REAL TIME IMAGE PROCESSING WITH RECONFIGURABLE HARDWARE

Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, Juan A. Gómez-Pulido

Univ. de Extremadura. Dept. de Informática
Escuela Politécnica. Campus Universitario, s/n. 10071 Cáceres. Spain
E-mail: mavega@unex.es *Fax: +34-927-257202*

ABSTRACT: Digital image processing is a very important field within machine vision. Because of it is hard to implement real time image processing operations through software techniques, image processing field is one of the most active areas for reconfigurable computing. This paper introduces a new PCI-based system for real time image processing with reconfigurable hardware. The system uses the HOT2-XL PCI board, and we have implemented a Visual C++ application in order to validate our hardware designs. This environment is based on a library of hardware modules implementing some of the most common operations in image processing. The practical results show that our hardware modules get real time processing, a minimum resource use and a high operation frequency.

1. INTRODUCTION

Machine vision can substitute human vision for target tracking [1], robot guidance [2], and inspection tasks [3]. Usually, such systems extract information from an image following the steps: Image acquisition, image processing, feature extraction, decision making. The main challenge is that machine vision systems are normally used in real time applications. The use of a general-purpose processor does not usually allow real time processing. So, traditionally, high speed ASICs [4] have been required. However, it would have a limited functionality due to the predefined ASIC architecture. Reconfigurable (or custom) computing systems [5] deliver the benefits of hardware speed and software flexibility, because they can modify their architecture by the software to suit the application at hand. FPGA [6] has emerged as the natural platform for custom computing machines due to its reprogrammability. Furthermore, FPGAs provide significant price/performance benefits compared to ASICs.

From among the four steps already mentioned, we have focused on the image processing. Any situation requiring the enhancement, restoration, analysis, or creation of a digital image is a candidate for these techniques [7].

In this paper we present an integrated hardware and software environment dedicated to processing images in real time. The system has software processes to be executed by the CPU and hardware processes to be implemented into a FPGA, so it is based on the emerging field of hardware-software codesign [8]. We try to make a machine vision system, suitable for both scientific purposes and commercial applications. To do that, the goal we try to keep in mind is to maximize both flexibility and efficiency (high throughput) with a reduced cost.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the HOT2-XL board that is used as coprocessor for image processing. The following section presents our hardware module library. Section 4 describes the front-end application we have designed. Then, in section 5, we show and analyze detailedly the obtained results, indicating the advantages of the hardware/software co-design. Finally, in section 6, the conclusions and future work are indicated.

2. THE HOT2-XL BOARD

HOT2-XL board is one of PCI-interfaced, FPGA-based custom-computing boards offered by Virtual Computing Corporation. We have chosen a PCI board because of the properties of the PCI bus [9] and its growing popularity for image processing applications. We use the HOT2-XL board as a reconfigurable coprocessor aimed to image processing. The board has a XC4062XLA-09HQ240C FPGA, two fully independent 32-bit banks of RAM for a total of 4 MB, and a Configuration Cache Manager [10].

3. LIBRARY OF HARDWARE MODULES

The purpose of our research is the development of new architectures for image processing algorithms in real time using FPGAs. So, we are developing a library formed by hardware modules of very diverse types: point, histogram, convolution, mathematical morphology,... operations. Indeed, at the moment we have finished the following 16 modules: complement image (CI), binary contrast enhancement or thresholding (BCE), histogram (H), brightness slicing (BS), vertical gradient filter (VGF), horizontal gradient filter (HGF), diagonal gradient filter (DGF), low-pass filter (LPF), high-pass filter (HPF), laplacian filter (LF), histogram sliding and stretching or brightness/contrast adjustment (HSS), binary erosion (BE), binary dilation (BD), gray-scale erosion (GSE), gray-scale dilation (GSD) and median filter (MF). [7] gives more details about these operations.

The final purpose is to combine all these modules with software processes into a hardware/software co-design environment aimed to machine vision applications. The hardware modules will be downloaded and HOT2-XL board's FPGA quickly reconfigured as required by the applications.

Hardware modules are "programmed" by creating VHDL models and schematics which are designed, simulated, refined, and synthesized using Xilinx tools [11]. Every module is implemented using in each case the architecture that is best suited for real time operation and for the FPGA device we use. We have performed very diverse optimizations in each module: use of techniques of parallelism like replication and pipelining, optimization of the multipliers by means of adder trees in the convolution modules, reduction of the sorting and selection network in the median filter and the gray-scale morphological operations, search for a high regularity, reutilization of common resources, etc.

As example, figure 1 shows the implementation we have followed for the median filter. Because of we have 32-bit words (so, four 8-bit pixels by word) we have replicated the functional units, reusing common resources, in order to apply the median filter simultaneously on four pixel neighbourhoods and accelerate the operation four times. The computation of the four pixels (P1, P2, P3 and P4) is performed in two stages (pipelining). While stage 1 computes the pixels P1, P2 and P3 according to the three words read in the current clock cycle, stage 2 computes P4 and writes the resulting word associated with the three words read in the previous cycle.

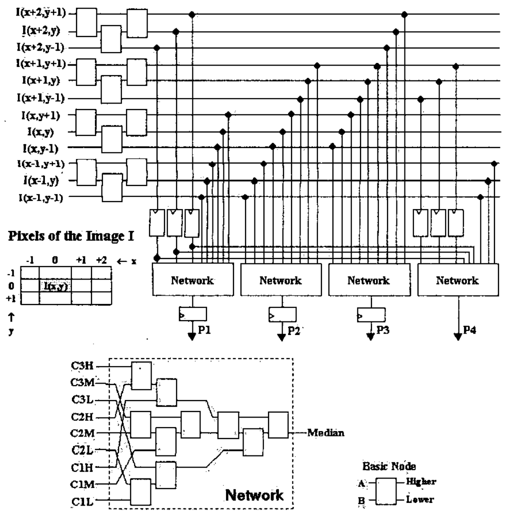
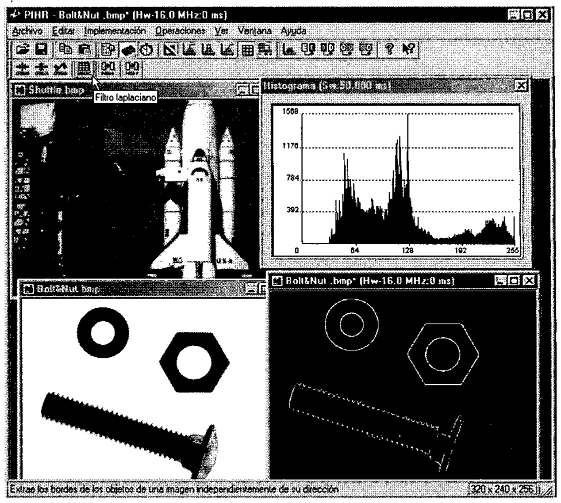


Figure 1. Implementation for the median filter.

4. THE PIHR APPLICATION

We have implemented the PIHR application (in Spanish "Procesamiento de Imágenes mediante Hardware Reconfigurable", Image Processing by Reconfigurable Hardware) in order to validate our library of hardware modules. PIHR operates on PC systems with Windows, and it has been written in Visual C++. The application offers a Windows typical graphic interface (see figure 2) and uses the HOT2-XL PCI board, our library of hardware modules and the HOT Run-Time Reconfiguration (RTR) method.



PIHR - Bolt&Nut.bmp* (Hw-16.0 MHz:0 ms)
Archivo Editar Implementación Operaciones Ver Ventana Ayuda
Shuttle.bmp
Filtro laplaciano
Histograma (Sw:50.000 ms)
1568
1176
784
392
0
64
128
192
255
Bolt&Nut.bmp
Bolt&Nut.bmp* (Hw-16.0 MHz:0 ms)
320 x 240 x 256

Extrae los bordes de los objetos de una imagen independientemente de su dirección

Figure 2. Graphic interface of the PIHR application.

User can load, store, copy, paste and visualize any bitmap (.bmp) image file. After selecting a window with the source image, user can choose how the image processing operation is done: using software (CPU) or hardware (HOT2-XL). In any case, the title bar of the resulting image window will include the number of milliseconds taken to do the operation, and the text "Sw" or "Hw". In the hardware version title bar, the clock frequency of the hardware module is also shown. So, for a functional test of our hardware modules we can execute the software version and then compare it with the image obtained from the FPGA. Furthermore, this allows us to compare the difference in performance of both versions. To find more general measurements, the application shows in the statusbar the dimensions of the processed image, as well as its number of colors. In this way, it is possible to obtain the number of pixels per millisecond.

The clock frequency of the hardware version can be changed (between 360 KHz and 100 MHz) via a dialog window, and so, we can probe the computation speed-up for hardware version with different frequencies; as well as other effects because of the increase or decrease of the clock frequency.

PIHR also allows the caching of hardware modules to reduce the configuration overload of frequently used modules. For that, it uses the on-board Configuration Cache, managing a list of the hardware modules in cache. When the Configuration Cache is full, the application replaces the hardware modules using a LRU (Least Recently Used) policy. In conclusion, reconfiguration time of the system with a different hardware module is reduced because it will probably be in cache.

5. RESULTS

Table 1 shows the results obtained for the different hardware modules. The operations are sorted out from less to more execution time for the software version, therefore, from less to more complexity.

Operation[1]	Maximum Frequency (MHz)	Average Execution Time (ms)		Resource Use (CLBs)
		SW	HW	
CI	32.303	150.391	574.766	41 (1.78%)
BCE	31.169	457.422	579.063	71 (3.08%)
H	33.161	464.453	1578.672	73 (3.17%)
BS	33.858	533.984	581.016	99 (4.30%)
VGF	31.426	858.203	581.016	65 (2.82%)
HGF	32.633	862.109	718.125	105 (4.56%)
DGF	32.449	882.422	717.734	109 (4.73%)
LPF	30.000	1333.984	867.734	238 (10.33%)
HPF	32.853	1660.547	868.516	322 (13.98%)
LF	33.638	1712.109	867.344	290 (12.59%)
HSS	30.942	2633.203	718.516	1420 (61.63%)
BE	32.519	2809.766	868.906	163 (7.07%)
BD	32.519	3362.500	868.906	163 (7.07%)
GSE	17.490	8162.109	868.906	791 (34.33%)
GSD	17.182	8479.688	869.297	791 (34.33%)
MF	16.162	84459.766	998.203	634 (27.52%)

[1] See section 3 for the abbreviation of each operation

Table 1. Experimental results for the different operations.

Second column presents the maximum frequency admitted by each of our hardware modules. The quantities presented in this one and the fifth column have been obtained from the reports generated by the Xilinx Foundation Series tools [11] after the synthesis of each hardware module. Therefore, they are real measurements on implementations already carried out, and not estimations.

Observing the table we conclude that almost all the hardware modules admit a maximum clock frequency about 33 MHz, coinciding with the PCI bus specifications for a 32 bit/33 MHz system, and therefore, an 132 Mbytes/sec. bandwidth [9].

It is necessary to highlight the gray-scale erosion and dilation operations, as well as the median filter. Their maximum frequency decreases significantly regarding the rest of modules. This fact is fundamentally due to they include a pixel sorting and selection circuit. This type of circuits has a great quantity of logical levels, generating a great quantity of propagation delays (logic and routing delays). These increase the minimum clock period, and therefore, decrease the maximum frequency.

Third and fourth columns show a comparison between the hardware and software versions. In both columns, the average execution time is indicated for each operation on 30 images of 640x480x256. With a 16-MHz clock, in the HOT2-XL board, the time to process 30 images oscillates between 574.766 and 1578.672 ms.The reconfiguration time of the board (440 ms) is included in these times. Notice that if we use the same operation repeatedly, the board would only be configured the first time.

The same operations, written in Visual C++ and compiled with the appropriate optimizations, require between 150.391 and 84459.766 ms in a PC with a 350 MHz Pentium II processor and 64 Mb of RAM. Therefore, the hardware implementation represents an important improvement over the yield of the software versions except for the complement image and the histogram. The complement image is a very simple operation, for what the hardware implementation is not advantageous due to the reconfiguration time overload. On the other hand, the histogram is the only operation whose hardware version possesses clearly a low yield. The reason is its implementation, which carries out a total of 8 sequential read/write operations for each word (32 bits) of the input image. This implementation is forced by the HOT2-XL architecture. Mainly for the number of fully independent memory banks that the board has, forcing to treat multiple memory operations in a sequential way (one after another), not taking advantage of the inherent parallelism.

It can also be observed that the improvement produced with the hardware implementation increases as the operation is more complex, obtaining a time up to 85 times smaller than the respective software version (median filter). Note that the HOT2-XL is running at a clock rate approximately 1/22 (16 MHz) that of the microprocessor (350 MHz), and the clock of the hardware versions could be changed to larger frequencies, inside the limit indicated in second column, if it is necessary to obtain a better performance.

If we intend to process images in real time (30 images per second), and we suppose that a processing operation by image is only applied, we must admit a maximum time of 1000 ms. Table 1 shows that more than the half of the operations in software version can not reach this speed. In fact, the software only gets real time for the simplest operations. However, the real time processing is possible using our hardware modules except for the histogram operation, whose implementation has been adversely affected by the board architecture.

The overload due to the reconfiguration time has great importance in the shown data. For this reason, we have used the Configuration Cache included in the HOT2-XL. If a hardware module was already loaded in the Configuration Cache the average reconfiguration time would decrease to 270 ms. Therefore, all the operations would have in their hardware version a significantly lower average execution time to the one shown in the table.

In conclusion, the more operation complexity the more hardware improvement. On the one hand, the reconfiguration time influences less. On the other hand, the performance gain produced by the exploitation of the parallelism (replication and pipelining) in the hardware versions becomes more notorious. Table 1 shows clearly the advantages of the hardware/software co-design. The simplest operations (complement image) would be implemented in software, while the most complex ones would be performed in hardware exploiting its higher yield.

In fifth column we show the use of resources of the FPGA expressed by number of CLBs (Configurable Logic Blocks), knowing that the XC4062XLA FPGA has a total of 2304 CLBs. The percentage of CLBs used of those 2304 is also indicated. It should be kept in mind that each module (except the histogram) has its functional units replicated four times to take advantage of the HOT2-XL board architecture (in a 32-bit word there are four 8-bit pixels). This fact multiplies the use of resources by four. Moreover, each module also includes the circuitry required to the internal management of the on-board memory. In spite of everything, the designed hardware modules present an efficient use of the FPGA resources. This low use of resources is useful for future lines of work, with the intention of combining several hardware modules inside the FPGA at the same time, avoiding intermediate reconfiguration times. In this way, pipelines could be designed where each stage is an image processing operation.

The HSS module highlights because it uses a 61.63% of the FPGA resources. The reason is that it includes an integer pipelined multiplier and an integer pipelined divider, and both circuits are enormously complex and need a great quantity of resources. This fact shows the high benefits of the implementation followed for the necessary multipliers in the convolution operations (by means of adder trees, shifts, etc.). Notice that operations like the laplacian filter, the low-pass filter, the high-pass filter,... present a very lower use of resources, although they also include multiplications (or even divisions). Also, it is observed that the circuits with similar functionality, and therefore complexity, use the same number of resources (erosion and dilation).

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a PCI-based real time image processing system. The system can be reconfigured in run time to adapt it to the requiriments of an application, carrying out several different operations per execution. To do that, we used reconfigurable hardware, and more concretely, the HOT2-XL board. In order to configure the HOT2-XL a library of hardware modules is being implemented. When the library be completed, it could be reused by other designers saving cost, design time, etc.

The PIHR application is being implemented as a mix of software and hardware, so it is based on in the emerging field of hardware/software co-design [8]. This adds a great deal of flexibility to the development process, being possible to explore the trade-offs between hardware and software implementation of application ideas.

The presented results illustrate the efficacy of FPGA-based reconfigurable systems to image processing applications, showing a great speedup over software versions as the operations are more complex. Then, our system will help us to relax the constraint on the number of operations in a given real time application, thus allowing the implementation of more complex algorithms. The practical results also show the effectiveness of the followed architectures and optimizations: use of techniques of parallelism like replication and pipelining, reutilization of common resources, etc. Everything has allowed us to get real time processing, a minimum use of resources and a high operation frequency.

Now, we are investigating the use of this system as part of an industrial inspection system, more concretely, for the evaluation of the cork stopper quality applying digital image processing techniques, with the benefits of hardware speed, as well as flexibility and price/performance ratio of FPGAs.

7. REFERENCES

[1] D. Geman and B. Jedynak. "An Active Testing Model for Tracking Roads in Satellite Images". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1), January 1996, pp. 1-14.

[2] M. Bertozzi and A. Broggi. "Vision-Based Vehicle Guidance". *IEEE Computer*, July 1997, pp. 49-55.

[3] A.D.H. Thomas, M.G. Rodd, J.D. Holt and C.J. Neill. "Real-Time Industrial Visual Inspection: A Review". *Real-Time Imaging*, vol. 1, 1995, pp. 139-158.

[4] M.J.S. Smith. "Application-Specific Integrated Circuits". Addison-Wesley, 1997.

[5] J. Villasenor and W.H. Mangione-Smith. "Configurable Computing". *Scientific American*, 276(6), June 1997, pp. 54-59.

[6] S. Brown and J. Rose. "FPGA and CLPD Architectures: A Tutorial". *Proc. IEEE Design & Test of Computers*, 13(2), Summer 1996, pp. 42-57.

[7] G.A. Baxes. "Digital Image Processing. Principles and Applications". John Wiley & Sons, 1994.

[8] G. de Micheli and R.K. Gupta. "Hardware/Software co-design". *Proc. IEEE*, 85(3), March 1997, pp. 349-365.

[9] PCI Special Interest Group. "PCI Local Bus Specification - Revision 2.1". http://www.pcisig.com, June 1995.

[10] Virtual Computer Corporation. "H.O.T. II Hardware Guide. Version 2.0". 1999.

[11] Xilinx Inc. http://www.xilinx.com. 2001.