not represented on the figure 2 but have been implemented in the examples of the section V.

The edge detector stage is divided into three successive blocks. Firstly, two Robert derivative filters are applied on the filtered image $Y_{smooth}(x, y)$ to extract the horizontal and the vertical components of the gradient (eq. (2,3)). Then, the gradient magnitude and gradient direction are computed to enable the magnitude maxima detection in the direction of the gradient. These maxima correspond to the image edges. Possibly, a threshold can be added on the comparator block to mitigate the noise effect and to enhance the detection quality.

$$G_h = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \tag{2}$$

$$G_v = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \tag{3}$$

IV. Design environnement

A. DE2 development board

We have chosen the Altera DE2 Cyclone II Development board for the design project because it includes, among others devices, a large FPGA circuit, a SRAM suited for 512x512 image memorisation, and video In/Out devices (NTSC decoder and VGA controller) [1]. It includes also a large number of switches which can be used for debugging, and for setting the value of the smoother coefficient and the value of the detection threshold.

B. Video processing environment

Moreover, the DE2 package includes a NTSC video-to-VGA display project (DE2_TV) in which any video filter can be mapped. The figure 3 shows the DE2 video processing architecture. The video decoder block extracts YCrCb 4:2:2 (YUV) video signals from the external NTSC TV decoder. Because the NTSC video signal is interlaced, it is required to perform de-interlacing on the data source. That is done thanks to the SDRAM dual frame buffer and the field selection multiplexer controlled by the VGA controller. Finally, the edge detection filter is inserted between YUV 4:2:2 to YUV 4:4:4 converter block and YUV to RGB converter block as depicted on the figure 3. The edge detection is processed only on the Y video channel.

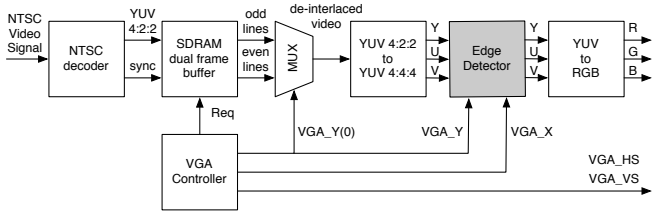
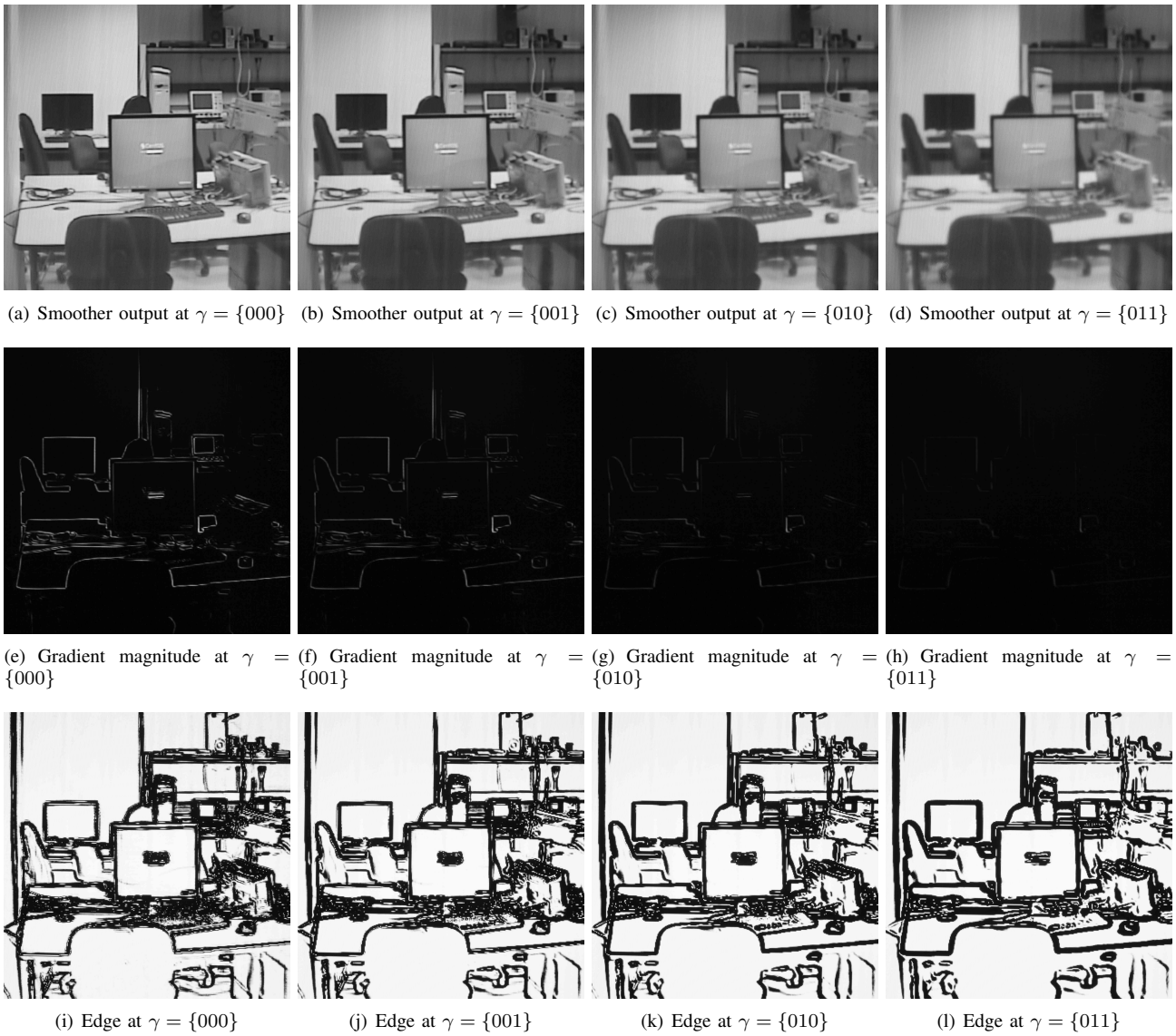


Fig. 3. DE2 video processing architecture



(a) Smoother output at $\gamma = \{000\}$ (b) Smoother output at $\gamma = \{001\}$ (c) Smoother output at $\gamma = \{010\}$ (d) Smoother output at $\gamma = \{011\}$

(e) Gradient magnitude at $\gamma = \{000\}$ (f) Gradient magnitude at $\gamma = \{001\}$ (g) Gradient magnitude at $\gamma = \{010\}$ (h) Gradient magnitude at $\gamma = \{011\}$

(i) Edge at $\gamma = \{000\}$ (j) Edge at $\gamma = \{001\}$ (k) Edge at $\gamma = \{010\}$ (l) Edge at $\gamma = \{011\}$

Fig. 4. Edge detection examples with respect to the smoothing parameter γ

We note that although the Altera DE2 Cyclone II Development board integrates a PAL TV decoder, we choose the NTSC video standard because it simplifies the clock management in the system. Indeed, NTSC systems use a refresh rate of 60 fields per second (one image composed of 2 fields), which corresponds exactly to the refresh rate of a standard VGA display (60Hz). For comparison, PAL system uses a refresh rate of 50 fields per second. The use of PAL standard would infer a resampling factor of $6/5$ which is, obviously, more complex to implement.

The video flux processed by the edge detector is clocked at 27 MHz. Because of the special memory management that requires two accesses (read and write) during one pixel cycle, the user filter and the external SRAM memory are clocked at 54 MHz.

C. EDA tools

The whole project is designed in RTL VHDL code and the functional simulations and validations are performed with *Mentor Graphics ModelSim* software. Then, the *Altera Quartus II* software is used to do the synthesis, and the place-and-route steps. Possibly, post place-and-route simulations are performed in *Modelsim* before real test on the DE2 development board.

A comprehensive example is provided to students to help them in the design. It consists in a very simple motion detector which is implemented at the same location than the edge detector (figure 3). Its principle is to compare the value of the incoming pixels with the value of the pixels of the previous image stored in the image memory so as to detect changes in the scene. Moreover, a set of VHDL test benches are

provided to simulate the operation of the DE2 video processing architecture.

V. Illustrative example

The figure 4 shows illustrative examples obtained during lab sessions on the DE2 development board. The figures 4(a), 4(a), 4(c) and 4(d) show the output of the smoothing filter for different values of the smoothing parameter γ, the images of the computed gradient magnitude are displayed on figures 4(e), 4(f), 4(g) and 4(h) as the final edge images are shown on figure 4(i), 4(j), 4(k) and 4(l). These examples illustrate the effect of the smoother coefficient as described in section II.

The table I provides a summary of the main hardware resources used by the edge detector filter and by the whole DE2 video processing architecture. We note that the memory lines are synthesised on the FPGA while the memory image used the external SRAM memory. Then, 256 Kbytes are used in the SRAM to implement the image memory of the smoother stage plus 256 Kbytes to implement the optional image memory required to rectify the image orientation.

TABLE I
Ressources Utilisation Summary - Cyclone II EP2C35F672

	LUT	Flip-Flop	RAM Bits	Multipliers
de2tv	3537 (5%)	1700 (5%)	58368 (12%)	20 (29%)
edge detector	2057	739	33792	2

VI. Educative issues

As mentioned in the introduction, this design project targets third year engineering students. For two years, it has been proposed to the students of the Phelma engineering School with the CIME Nanotech support. This experience lets think us that the project can be achieved into 7 supervised labs of 4 hours, plus several hours of personal work.

We think this design project presents several pedagogic interests. Indeed, the complexity of the signal processing algorithms implemented in this design project leads students to explore different architectures to choose the one which presents the better performance/complexity tradeoff. As an example, the IIR filter used in the smoother stage can be implemented along different architectures which lead to different performances or complexities (data precision, hardware resources, latency...). In the same way, the clock frequency constraint (54MHz in the edge detector) imposes to implement *pipeline* technic. The students have to explore different combinational partitioning solutions to chose the one which matches the clock requirements and presents the lowest latency. We think these different issues perfectly introduce the concept of the adequacy between an algorithm and its hardware implementation.

Moreover, the Garcia-Lorca implementation presents some tricky memory optimisations that seems very interesting for digital designer. From our experience, the implementation of these special memory management appears as the biggest difficulty for the students.

As well, the complexity of the project, as the fact it is carried out by groups of two or three students, leads them to propose an efficient design and validation methodology. Indeed, each student takes in charge the design and the validation of a sub-part of the project before gather together the whole project. Then, they have to clearly specify the behaviour of the different parts as well as the interfaces.

Finally, and more generally, the project covers the entire design flow of integrated electronic systems, from the architecture specification to the hardware implementation.

VII. Conclusions

This paper presents a digital design project that consists in implementing a real-time edge detection video filter on a FPGA development board. The section II introduces the edge detector theory, while the section III describes the Garcia-Lorca hardware implementation on which the project relies on. The fourth section presents the design environment and especially the video processing architecture where the edge detector is implemented. The section V provides some labs results, and the pedagogic interests of the project are discussed in the last section. To conclude, we note that the framework proposed here can be reused to implement any other kind of video filter.

References

[1] Altera. Altera DE2 Development and Education Board. [Online]. Available: http://www.altera.com/education/univ/materials/boards/unv-de2-board.html

[2] D. Ziou and S. Tabbone, "Edge detection techniques - an overview," *International Journal of Pattern Recognition and Image Analysis*, vol. 8, pp. 537–559, 1998.

[3] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[4] R. Deriche, "Using canny's criteria to derive a recursively implemented optimal edge detector," *International Journal of Computer Vision*, vol. 1, no. 2, pp. 167–187, 06 1987.

[5] ——, "Fast algorithms for low-level vision," in *9th International Conference on Pattern Recognition*, vol. 1, Nov 1988, pp. 434–438.

[6] F. Lorca, "Filtres récursifs temps réel pour la détection de contours : optimisations algorithmiques et architecturales," Ph.D. dissertation, Orsay, 1996.

[7] F. Lorca, L. Kessal, and D. Demigny, "Efficient ASIC and FPGA implementations of IIR filters for real time edge detection," in *International Conference on Image Processing*, vol. 2, Oct 1997, pp. 406–409.