Architecture Based on FPGA's for Real-Time Image Processing

Conference Paper · March 2006
DOI: 10.1007/11802839_21 · Source: DBLP

CITATIONS
6

READS
71

5 authors, including:

Ignacio Bravo
University of Alcalá
79 PUBLICATIONS **418** CITATIONS
SEE PROFILE

Pedro Jimenez
University of Malaga
37 PUBLICATIONS **613** CITATIONS
SEE PROFILE

Manuel Mazo
University of Alcalá
217 PUBLICATIONS **2,544** CITATIONS
SEE PROFILE

José Luis Lázaro
University of Alcalá
130 PUBLICATIONS **908** CITATIONS
SEE PROFILE

Architecture Based on FPGA's for Real-Time Image Processing

Ignacio Bravo, Pedro Jiménez, Manuel Mazo, José Luis Lázaro, and Ernesto Martín

Electronics Department. University of Alcalá.
Alcalá de Henares (Madrid), Spain
{ibravo, pjimenez, mazo, lazaro, ernesto}@depeca.uah.es
http://www.depeca.uah.es

Abstract. In this paper an architecture based on FPGA's for real time image processing is described. The system is composed of a high resolution (1280×1024) CMOS sensor connected to a FPGA that will be in charge of acquiring images from the sensor and controlling it too. A PC sends certain orders and parameters, configured by the user, to the FPGA. The connexion between the PC and the FPGA is made through the parallel port. On the other hand, the resolution of the captured image, as well as the selection of a window of interest inside the image, are configured by the user in the PC. Finally, a system to make the convolution between the captured image and a nxn-mask is shown.

1 Introduction

One of the most important goals in vision-applications is the description of a certain scene in an automatic way. Description is understood as the localization and identification of the objects in the scene, depending on their features [1]. The main handicap in vision applications is the high computational cost that the algorithms that extract those features of the scene it implies. Nowadays, the large size images makes the number of operations needed increase considerably. Moreover, if real time performing is desired, execution-time of algothim must be as low as possible.

Usually, the platforms used to implement these algorithms are systems based on sequential programs. These are not, however, the most suitable elements for this kind of applications from the performance point of view, so, the search for new image processing systems is justified. In this sense, most of the vision systems can be divided in three levels, attending to the computational features: low level (such as filtered or convolutions), medium level (such as image segmentation) and high level (such as matching algorithms) [2].

It is, thus, important to select the necessary hardware platform depending on the complexity of the processing tasks. Since a conventional PC cannot carry out a bit operations concurrently, the system performance on a PC for image processing would be very poor. However, an ad-hock hardware design platform may overcome this problem. On the contrary, the kinds of operations to be done

K. Bertels, J.M.P. Cardoso, and S. Vassiliadis (Eds.): ARC 2006, LNCS 3985, pp. 152–157, 2006.

keeping high performance are relatively simple (filtering, image decompression, etc.). If it is desired to implement more complex algorithms, they must be reformulated in order to exploit the characteristics of the platform where they are implemented.

In this work a platform based on a FPGA and intended for vision applications is presented. It will be in charge of: capturing images provided by a high resolution sensor, applying a convolution with a mask over that image and further transmission of that preprocessed image to a PC by USB bus. The whole architecture is depicted in the figure 1.a).

The sensor chosen is the MT9M413, by Micron [3]. This monochrome sensor has as best feature the high speed that it can reach: 500fps with a 1280x1024 pixel-resolution. Its internal architecture permits connection with an external controller, allowing other operations to be carried out, as exposition-time configuration or choosing the desired window, by means of a set of control-signals.

The rest of the chapter is organized as follows: chapters 2 and 3 describe general remarks and a block diagram of the proposed design. Finally, a results section where time-relations and consumed resources inside the FPGA are shown.

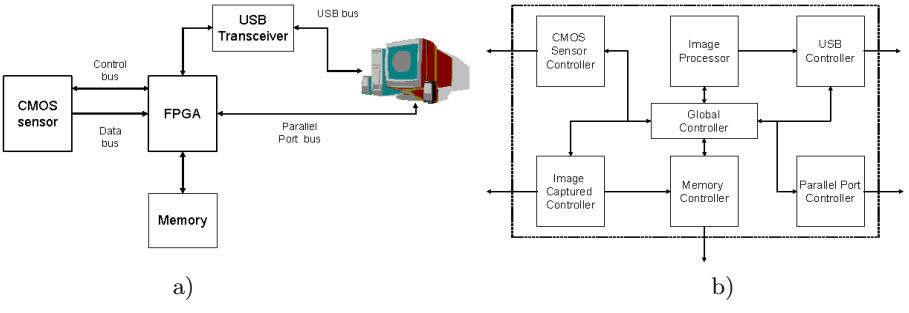


Fig. 1. a) Block-diagram of the developed real-time image-processing system. b) Block-diagram of the implemented system.

2 System Description

As it has been already said, one of the goals of this work has been the design of a specific architecture for image processing. The whole system is controlled by a FPGA of the Virtex II-Pro family, by Xilinx (XC2VP7) - aVirtex II Pro device has been chosen because of its versatility. This FPGA is mainly in charge of:

PC-parallel port of communication: It is only used to configure some parameters of the CMOS sensor (exposition time, window selection, status, etc). It is used used for this purpose due to its high speed and simplicity. Hence, an interface or parallel-port controller has been implemented in the FPGA which manages properly the data lines and the control lines of the parallel port. A PC will communicate with the FPGA through the parallel port

using a set of registers implemented in the FPGA. The global controller of the FPGA (see Fig.1.b) will be in charge of handle the information from/to each register.

Memory controller: The system includes 8 Mbytes of external SDRAM memory where captured images, and further processed ones, are stored. The main feature of this kind of memories is the need of continuous refresh. Other features of these memories are: data-bus width: 32 bits, maximum clock frequency: 125MHz, memory internal configuration: 4 banks x 512K x 32bits (8MB), manual refresh control and different access modes dependin g on the data-burst size to be read and written. The memories have $\overline{CAS}$, $\overline{RAS}$ and $\overline{WE}$ control signals to manage the writing, reading and refreshing processes. The memory controller has also been designed using VHDL.

USB controller: This other communication channel is used to transmit processed images. The FPGA has been connected to an external USB-transceiver, the GT3200 by SMSC, in order to send images at high speed. This solution provides a simple communication channel between the FPGA and the PC. The transceiver implements the USB 2.0 protocol so, the maximum data transfer speed rate could reach 480Mbps (60MBps). The interface designed in the FPGA has been developed from the one shown in www.opencores.org [4]. The maximum speed supported by this bus permits transfering images without any problems. For all these reasons, the use of this channel to send processed images from the FPGA to the PC is suitable.

Image processing: In this case, a convolution with a squared nxn-elements mask is made. The mask size is configurable in compilation time but its values are assigned by the user at execution time. This block can be replicated or made more complex as long as the process algorithms require or the FPGA resources allow.

Image-capture and CMOS sensor controller: The sensor used has some control signals that must be enabled from the FPGA. Besides, apart from the image processing block, and depending on the desired image size, an average or decimating of the captured image is made. Two solutions have been designed for our proposal in case that the size of the desired window is smaller than the maximum size:

a) The first one has been making a decimate (D) of value 8, 4, 2 or 1. This option is the simplest one to realize as the system keeps only one row and one pixel out of D rows and D pixels respectively. In this way, the final size of the image is reduced by a DxD factor. However, this choice presents the serious drawback of excessive aliasing.

b) The second option consists in implementing a module (binning block) that averages the samples that are desired to be decimated. If this choice is taken an average of all the decimated samples will be done so that a result affected by less aliasing is obtained, in respect with the previous case.

Both choices are selectable by the user. Its internal structure can be seen in figure 2.a).

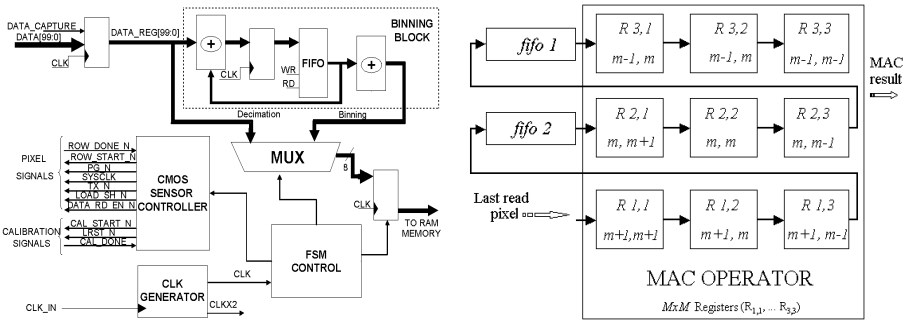


Fig. 2. a) Internal structure of the CMOS sensor controller and the image capturing system of the developed system. b) Convolution process between an image and a 3x3 mask.

3 Image Processor

Once the image has been captured, and a decimate or an average has been carried out, the image is processed. In this work, a system able to make a convolution a mask formed by the coeffients of $n \times n$ elements has been developed. These coefficients may be changed by the user through the parallel port. The maximum convolution-mask size depends on the resources of the FPGA and it must be specified during design time.

The new approach proposed is a block based on the use of some FIFO memories, which allows generating an output pixel at each clock cycle once the inicial latency has passed (see Fig. 2.b). The block stores every complete row of the image in a FIFO, so each pixel is ready to use when the calculation of the convolution for each row takes place [5].

This way the latency of the system is minimized and at the same time the use of the memory is optimized as data do not need to be stored to computed the convolution. As it is shown in Fig. 2.b, once the last pixel of the image has arrived multiplication by the last coefficient of the mask ($C_{m,m}$) is done, as well as the sum (first term in the sum in (1)) with the accumulated rest from the whole previous convolution operation (1).

$$R_{i,j} = P_{m,m} \cdot C_{m,m} \sum_{k=1}^{M} \sum_{l=1}^{m-1} C_{k,l} \cdot P_{i+k-m,j+l-m} \tag{1}$$

Where the second term in (1) is what it has been called accumulated result of the MAC-operation. The lapse of time since the the camera delivers the last pixel of the image, until the last convolution is performed generating the last result, is call *latency* (L_i), and is calculated as:

$$L_i = (X \cdot m + m + 1) \times T_{CLK} + W \cdot T_{CLK} \tag{2}$$

The first term corresponds to the time needed to process all the remaining points after the last pixel has been captured, and W is the processing-system depth. Here, the pixel capture-time, the multiplication and the sum are included. Finally, T_{CLK} is the period of the system clock-signal.

4 Results and Conclusions

In this section, an example of a captured and further processed image is shown, as well as an analysis of the resources and the time consumed by the system.

A 1024x1024 size image has been taken and a 4x4 binning has been carried out so that the final size becomes 256x256 pixels (see Fig. 3.a). Further, a convolution with an edge-detection specific mask is applied to the image (see Fig. 3.b). Thanks to the binning operation the image size can be efficiently reduced without loosing too much resolution. In this case the mask C is the 3x3 Laplacian operator, normalized to integer values. This mask is specially useful for image edge-detection.

In table 2 the results form the point of view of the resources consumed for a Virtex Pro (XC2VP7) with 4928 Slices are shown.

The maximum frequency allowed is over 100 MHz but, however, the global clock of the system runs at 100 MHz, so the system has been tested for this frequency. For this value the memory has 2 cycles latency.

Table 1. Consumed-resources summary of the binning-based design for a XC2VP7

Consumed slices 1084 slices (21 %)	
Capture:	130 Slices
Binning:	344 Slices
Convolution:	444 Slices
Max. freq:	166,66 M:z
RAM blocks:	6 (3 FIFO's)
Multipliers:	9 hardware multiplier

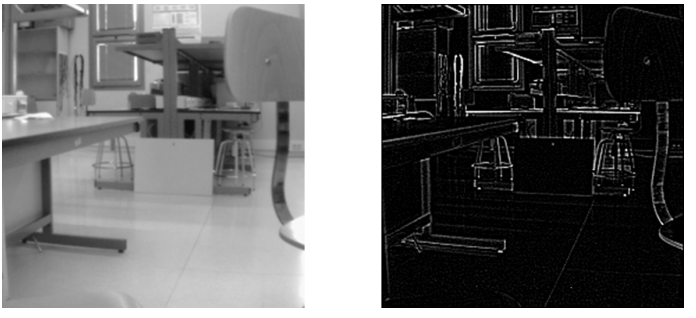
$$T = T_{capture} + L_{total} \quad , \text{ where } T_{capture} = 132 \times 1024 \times T_{CLKCAMARA} \tag{3}$$

$$L_{total} = L_{capture} + L_{binning} + L_{conv} = 10T_{CLK} + 4T_{CLK} + (\frac{n}{B}^2 + 4) \times T_{CLK} \tag{4}$$

Where L_{TOTAL} is the total latency time, being $T_{CLKCAMERA}$ the camera clock-period, n, the maximum square-matrix size (1024) T_{CLK} the FPGA clock period and B the binning factor. Hence, for $B = 4$ (256×256 size of the output image), $T_{CLK} = 100$ MHz and $T_{CLKCAMERA} = 10$ MHz, the system spends about 3.5 ms, what implies a total processing speed of 74 frames/s.

As conclusions, we may note that the architecture presented in this work has been designed to be used as base-platform for different artificial vision applications. The use of a high resolution and high frequency sensor, together with a FPGA, allows this platform to be used for many different algorithms. Also, a

very fast image-convolution system has been shown in this work. Finally, the PCA (Principal Component Analysis) algorithm is currently being developed based on this platform.



a) b)

Fig. 3. a) Image-size reduction by means of a 4×4 binning. b) Application of a 3×3 mask to the image on left side.

Acknowledgements

This work has been possible thanks to the project *SILPAR* project of Ministerio de Ciencia y Tecnología (ref: DPI2003-05067) and "Cátedra de control electrónico en transportes" founded by *LOGYTEL* and *RENFE*.

References

1. Ratha N.K., Jain A.K.. Computer Vision Algorithms on Reconfigurable Logic Arrays. IEEE Transactions on Parallel and Distributed Systems. Vol. 10, No. 1. (1999) 29-43.
2. Hamid G. An FPGA-Bases Coprocessor for Image Processing. IEE Colloquium on Integrated Imaging Sensors and Processing, 1994, pp: 6/1 - 6/4.
3. Datasheets 1.3 Megapixel CMOS Active pixel digital image sensor: MT9M413.
4. Usselmann, R. USB Function IP Core Rev 1.5. `www.opencores.org`. 2002
5. Bravo, I.; Hernandez, A.; Gardel, A.; Mateos, R.; Lazaro, J.L.; Diaz, V.; Different proposals to the multiplication of 3/spl times/3 vision mask in VHDL for FPGA's Proceedings of IEEE Conference on Emerging Technologies and Factory Automation, 2003. ETFA '03. Vol 2, pp:208-211. 2003