Real-time FPGA Connected Component Labeling System

Elisa Calvo-Gallego[1,3], Alejandro Cabrera Aldaya[2], Piedad Brox[3], Santiago Sánchez-Solano[3]
[1]Department of Electronics and Electromagnetism, University of Seville, Seville, Spain
[2] Instituto Superior Politécnico José Antonio Echevarría (CUJAE), La Habana, Cuba
[3]Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain
{calvo, brox, santiago}@imse-cnm.csic.es, acabrera@udio.cujae.edu.cu

***Abstract*— The implementation of a connected component labeling algorithm (CCL) for real-time operation is presented in this paper. The algorithm, which was designed and implemented following a model-based methodology centered on Matlab/Simulink and Xilinx-System Generator, uses horizontal and vertical blanking periods to improve the quality of labeling and increase the operation speed. Its performance, with a VGA 640 x 480 P @ 60 Hz video, is shown by means of its integration on a complete video processing system over a Spartan-3A DSP 3400 development board.**

I. INTRODUCTION

Nowadays, the integration of vision systems in embedded devices, like PDAs, sensor networks or mobile phones, is more and more usual and necessary. However, the requirements on power consumption and size and the need of working in real time determine its performance and force to carry out a revision of the existent algorithms with the aim of adapting them to the new platforms. This adaptation process is really important for CCL algorithms since they are fundamental elements to perform intermediate processing tasks and play an essential role in the development of high level image processing applications.

As a result of that relevance, from the 60's, a great effort has been made to develop and improve this kind of algorithms, whose objective is the assignation of a unique label to each set of pixels with the same properties in the image. As a consequence, different research lines, which can be classified according to countless criterions (e.g. regularity in memory accesses, image representation way, etc), have been followed. In addition, the evolution of integrated circuits fabrication technologies has made possible the utilization of a large number of parallelism techniques that had been previously discarded due to its very high requirements (They established restrictions, mainly in image resolutions, and required external memories) [1-2].

In this paper, the implementation of a CCL algorithm inspired by Bailey's algorithm [3] and its integration on a real time system are described. Bailey's algorithm is considered a reference point in this area. It is a two-scan algorithm that requires low resources and uses the horizontal blanking periods, typical in different video standards (Fig.1), to improve the quality of labeling and clock frequency. The proposed implementation, which has been designed, verified and implemented following a model-based methodology, unlike the previous ones [4], is able to performing the full labeling process in real-time.

Moreover, the performance of the algorithm was corroborated through its integration on a demonstrator. The complete video processing system is built on a Spartan-3A DSP development board that receives the input video sequence from a camera Micron MT9V022.

The paper is organized as follows. Section II summarizes a set of basic concepts for image processing and a brief revision of CCL algorithms in recent literature. Section III describes Bailey's algorithm and the modifications that have been proposed to achieve an optimal implementation, which provides the image label in real-time. The whole system where the CCL algorithm is integrated is detailed in Section IV. Finally, conclusions are given in Section V.

II. SUMMARY OF CCL ALGORITHMS

CCL algorithms, which have been considered in this work, are applied over binary frames composed by a background (F_B, black pixels, '0') and a foreground (F_O,white pixels, '1').

They are based on connectivity analysis. Two pixels belong to one component when both are in the background or in the foreground and between them there is a path of pixels of the same type. That is, when (1) is verified, being S a subset of pixels of the image and N (S_i) the neighborhood considered for pixel s_i. Fig.2 shows an example of the number of found

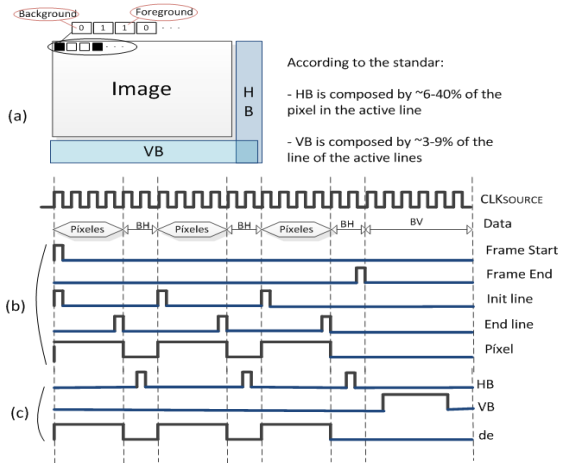

Fig. 1 (a) Image blanking periods (b), (c) Examples of synchronization signals in a video interface

elements according to the connectivity.

$$p \text{ is connected to } q \Leftrightarrow \exists \{s_i \in S \mid s_1 = p, s_{n+1} = q, \\ s_{i+1} \in N(s_i),\ i = 1, \dots, n\} \quad (1)$$

Although CCL algorithms could be classified according to multiple criterions, as our system will receive the frame in a raster scan, a classification based on the number of scans made over the image has been established. According to it, the following types of algorithms can be found:

• One-scan algorithms [5-6]: Inside this group we can find mainly contour tracing algorithms, region growing algorithms and feature extraction algorithms. The most important drawbacks of the two first kinds of algorithms are: irregular and random memory accesses for getting temporal assigned labels or pixels initial values and the difficulty to predict execution times. On the other hand, the third group of algorithms provides a set of characteristics but the image is not labeled.

• Multi-scan algorithms [7-8]: Memory accesses are regular but execution times depend on the position of pixels inside the image. Its implementation, both hardware and software, is simpler than the implementation of algorithms that belong to other categories.

• Two-scan algorithms [9-11]: In general, proposals of two-scans algorithms differ from each other in the methods and data structure used to save label equivalences and in the way how the final resolution is performed. Regarding this group of methods, many efforts have been made to search for an optimal solution in neighborhood exploration in order to minimize the number of memory accesses [12].

III. Algorithm and Implementation Details

A. Bailey's Algorithm

The algorithm performs two passes over the image.

1) First Scan:

A new pixel from the image b(x,y) arrives each clock cycle. Its label is calculated according to (2):

$$g(x,y) = \\ = \begin{cases} F_B \ \ if & b(x,y) = F_B \\ m \ \ if & [\, b(x,y) = F_o\,] \ \wedge \\ & \left[b(i,j) = F_B \,\forall \{(i,j) \in N\big(b(x,y)\big),\ \ (i,j) \neq (x,y)\}\right] \\ L \ \ if & any\ other\ case \end{cases}$$

$$L = min\begin{bmatrix} g(i,j-1), T\big(g(i-1,j-1)\big) \\ , T\big(g(i-1,j)\big), T\big(g(i-1,j+1)\big) \end{bmatrix} \quad (2)$$

- If the pixel belongs to the background, a zero label is assigned to it.
- If the pixel belongs to the foreground and all the pixels in its neighborhood (N(x,y)) belong to the background, a new label is assigned (m), whose value is increased each time that this case occurs.

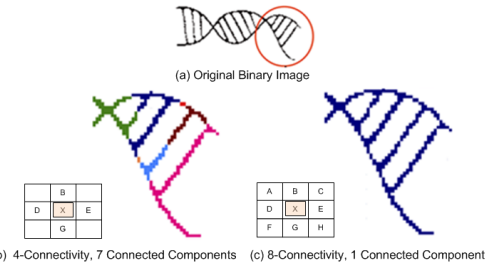

Fig. 2 Example of labeling according to the connectivity

- If the pixel belongs to the foreground and one or several pixels in its neighborhood belong to the foreground, the minimum value among the equivalent of labels of pixels that are located on the previous row inside the mask and the label of pixel that is located on the left to the considered pixel is assigned to it. When two of these values in the neighborhood are different, it is said that an equivalence has been found. In that case, if the first found element has higher label than the second one, the second one is saved in the entry of the equivalence table of the first element. On the other hand, if the second one has a higher label that the first one, the equivalent pair is saved in a two rows FIFO structure to be solved during the horizontal blanking period.

  At the end of each line, the pair of equivalent labels saved in the FIFO are extracted and processed. To each entry, (3) is done.

  $$T[F(1, FIFOFirst)] = T[F(2, FIFOFirst)] \quad (3)$$

  Once the first scan has been finished, the equivalence table is scanned. To each entry in the table, the next new value is saved

  $$if\ T(Label_i) \neq Label_i \rightarrow T(Label_i) = T\big(T(Label_i)\big) \quad (4)$$

2) Second Scan:

Each temporal label assigned after the first scan (g) is replaced with the permanent label, that is, with its final equivalent which is stored in the equivalence table (T).

The aim of the implementation proposed by Bailey in [4] was the extraction of some features (like the area or the gravitational center of the connected components) in VGA @ 60 Hz. For this reason, the labeled image was not provided at the end of the execution. The implementation was performed on a Celoxica RC100 Board (Spartan-II XC2S200 FPGA) using a Handel-C description.

B. Proposed implementation

The aim of our implementation is to process video in real-time providing labeled frames. To achieve this purpose the vertical blanking periods have been used for the resolution of equivalences between rows. Furthermore, temporal parallelism using pipeline techniques has been exploited between the two phases of the method. Fig. 3 shows a block diagram of the algorithm implementation. The main blocks are:

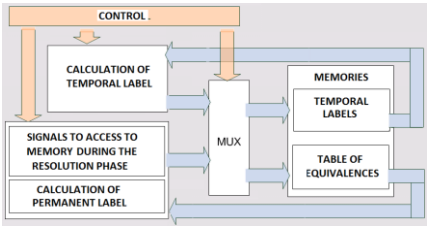

Fig. 3 Block diagram of the proposed CCL algorithm implementation

- Control Blocks, which allow to generate control signals for the algorithm from usual signals of a video interface. Two control blocks have been generated for (b) and (c) sets of signals shown in Fig. 1.
- Memories. Three different memory structures are necessary: one to save temporal labels, another one to save the equivalences between labels, and the last one (FIFO) to save equivalent pairs that have to be solved in the horizontal blanking period. All of them have an enough word size to codify all labels that can exit in the image[1] and are implemented using BlockRam[2]. The depth of the two first memories is the dimension of the image (Num_rows * Num_colum) whereas the depth of the FIFO is the number of columns of the image.
- Multiplexors to access, in the different phases of the method, to the shared resources (memories).
- Blocks for accessing to memory (to generate address/data/wre signals) and for calculating temporal labels and equivalent labels.

The main drawbacks of the proposed implementation are the amount of necessary memory, which establishes a limit in the resolution of the image to work with, and the labeling errors, that could be produced as a result of a possible lack of time to solve equivalences during the vertical blanking period. However, after software simulations [13], it has been proved that, in the majority of cases, vertical blanking period is enough to solve all the equivalences and that WCIF (512x288) resolution could be achieved without problems.

This image resolution is good for a wide range of applications. However, if the requirements force to work with higher resolutions, it would be possible to locate part of the storage space out of the FPGA by using the DDR2 Memory available on the development board. With this consideration, it was proved [13] that, in the majority of cases, WUXGA (1920x1200) could be processed. In the next section, this consideration is taken into account.

The process that has been followed to design, verify and implement the algorithm, as well as the used tools, are shown in Fig.4. Starting from software implementation of algorithms in Matlab, Simulink models, with blocks from the Xilinx blockset library, were created. After checking the functionality of these models, they were compiled in order to generate an ISE project. From ISE environment, area and time estimations as well as others test operations were done. Finally, to verify the real performance of the system, a co-simulation hw/sw was carried out, closing the whole design loop.

IV. FPGA DEMOSTRATOR

A general block diagram of the system implemented on a Spartan-3A DSP 3400 development board is shown in Fig. 5.

By means of a Micron MT9V022 camera, images (720 x 480 pixels) are captured and serialized. In that way, each clock cycle, 9 bits, which correspond to one pixel and contain information related to its position inside the image (frame valid and line valid) and its chromatic value, enter in the system with a 26,66 MHz clock frequency.

First, this image information goes through two cores developed by Xilinx in System Generator. The first one tries to remove stuck pixels with an adaptive median filter and changes the image format from Bayer to RGB. The second one is a correction gamma filter that reduces the distortion introduced as a result of the logarithmic relation between input and output light in sensor.

Once this preprocessing has been done, the images are continuously saved in the first video frame buffer (VFB1) located on the 256 MB DDR2 external memory available in the FPGA board. In this case, VFB1 has been configured to keep three consecutive frames from the camera. The writing in it is done through a video_to_vfbc core and one of the four 'Personality Interface Module (PIM)' of the 'Multi-port Memory Controller (MPMC)' that has been configured as a 'Video Frame Buffer Controller (VFBC)'[3]. This kind of interface implements a 'video direct memory access (VDMA)' core that allows a fast and correct transference of useful image information.

When the frames start to be saved in VFB1, a reader start to get frames from that VFB too, using the second PIM configured in the MPMC, although its resolution is smaller than the written one (VGA, 640x480). To synchronize the writer and the reader, a mechanism of lock based on register address is established. In this way, the writer tells to the reader the next frame to be read.

The output of the first reader is the input of the CCL algorithm. To process the incoming data, this block has previous RGB_to_GRAY and threshold filters created using SysGen. It introduces a small latency in the processing. Each pixel is labeled and, after that, saved using the third PIM in the second VFB, smaller than the previous one (because, although the number of frames to be saved is the same, the size of this frame is not the same (the frames were cropped before entering in the CCL algorithm)). VFB2 also has a lock procedure which makes that, while one frame is filled with the temporal labels, the previous one is being read using the last

[1] As 4-connectivity is being considered, the maximum number of labels that can appear in the image is Num_rows*Num_colum/2 (A chessboard pixel image).

[2] A BlockRam (BRAM) is a dedicated two port memory included on the FPGA fabric. Using this kind of memory instead of external memory, fast transfers can be achieved.

[3] All ports have the same static time to access to memory since the arbitration protocol is 'Round & Robin'. Writers VFBC PIMS are configured for using only command and write FIFOs whereas readers VFBC PIMS are configured for using only command and read FIFOs

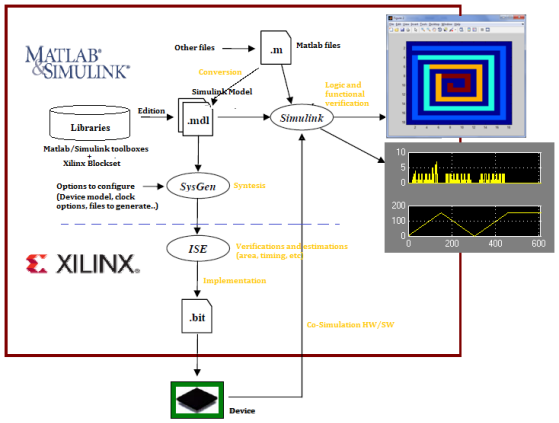

Fig. 6 Tools and methodology followed to design, verify and implement the algorithm

PIM configured as VFBC and go back to the CCL block to be the address signal of the table of equivalences. When the replacement has taken place, the pixel enters in dvi_out core to transform the signals to the format admitted by CH7301C standard. Finally, frames are displayed on a screen working at 27 MHz.

All the described IP modules are initialized and controlled from MicroBlaze processor. Frames from the experimental assembly are shown in Fig. 6.

In order to evaluate the goodness of this implementation, a software version of Bailey's CCL algorithm has been executed on a MicroBlaze soft processor core at 66 MHz. On average, the latter was only capable to process a VGA sequence at 6 fps. It was only able to reach 60 fps in some prepared cases or/and with low resolution frames.

V. CONCLUSIONS

An efficient implementation of a CCL algorithm and its integration in a complete image processing system are described in this paper. The algorithm takes advantage of the blanking periods in video standards and temporal parallelism to allow that the whole system can process VGA (640 x 480 P) @ 60 Hz video, improving in this way the results that are achieved with a software implementation that has the same properties.

ACKNOWLEDGMENT

This work was partially funded by Spanish Ministerio de Economía y Competitividad under the Project TEC2011-24319 and Junta de Andalucía under the Project P08-TIC-03674 (both with support from FEDER), and by the European Community through the MOBY-DIC Project FP7-INFSO-ICT-248858 (www.mobydic-project.eu) .

REFERENCES

[1] D. Nassini, S. Sahni, "*Finding connected components and connected ones on a mesh-connected parallel computer*", SIAM Journal on Computing, vol. 9, n. 4, pp. 744-757, 1980

[2] S.-W Yang et al, "*Parallel 3-Pixel Labeling Method and its Hardware Architecture Design*", Proc. 5th Int. Conf. on Information Assurance and Security, 2009

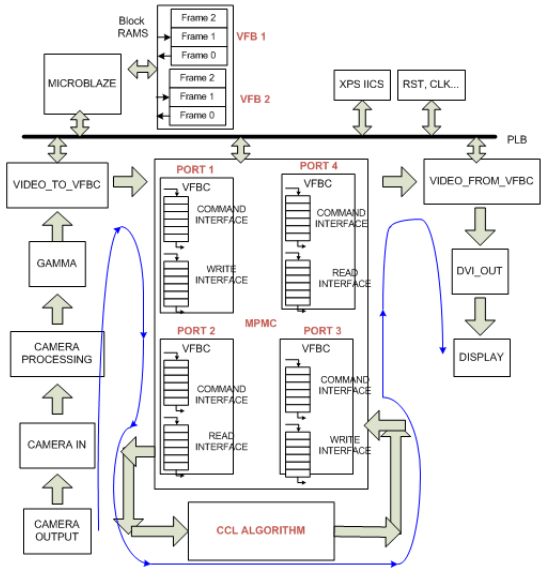

Fig. 4 Block diagram of the complete system

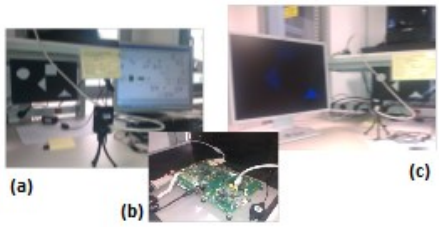

Fig. 5 Real complete system: (a) Camera capturing a reference photograph, prepared image and CCL system generator model (b) FPGA Board (c) VGA Output displayed on a LCD

[3] D. G. Bailey, "*Design for Embedded Image Processing on FPGAs*", Chapter 11: Blob detection and Labeling, Wiley, 2011

[4] C. T. Johnston, D.G. Bailey, "FPGA Implementation of a Single Pass Connected Components Algorithm", Proc. 4th IEEE Inte. Symp. on Electronic Design, Test & Applications, 2008

[5] F. Chang, C. Chen, "A component-labeling algorithm using contour tracing technique", Proc. Int. Conf. Document Analysis and Recognition, pp. 741–745, 2003

[6] D. G, Bailey, C. T. Johnston, "Single Pass Connected Components Analysis", Proc. of Image and Vision Computing, New Zeland, 2007

[7] R. M. Haralick, "Some neighborhood operations". M Onoe, K. Jr. Preston, A Rosenfeld. "Real Time Parallel Computer Image Analysis". New York: Plenum Press, pp. 11-35, 1981.

[8] K. Suzuki, I. Horiba, N. Sugie, "Linear-time connected-component labeling based on sequential local operations". Computer Vision and Image Understanding, vol. 89, pp. 1-23, 2003.

[9] L. Di Stefano, A. Bulgarelli, "A Simple and Efficient Connected Components Labeling Algorithm", Proc. 10th Int. Conf. on Image Analysis and Processing, IEEE Computer Society, Washington, DC USA.

[10] A. Rosenfeld, J. L. Plazt, "Sequential operator in digital pictures processing", Journal of ACM, vol. 13, n. 4, pp. 471-494,1966.

[11] L. He, Y. Chao, L. Suzuki, "A run-based two-scan labeling algorithm", IEEE Transactions on Image Processing, vol. 17, n. 5, pp. 749–756, 2008.

[12] K.Wu, E.Otoo, K.Suzuki, "Optimizing two-pass connected-component labeling algorithms", Pattern. Anal. Applic. 12, pp. 117-135, 2009

[13] E. Calvo-Gallego, P.Brox, S. Sanchez-Solano, "Implementación sobre FPGA de un algoritmo de etiquetado en tiempo real", XII Jornadas de Computación Reconfigurable y Aplicaciones (JCRA), Elche, Spain, pp 72-77, 2012