Research Article
Smart Camera Based on Embedded HW/SW Coprocessor

Romuald Mosqueron,[1] Julien Dubois,[2] Marco Mattavelli,[1] and David Mauvilet[1]

[1] GR-LSM, Faculté STI, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH 1015 Lausanne, Switzerland
[2] Laboratoire LE2I, Faculté des Sciences Mirande, Université de Bourgogne, 21000 Dijon, France

Correspondence should be addressed to Romuald Mosqueron, romuald.mosqueron@epfl.ch

This paper describes an image acquisition and a processing system based on a new coprocessor architecture designed for CMOS sensor imaging. The system exploits the full potential CMOS selective access imaging technology because the coprocessor unit is integrated into the image acquisition loop. The acquisition and coprocessing architecture are compatible with the majority of CMOS sensors. It enables the dynamic selection of a wide variety of acquisition modes as well as the reconfiguration and implementation of high-performance image preprocessing algorithms (calibration, filtering, denoising, binarization, pattern recognition). Furthermore, the processing and data transfer, from the CMOS sensor to the processor, can be operated simultaneously to increase achievable performances. The coprocessor architecture has been designed so as to obtain a unit that can be configured on the fly, in terms of type and number of chained processing stages (up to 8 successive predefined preprocessing stages), during the image acquisition process that can be defined by the user according to each specific application requirement. Examples of acquisition and processing performances are reported and compared to classical image acquisition systems based on standard modular PC platforms. The experimental results show a considerable increase of the achievable performances.

1. INTRODUCTION

Nowadays, *smart* cameras are more and more applied for their specific performances and their processing capabilities in different application fields. We can distinguish three typical classes of *smart* cameras.

(i) *Artificial retinas:* in which dedicated processing is directly integrated aside the pixel. The processing capabilities are usually fixed or limited to a few simple and local functions [1–3].

(ii) *Standard cameras directly connected to a computer via a standard interfaces:* all the processing is performed into the computer CPU [4, 5].

(iii) *Cameras including embedded processing units:* the processing is performed into the camera and only the processing results or some image features are transferred outside the camera [6, 7].

For the class of the artificial retinas cameras, the image processing capabilities, or better the pixel imaging capabilities, are usually fixed locally to a small pixel neighborhood and remain very limited in scope. No application specific processing can be added at the image acquisition stage. However, such kind of sensors can achieve very high frequency acquisition rates that are necessary for some class of applications. In the case of a standard camera interfaced with a computer, the data transfer between the camera and the computer is limited by the connection interface. When dealing with applications requiring high frame-rate or very high resolution cameras, usually the problem is the amount of data that needs to be transferred to the CPU. This may largely exceed the available standard interface bandwidth. For such class of applications, different camera architectures that include embedded processing units have been developed.

This paper proposes a novel smart camera architecture based on a specific coprocessor designed for on-line industrial process control. This paper describes a coprocessor unit design providing an interface for the full control of the sensor acquisition process driven from the main application CPU. This key feature enables the acquisition to be controlled on the fly in function of the algorithm's scheduling. The main processor and the coprocessor are, respectively, in

charge of the high-level tasks, the acquisition and processing decision imposed by the application, and the lower-level tasks, characterized by high level of processing regularity and parallelism. The processing can be adapted easily, thanks to the structure, to the application's requirements.

The paper is organized as follows. Section 2 presents the context and the objectives of the new embedded system. Section 3 presents the coprocessor platform. The coprocessor architecture is presented in Section 4 and its features are discussed in detail. The performance of the coprocessor architecture are reported in Section 5 and compared to a classical image acquisition and processing scheme. Results of a complete postal sorting application are presented in Section 6 showing the potential parallelism of this platform. Finally, Section 7 concludes the paper presenting the perspectives of further work.

2. CONTEXT AND COPROCESSOR INTO PROCESSING/ACQUISTION LOOP

For image-processing applications requiring very high-performances, an adaptive image acquisition stage is very often the key feature to satisfy the real-time constraints. Although we can nowadays observe the wide availability of low-cost high-speed high-resolution sensors, the high pixel rate to be transferred to the central processing unit from the image sensor is often the main system bottleneck in terms of performance. This fact indeed pushes the theoretically achievable system performance to higher and higher levels so as to be able to cover new demanding applications. However, higher pixel rates require new architectural approaches so as to reduce the costs of the interfacing and processing stages that are now the real bottleneck of such systems. Whenever such high pixel rate can be reduced according to the analysis of its intrinsic semantic content (i.e., image portions can be discarded not being relevant for the application), the response time of common system architectures is too slow to timely adapt the acquisition stage to the relevant image sequence content. Indeed, the transfer time from the sensor to the CPU unit is often too large to enable the system to react, and finally results in being too expensive in terms of equipment and interfaces.

The coprocessing approach has been investigated in the last few years by several authors. Some works presented in literature are based on hardware coprocessing designs specifically dedicated to a single application [8–10]. The performance improvements reported in literature are quite relevant, when comparing architectures with or without coprocessor, those results present speed-up factors up to few hundreds. Other authors have proposed generic systems whose property is the possibility to implement different algorithms on a coprocessing-based architecture [11]. In the class of "generic" coprocessor units, only a few authors have mentioned the possibility to control the image acquisition stage simultaneously with the processing stage. Gorgon proposed a coprocessor unit to control the acquisition stage of charge coupled devices (CCDs) sensor [12]. Jung et al. presented a preprocessing unit to control CMOS sensor [13], but the achieved functionality operates only on the specific

image corrections used to compensate physical limitation of the CMOS sensor. The key point is to control the acquisition on the fly in function of the different algorithm tasks.

The integration of a coprocessing element into the image acquisition loop of a CMOS sensor has very interesting features. Standard CCD-based image systems are synchronous and require that the full image is downloaded before proceeding to a new acquisition. CMOS sensors are much more flexible because not only they are intrinsically asynchronous, but they are also capable of performing image acquisitions on limited section of the sensor up to the acquisition of single pixels. For several applications such flexibility can be successfully exploited so as to reduce the data transfer to the central CPU thus considerably reducing the necessary data bandwidth. As a consequence, the overall processing requirement of the application has just to process a limited portion of the original image. The key to achieve such results is to be able to provide to the main application the necessary information to adapt the acquisition stage without the need to transfer the full image to the central CPU. In other words, CMOS imaging can achieve the following:

(i) a selective image acquisition stage depending on the image content itself and on the requirements of the application,

(ii) a relevant reduction of the data volume to be transmitted to the processing unit once the selective acquisition stage has been activated.

The condition for which such features can be achieved is that a "coprocessing" element is inserted in the image acquisition loop driven by the "high-level" application. Different approaches can be considered; the implementation of an embedded ASIC or a configurable processor is one of them. The processing capabilities of these components, as the STV0676 (ST Microelectronics: STV0676 datasheet; available at http://www.datasheetcatalog.org/datasheet/stmicroelectronics/9068.pdf) or the CMOS coprocessor introduced by [14–16], are examples. However, the level of flexibility of such architecture is quite limited since only a few processing parameters can be configured according to the application constraints. All the components are fixed with their own processing. A recently designed chip with a coprocessor is the named OMAP DM-510 (Texas Instrument: OMAP DM-510 page; available at http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=12802&contentId=41258). Such component is very interesting, but only addresses low-power applications, typically mobile phones. Moreover, all the chip components have dedicated interfaces, thus the control of specific CMOS is very complex or even impossible, for the number of control signals to handle. For instance, the high-speed CMOS sensor MT9M413 (Micron-Aptina: MT9M413 datasheet; available at http://www.aptina.com/products/images_ensors/mt9m413c36stc/#overview) required almost 150 pins to be implemented (included data). The "artificial retina" approach has similar features in terms of processing possibilities [17]. These architectures are frequently developed

to process a neighborhood of the pixels [1–3], however the process is usually reduced to small size neighborhoods (usually smaller than 16×16). The acquisition can be controlled, but is dependent on the architecture and the targeted applications. So as to obtain the features presented above, a heterogenous architecture is proposed: a processor associated to an FPGA. The acquisition is a task of the FPGA, therefore any CMOS sensor can be controlled. Consequently, any CMOS sensor with specific acquisition modes (as, e.g., scan line) can be interfaced. The flexibility of the heterogeneous structure offers a large panel of solutions for the implementation of image processing for on-line industrial control.

In such architecture the "coprocessing" unit besides the control of the acquisition stage becomes naturally in charge of the standard low-level repetitive tasks such as filtering, denoising, and binarization. In fact, the full control of the acquisition stage enables the right control of the preprocessing tasks usually performed at the level of the central CPU or high-level application.

For instance, the "instructions" for a selective image acquisition stage, that is, an acquisition stage for which only a (small) portion of the image that presents certain features needs to be "acquired" and transmitted to the central CPU for further high-level processing, are handled by the "coprocessor" accessing directly the CMOS sensor itself in an asynchronous manner. At this point also the processing associated to the specific feature "found" in the image can be efficiently implemented at the "coprocessor" level. Then, only the "selected" image portion already preprocessed and/or prefiltered is transferred to the central CPU unit. The coprocessing task schedule can be selected on the fly depending on the acquisition commands and is adapted to the acquisition form that is region/pixel-based. By this approach, the necessary data bandwidth can be drastically reduced eliminating in most of the cases the major system limitation. An example of achievable performance for some classical preprocessing stage is provided in Section 5. The main processor, freed from image acquisition and preprocessing tasks, can then be used for further processing and/or high-level algorithms defined by the specific application.

The challenging aspects of the coprocessor design are mainly related to the variable acquisition mode (i.e., input image format and layout). Obviously, the bandwidth associated to a window processing can be optimized; moreover, the nature, the complexity, and the number of possible processing stages can be adapted at each acquisition mode. Many different acquisition modes are then available. In all modes, a window can be selected in the full-range image, the size and the integration time are defined, and a subsampling (on Y and X) can also be specified. In the simple multiexposition mode, the same window is acquired several times or periodically and the delay between two acquisitions can be defined. Also in the tracking multiexposition mode, the window can be translated. Such modes allow to create a "subimage" image by row or column accumulation when the sensor is used as line sensor even with lines varying their position during the acquisition itself.
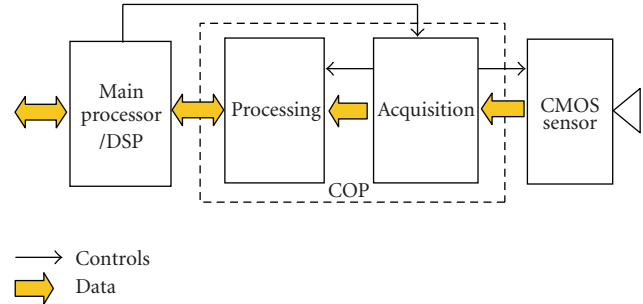

FIGURE 1: Block diagram of the coprocessor-based architecture.

The first interesting result achieved by implementing this architecture is that relevant speed-up factors are obtainable for reconfigurable processing modules, thus providing enough flexibility in terms of choices of processing and in terms of acquisition modes defined on the fly by the application itself (selection and preprocessing of any kind of area of interest). The second interesting result is that such on-the-fly adaptation of the acquisition mode yields a further bandwidth reduction for the transfer of the image data to the central CPU. This feature represents for some application a further speed-up in the overall system performance in terms of reduction of processing or increase of the achievable acquisition/processing frame rate.

3. ARCHITECTURE OF THE COPROCESSOR CAMERA

The overall system can be described as an autonomous intelligent camera with powerful embedded processing when compared with modular systems associated with a computer.

The system has been thought for monitoring applications such as road monitoring [18] or intrusion detection or any other similar application. Quality control and control of industrial processes, where very high frame-rate on specific image sections are required, is another application field of the system. For such kind of processes, only the "relevant" portions of the images are necessary to be transmitted to the host CPU for further processing. In some cases, only the result of the preprocessing, or of the processing (i.e., the detected feature), is needed to be transmitted outside the system to a local host PC or via Internet.

The system is composed of an embedded frame-grabber equipped, at different levels, of processing capabilities for the image acquired by the sensor and it is illustrated in Figure 1. This figure illustrates the main architectural components of the camera with embedded coprocessing stage. In this architecture, the coprocessor part (COP) is divided in two parts: a part dedicated to the acquisition and a second part dedicated to a preprocessing stage.

The association of the processing and acquisition stages aims at reducing the pixel rate for applications where "irrelevant" image portions are detected by the coprocessor. The processing is then complemented by the processor for higher-level tasks at a possibly lower pixel-rate. The partition

of the tasks is made by exploiting the specificity of each element, to use it as efficiently as possible, thus reducing the pixel-rate when possible and the processing time so as to increase the overall throughput. This architectural approach to the processing of sequences is particularly adapted to and performing, but not limited to, tracking applications, pattern recognition applications, and compression applications. Video compression is generally used in camera systems so as to reduce the bandwidth of the data transfer and to be able to use a standard communication channel without addition of acquisition boards such as the camera-link for instance. However, with high performance sensors, there is immediately the problem of the connection that becomes now the system "bottleneck", and prevents from transferring the images rate provided by the sensor. The system described in this paper also supports the implementation of a compression stage (thanks to multimedia processor's functionalities) that makes it possible to approach to the sensor limit capabilities.

The system is composed of a compact stack of 4 boards, enabling to easily interface various types of sensor/cameras and thus answering to various resolution and acquisition speed requirements in the most modular and economic way. The four boards described hereafter are the following:

(i) the motherboard containing the main processor,

(ii) the communication board,

(iii) the board including the coprocessor,

(iv) the camera interface board.

3.1. Motherboard

The motherboard contains a Nexperia PNX1500 (Philips: PNX 1500 datasheet; available at http://www.nxp.com/acrobat_download/literature/9397/75010486.pdf) processor at 300 MHz and includes functions for the sound and image processing. This processor has been selected for the powerful VLIW core and for the variety of supported integrated interfaces such as Ethernet controller, DDRam controller, and PCI. Moreover, it includes a 32-bit TriMedia 3260 CPU core with 5-issue slot engine and 31 pipelined functional units. It operates up to 10000 Mops and 1300 Mips and can execute up to 5 operations per clock cycle. In addition, Nexperia integrates a graphic 2D engine able to display up to a resolution of 1024×768 to 60 Hz. A large library can be used to program several standards (MPEG, H263, etc.) and several interfaces. The Nexperia power consumption depends on the processing charge, typically it is around 1.5 W. Around this processor, we can find communication interfaces such as Ethernet and ISDN, as well as acquisition and rendering of video images and analogical sound. The motherboard contains a 64 Mbytes DDR Ram (333 MHz) to store data. In addition, there are 32 Mbytes of flash memory used to store the programs or different information. Several interface components can communicate via a PCI bus, like Ethernet and VGA. Thus, motherboard contains an internal bus which is a PCI and all the platforms communicate via this bus.

3.2. Communication board

The second board is based on an FPGA Spartan XL (Spartan XL (Xilinx: Spartan XL datasheet; available at http://direct.xilinx.com/bvdocs/publications/ds060.pdf)) to manage the PCI arbiter, the communication interfaces such as USB2.0 and Firewire that can be driven to connect the camera with digital standard interfaces. This board provides four functionalities. It extends the communication of the motherboard with standards USB 2.0, IEEE1394, and Ethernet 10/100. The ethernet connection is important, because systems and a computer can communicate by this interface with a simple IP number. It also provides a centralized power supply for the system. On this board, a converter N/A is used to display on a standard VGA monitor. All the components cited before have a PCI input to avoid different link with the processor.

3.3. Acquisition and processing board

This board is the COP part in charge of the acquisition and the preprocessing stages of the video signal coming from the CMOS sensors or cameras. The preprocessing part is independent from the acquisition part. Its architecture is illustrated in Figure 2. The main functions are partitioned into two FPGAs. The first FPGA is a Virtex2Pro VP4-fg456-5 (Xilinx: Virtex 2 Pro datasheet; available at http://direct.xilinx.com/bvdocs/publications/ds083.pdf), their functions are to acquire images and communicate the configuration and orders to the camera. This FPGA drives the camera and can be adapted to several sensors or cameras; it is just an implementation with the right driver. Therefore, it implements a wide variety of acquisition modes (random region acquisition, variable image size, variable acquisition modes line/region based, multiexposition image). The second FPGA is a Virtex2Pro vp20-fg676-5[3], and high-performance image preprocessing (calibration, filtering, denoising, binarization, pattern recognition) is its principal function. These FPGAs include, respectively, 1 IBM Power PC, 28 18 kb BRAMs, 3008 slices and 2, 88, 9280 for the large one. Both FPGAs communicate through two high-speed serial channels, specific to Xilinx, called RocketIO (Xilinx: RocketIO User Guide; available at http://www.xilinx.com/bvdocs/publications/ug024.pdf). Moreover, each FPGA communicate with the processor via the PCI bus. Xilinx FPGAs have been chosen, because of their high-speed serial communication, PowerPC are available, and they are flexible and reprogrammable. Interface drivers are already developed and optimized by Xilinx.

This board contains, in addition of FPGAs:

(i) 2 SDRAM until 128 Mbytes associated with the acquisition FPGA used to store several images,

(ii) 2 ZBT until 8 Mbytes associated with the coprocessing FPGA used for processing tasks,

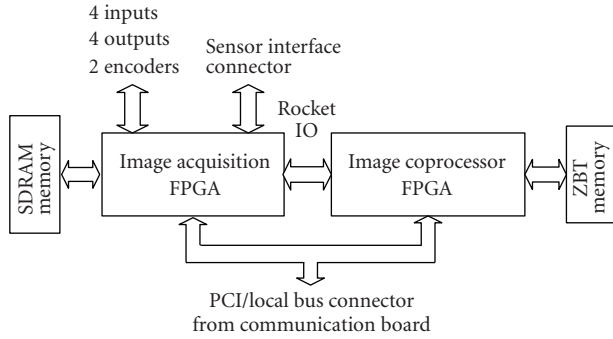(iii) 4 optocoupled inputs,

(iv) 4 optocoupled output,

(v) 2 encoders.

FIGURE 2: Block diagram of acquisition and processing board architecture.

FIGURE 3: Block diagram of the system architecture.

Power consumption of this board also depends on the processing performed. The power dissipation ranges from 3 up to 12 W depending on FPGA and memories utilization. For example, in the implemented application, explained Section 6, power dissipation is below 7 W.

3.4. Camera interface board

The fourth board is a simple interface board between the FPGA board and the camera. This board depends on the camera interface. It can be adapted and changed. Thus, the system can be used with several CMOS image sensors, for the results described in this paper, a sensor IBIS4-1300 (Cypress: IBIS4-1300 datasheet; available at http://www.datasheetcatalog.org/datasheet2/2/044ipph9289eg3l6qa3eadzds1fy.pdf), with a resolution of 1280×1024 pixels and a 40 MHz pixel frequency has been used for the experimental results. Its interface is a Camera Link.

The main boards communicate through bus PCI v2.2 allowing to transfer a large number of data (up to 133 Mbytes/s) to the host processor (in accordance with Nexperia bus interface). The entire architecture is illustrated in Figure 3, with processing components and available interfaces.

However, the main idea of the system architecture is indeed to reduce as much as possible the data rate after the coprocessor unit by transmitting only the processed image sections or by controlling the acquisition.

This architectural solution provides exceptional processing potential and offers wide communication possibilities. The processing part is built with pipelined or parallel HW processing modules to obtain high performance. Furthermore, a processing and data transfer, from CMOS sensor to processor, can be operated in parallel so as to increase performances.

The main additional advantages of this system, besides the capability of controlling the acquisition loop and the achievable processing performances compared to a traditional modular PC system, can be summarized as follows:

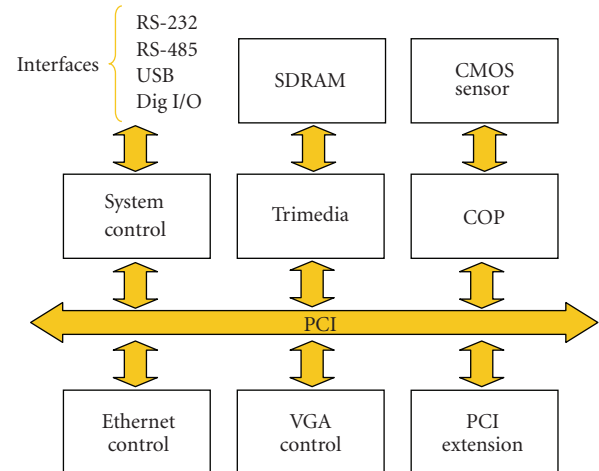   (i) a low dissipated power,

   (ii) compact dimensions,

   (iii) a greater robustness (mean time between failures) because it does not integrate mobile components (ventilators, hard disks),

   (iv) a greater commercial lifespan because components in the computers world are very volatile and cannot be replaced with components having the same characteristics. Sometimes, after only a few years, partial redesign of the system is required to critical applications.

Power supply of the system is operated at 12 V and can reach up to 36 W. In the test case applications, it works at about 20 W, including the power supply of the CMOS camera. Dimensions of this system are $150 \times 150 \times 60$ mm. In future versions, dimensions will be reduced and the internal communication bus could be directly replaced by the PC104 or another similar performing bus.

4. COPROCESSOR DESIGN

The essential problem of the coprocessor architecture is the tradeoff between processing efficiency and flexibility required to exploit the CMOS potential features. Two different parts essentially constitute the COP architecture (Figure 4): the processing and the acquisition parts. The functional blocks constituting the processing part are a processor interface (PCI interface), a command controller, a processing controller, a processing unit, and an SRAM. Thus, the COP architecture is essentially constituted by the following functional blocks (Figure 4):

   (1) a processor interface (bus interface),

   (2) a bus bridge, a command controller,

   (3) a processing controller,

   (4) a processing structure,

   (5) a CMOS sensor interface.

The command controller receives the acquisition commands, the processing commands from the main application. The
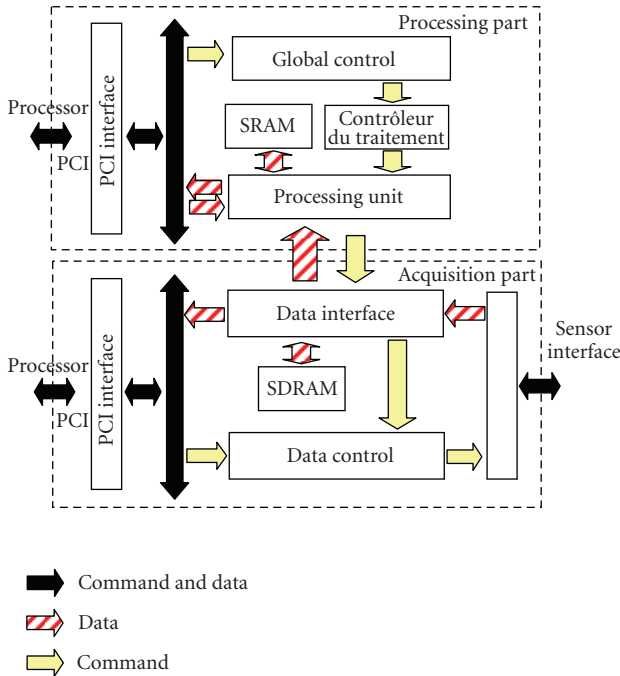
Command and data

Data

Command

FIGURE 4: Block diagram of the COP architecture.

task scheduling is controlled by the processing controller and is executed by the processing structure unit configured according to the received commands. The data and image portions, provided by the main CPU and used by the coprocessor for the actual processing tasks, are transferred to the processing structure via the bus bridge and via the processing controller. This feature enables to implement a true coprocessing stage and not a simple preprocessing.

The link between sensor and acquisition part is specific for each image sensor, consequently it should be modified after any sensor change. The connection between acquisition and processing is standard, therefore independent of the sensor. Acquisition commands are constituted of parameters defined to cover a large number of acquisition modes to enable to interface a large sensor sort (linear CCD, CMOS matrix). Eventually, the connection with coprocessor and processor are linked with standard PCI. Hence, the coprocessor is independent of the processor and could be used as embedded IP with any PCI system. The coprocessor architecture enables a full data rate to be obtained on PCI bus.

The possibility to adapt the number and nature of the processing and to operate on variable size/shape images is provided by the flexibility of the processing structure unit.

In essence, it is constituted by five different components (Figure 5): CONTROL_MEM is in charge of the main memory, CONTROL_PRO is in charge of the processing control, the processing modules, the system control, and the FIFO is in charge of the temporary storage. Such architecture implements several options for the data flow control (Figure 5). The input data, provided by CMOS sensor and by the processor, are referred to in Figure 5, respectively,

with the numbers 1 and 3. There is no FIFO in 1 since there is a memory in the CMOS interface. The broadcasting nets referred to as 2, 4, and 5 allow to copy the data and transfer them on each output branch. The copy is specified for each net by the command word. The nets referred to as 2 permit to transfer the input image without processing. The nets 4/5 permit to transfer the result image between two processings, simultaneously with data loading/result reading. The processing structure unit can be configured to adapt its processing in function of the acquisition mode and in function of the high-level application via software. The current acquisition data has to be stored into an internal memory to allow the preprocessing stage. Several types of preprocessing require a pixel neighborhood for each pixel process. A common way to operate is to use a video line to store few image rows. Unfortunately, such solution is not possible because the subimage size is not fixed. In the architectural solution presented here, an internal cache memory is associated at each processing. Consequently, the processing flow might not be synchronized with the output data flow of the memory MEM. Such solution enables to decrease the number of accesses to MEM. The size and features of the cache are defined to match the selected processing.

The processing modules are sharing the same input and output busses that are connected to the bidirectional main memory bus. So in order to store the results in the same memory, the input data enables to cascade the processing or to apply the same processing several times [19]. The tasks implemented in HW can be scheduled by the user by simple ordering of the tasks execution without any ordering constraint. To illustrate the intrinsic processing potential of the coprocessing, the performance of different implemented processings are reported in Section 6. A full application using SW/HW processing is presented in Section 5, to illustrate the capacity of the full system in terms of processing and data flow parallelism.

5. EXAMPLE OF ACHIEVABLE PERFORMANCE

The image processing presented in this section are commonly used by on-line industrial control systems. All these processing algorithms are quite *regular* and require a neighborhood of pixels and common operators (accumulation, multiplication, square, sorting, and logical). Therefore, they present a high potential parallelism that can be exploited during execution. Thus a hardware implementation is directly inserted in the coprocessor architecture. The image acquisition FPGA contains the camera driver, a PCI core and a rocket IO core. These FPGA resources are used for about 90%. No processing is implemented on this component. All the processings of the coprocessor are implemented in the other FPGA. Three different processing types have been implemented in the coprocessor:

   (i) a median filter on different basic kernels (1×3, 1×5, 3×3),

   (ii) a local adaptive binarization (Niblack algorithm) with a neighborhood of 8×8 or 16×16 pixels [20],
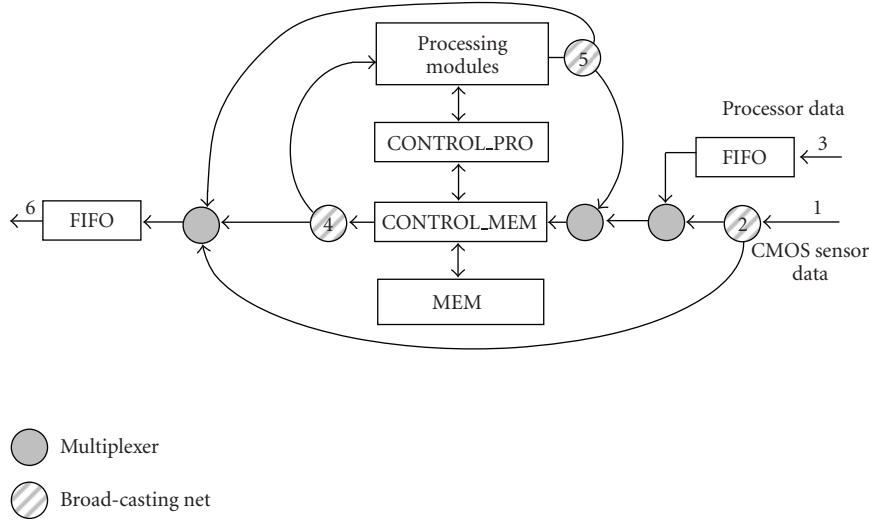
FIGURE 5: Diagram of the COP functional.

TABLE 1: Time processing of median filter 3×3.

	3×3	$1 \times 3, 1 \times 5$
Number of slices	313	265
Number of block RAM		9
Number of mult 16×16		0
Frequency (MHz)		100
Image size	Time processing (ms)	
512×512	5.22	2.61
256×256	1.30	0.65
128×128	0.32	0.16

(iii) a binary pattern recognition based on block matching with 32×32 and 64×64 pattern size.

All the implementation of the three processing includes a cache memory to provide data without latency to the designed architecture. The performances and the required hardware-resources obtained by the coprocessor architecture are reported in Table 1 for the median filter, Table 2 for the local adaptive binarization, and Tables 3 and 4 for the pattern recognition.

The median filter is a simple sliding-window spatial filter that replaces the center value in the window with the median of all the pixel values in the window. The median filter is normally used to reduce noise in an image. In the median filter implementation, two kinds of filter are implemented: a one dimensional (1×3, 1×5) and a two dimensional (3×3). The required resources for each kind are 265 and 313 slices for processing, the reunification use 893 slices which include all the processing tasks and the handling of all transfers and local buffers. Used in the preprocessing stage, the local adaptive binarization not only provides a very performing algorithm in presence of illumination or object variations, but also allows to reduce the bandwidth to the central CPU like an artificial retina sensor can perform. For example, a 1024×1024 full-range image requires 1 Mbytes

to be stored but the binarized image only 1 Mbits. If an area can be selected in the full-range image, for example a 256×256, the result image size would reduce to 64 Kbits. This process allows gaining a factor 64 on the original bandwidth. A threshold is processed for each pixel considering a fixed size neighborhood (8×8 or 16×16 can be selected). The following algorithm defines the local adaptive binarization:

$$\text{Threshold} = \text{Mean(NE)} - [0.1875 \times \text{Std(NE)}]$$
$$\text{If Std}^2(\text{NE}) < \text{STDREF then } B = 0 \qquad (1)$$
$$\text{Else if } P < \text{Threshold then } B = 1 \text{ else } B = 0,$$

where NE is the considered neighborhood, STDREF is the standard deviation of reference (determined experimentally), P is the pixel grey level and B is the binary value. Figure 6 describes the pipelined implementation of this algorithm (as mean, standard deviation, and variance). A cache memory (not represented in Figure 6) provides 4 pixels to the accelerator at each cycle. The signal Sel8o16 enables the size of neighborhood to be selected.

For the third processing, the binary shapes search, different tests have been made to compare with different image size and search sizes. The goal of this processing is to recognize one or more shapes in a binary image. The search window and the binary shape are both binary. The different shapes are searched in the defined window. The detection is based on the comparisons using the logical operator XNOR, which replaces the traditional matching criterion based on the sum of absolute difference (SAD). The binary correlation algorithm is defined as follows:

$$f(i, j) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} [s_1(x, y)\text{XNOR}s_2(x - i, y - j)] \qquad (2)$$

i and j between 0 and $N - M$. with S1 the binary shape ($M \times M$ size), S2 the binary search window ($N \times N$ size).
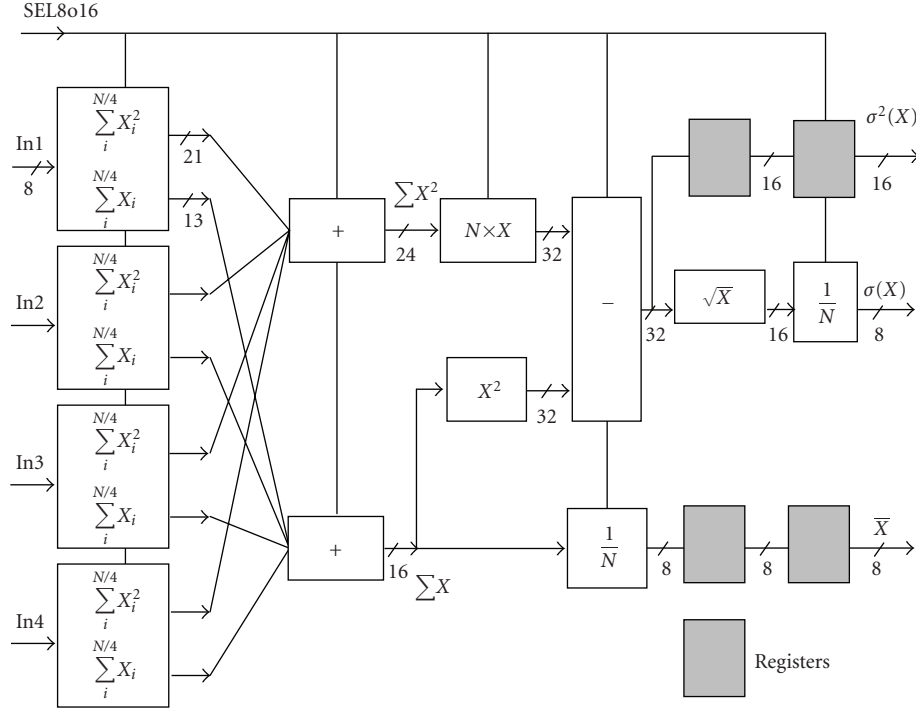
FIGURE 6: Block diagram of the coprocessor component implementing local adaptive binarization process.

TABLE 2: Local adaptative binarization.

		2	4	8
Number of points		2	4	8
Number of slices		477	965	1605
Number of mult 16×16		5	9	17
Frequency (MHz)		25	25	25
Size		Time processing (ms)		
Image	Block			
512×512	M16×16	158.09	79.04	39.52
	M8×8	40.80	20.40	10.20
256×512	M16×16	76.66	38.33	19.16
	M8×8	20.12	10.06	5.03
256×256	M16×16	37.75	18.59	9.87
	M8×8	9.92	4.96	2.48
128×128	M16×16	8.17	4.09	2.04
	M8×8	2.34	1.17	0.59
64×64	M16×16	1.54	0.77	0.38
	M8×8	0.52	0.26	0.13
Cycle time		40	20	10

This binary approach is commonly used in the optical character recognition field [21]. The binary shapes usually represent characters with different orientations. The entire processing is implemented and require 3021 slices and 9 blocks RAM of 18 kb each. In the table, only the required slices for a single processing are reported.

A comparison has been done between the performance obtained by the coprocessor architecture (COP) and a PC,

Bi-Xeon 1.7 GHz, 256 Mo Ram, Rambus 800 MHz (2 × 400 MHz). The performance results reported in Table 5 do not consider camera frame-grabber transfer time for the PC-based platform. The comparison shows that, besides the achieved speed-up factor up to a factor of 5 that would certainly result in higher considering the frame-grabber transfer time, the central CPU in the coprocessor approach is fully available for further processing. Moreover, when

TABLE 3: Binary shapes research with a 50 MHz frequency.

	64 × 64		32 × 32		16 × 16	
Shape size	64 × 64		32 × 32		16 × 16	
Block size	64 × 128		32 × 64		16 × 32	
Mem. blocks	6		3		2	
Slices/	2328	1300	1250	700	750	400
detect. block	127		70		40	
detection block used	16	8	16	8	16	8
Image Size	Processing time (ms)					
512 × 512	16.05	9.2	4.93	32.1	18.4	9.86
256 × 256	2.96	2.015	1.155	5.92	4.03	2.31
256 × 512	6.9	4.32	2.39	13.8	8.64	4.78
128 × 128	0.335	0.375	NC	0.67	0.75	NC

TABLE 4: Time to search a shape in an image (ms).

Number of shapes	5	4	3	2	1
256 × 512	68.52	59.88	51.24	42.6	33.96
256 × 256	32.67	28.64	24.61	20.58	16.55
128 × 128	6.73	5.99	5.24	4.49	3.74

TABLE 5: Processing comparisons.

Processing	PC (Mpixel/s)	COP (Mpixel/s)
Median 1 × 3	41	100
Median 1 × 5	28	100
Median 3 × 3	27	50
Niblack 8 × 8	5	25
Niblack 16 × 16	4	14

a bandwidth reduction is possible by means of adaptive acquisition, the coprocessor approach provides much higher speed-up gains.

6. APPLICATION EXAMPLE: READING A BAR CODE FOR POSTAL SORTING

6.1. Application description

The postal sorting is a real-world example showing the processing possibilities and the achieved level of parallelism of the system [22]. The goal of this application is to read bar codes on the letters, to enable the automatic sorting at the different stages of the logistic postal letter handling. If the bar codes cannot be read, the letter is rejected and need to be processed manually. This application has been developed with the objective of replacing an exiting platform which integrates a camera associated with a PC. The new embedded solution has been developed to increase as much as possible the processing performances and to obtain a portable and more flexible system. Indeed due to the fact that bar codes printed on letters may be of bad quality or superposed to other visual information the possibility of implementing more complex processing increase the rate of correct detections/decodings achievable. Ideally, to correctly read the largest percentage of bar codes, each processing stage

FIGURE 7: Example of an image which contains a bar code.

should require as much as possible processing power so as to guarantee that the bar code area is correctly localized (framed in Figure 7). In reality the processing resources are limited and, results are easier to extract and process a small part of the image that with a high probability includes the bar code, instead of dealing with the entire image of the letter which includes extra information that can potentially create errors for the code bar detection and decoding. In the postal sorting test application example, a letter is grabbed with the CMOS camera, and the speed of the transporter is around 4 meters per second. In the coprocessor platform, the processing stages used are

 (i) transposition,
 (ii) high pass filtering,
 (iii) dilatation plus subsampling,
 (iv) blobbing.

The final blobbing task is performed in the main processor. Details of these processing stages are provided in the following section. The final task is to read the bar code and send it to the postal sorting machine.

6.2. Details of the processing

So as to grab a letter, the CMOS camera is configured in the line scan mode since the high speed of the transporter (4 meters per second) would result into an image deformed as reported in Figure 8(a). The camera grabs the same line during a predefined number of lines or continuously and the acquisition FPGA rebuilds an image; this mode is shown in Figure 8(b). In this picture, the difference between the two modes is shown, and particularly the effect on the bar code. In area scan mode, the bar code would be completely unreadable. The first processing applied is a transposition. The transposition is used to rotate the rebuilt image to 90 degrees. A transposition is necessary because the other

(a) Area scan mode
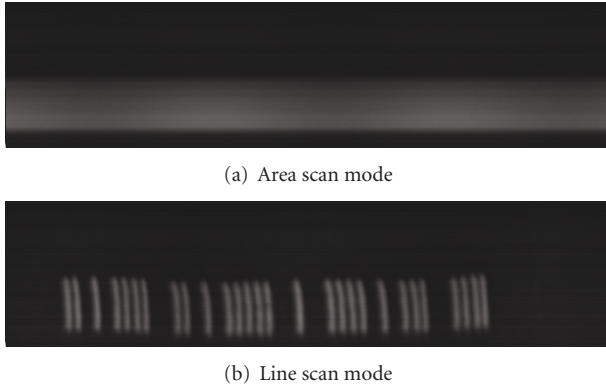

(b) Line scan mode

FIGURE 8: Visualization of the bar code with the two different acquisition modes of the camera.
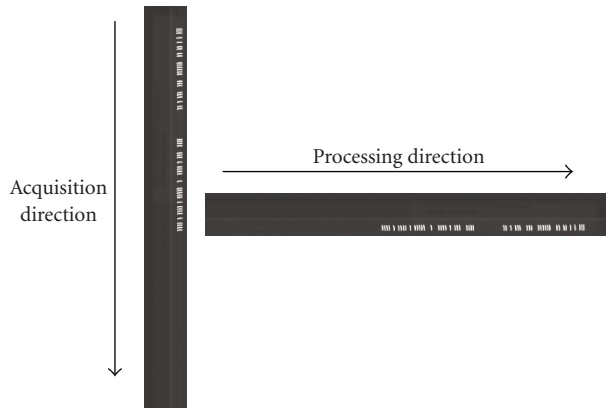

FIGURE 9: Transposition of an image to change in the appropriate processing sense.

processing stages are only compatible with a horizontal scanning (Figure 9). Four 32 bits words of four lines are stored in the memory, transposed, and stored again at a correct place in the SRAM memory. Transposition operation uses 4 registers to store temporally the result. The resources used by the FPGA are 186 slices (only about 2%). The first real image processing is a high pass filter. The high pass filter is used to delete the background and to raise the white bar code as shown in Figure 10(b) compared to the original image in Figure 10(a). As shown previously, the data bus is a 32 bits bus, and transfers 4 pixels at the same time. This task processes 4 pixels at the same time (i.e., 4 filters are active simultaneously). The high pass filter is a convolution between the image and the window which includes the coefficients. Results are directly stored in the SRAM. Obviously the coefficients of the filters are reprogrammable to be adapted to any other application. Resources used for this processing are 1125 slices (12%) and 44 (4×11) multipliers (50%).

The second processing stage is a dilation. The dilation is used to complete the region which integrates the bar codes, thus it will be easier to detect this region as shown in Figure 10(c). This latter processing is performed to replace

the central pixel by the maximum value of the 32 neighbor pixels.

The subsampling is the third processing. In fact, only one of 4 pixels is transferred, moreover one line over four (Figure 10(d)). The goal is to divide the size of the image by 16 and consequently the original pixel bandwidth. The dilation and the subsampling are executed in the same tasks to reduce memory access. As described in the two last processing, results are stored into the SRAM.

These three last processing stages are performed in one dimension (line dimension) to obtain the best result and to reduce the processing time with the access of the second dimension. The implemented processings (including the SRAM address controller) request all together 2623 slices (26%) and 44 multipliers (46%). Each processing is spilt in several dependent tasks in the pipelined architecture. Full application with all driver interface use 3880 slices (41%), 4 BRAMs (4%), and 44 multipliers (50%). 3 BRAMs are used like FIFOs to save the transfer of data in the interface, due to different clock domains. (*processing* ⇒ *PCI*, *RocketIO* ⇒ *processing* and *PCI* ⇒ *processing*). The fourth BRAM is used to store all the configuration sent via the PCI bus (configuration of each processing and of the application). Between each processing (transposition, high pass filter, and dilation + subsampling), a storage of results is made. Results after the high pass filter need to be stored and kept. In this image, only the areas will be sent after the coordinates calculation. As seen in the previous section, only the blobbing is made by the SW processor and all the other processing stages are performed by the FPGA (coprocessor). The blobbing is the last processing stage of the code bar detection. After the dilation, several white (or grey) areas are labeled. In Figure 10(d), two large areas are detected (the number of areas depends on the number of sections in which the bar code is partitioned), that correspond to the area including the bar codes, but other areas may be detected which are probably not part of the code. The goal is to determine the coordinates of the two zones that contain the code. The processor receives the result image of the dilation from the SRAM of the coprocessor and stores it into the DDR RAM associated with the processor. So as to detect a region (blob), the image is described row by row and when a pixel value exceeding the threshold is founded, the object is squared and associated with a label. Once the image is fully analyzed and labeled, the two largest areas, that are chosen probably including the bar code and the coordinates of these two objects, are extracted. Figure 11 shows a part of the blobbing image where white areas are detected (squared in grey). The transfer of these coordinates is made to the coprocessor and the coprocessor transfers only the selected regions to the processor. The regions are taken on the filtered image which is stored temporally in the SRAM (Figure 12). When bar codes are transferred to the processor, the decoding can be activated. To decode the bar code, an FFT is made following several tests to read correctly percentage rates approaching 100% of bar code detected. As shown in Figure 12, the decoded bar code value in the picture example is "111101011110101111100100111101011001111" and after all the processing the system correctly read the code.
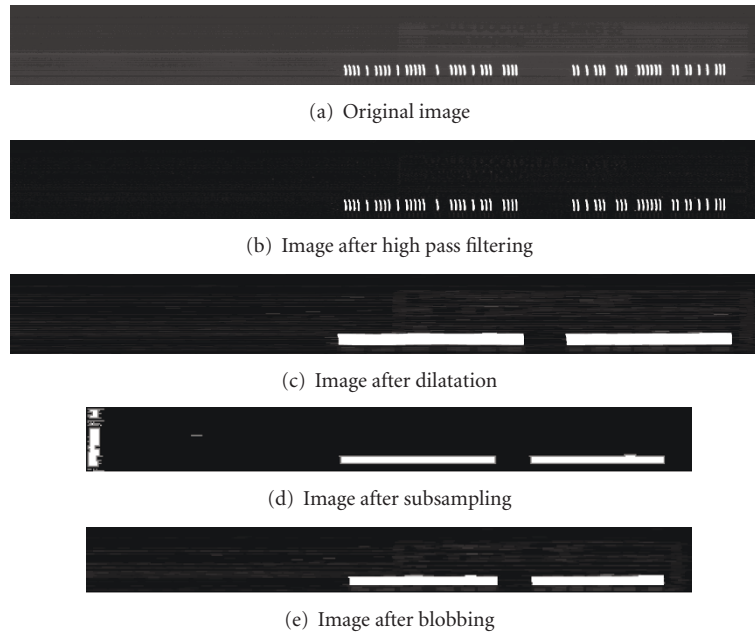
(a) Original image

(b) Image after high pass filtering

(c) Image after dilatation

(d) Image after subsampling

(e) Image after blobbing

FIGURE 10: Resulting images after each processing.

(a) Zoom of the subsampling image on one bar code area

(b) Zoom on the same bar code area after the blobbing
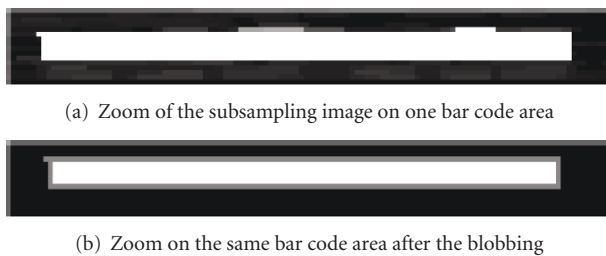
FIGURE 11: Principle of the blobbing processing.

FIGURE 12: Zone transferred after processing to read the bar code.

(a) Original image not readable at sight

(b) Image readable after processing
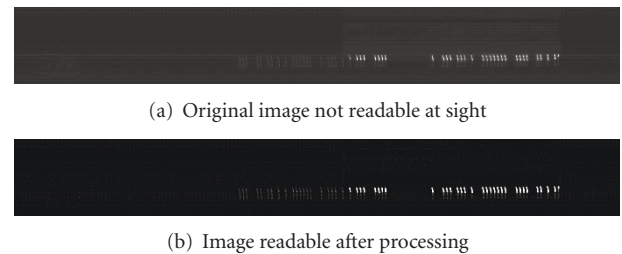
FIGURE 13: Preprocessing for improvement of "unreadable" bar code.

In Figure 13, the efficacy of the system is illustrated also with a bad bar code image which is not even readable at sight and without an appropriate processing. However, the system can correctly read it. Here, the bar code cannot be read exactly, but the system reads the correct bar code which is "11100110110101011111100101110101011001111". It proves the efficiency of the system.

So as to speed up processing tasks and decoding, the different stages can be performed in parallel and not sequentially. The application is divided in three main stages: acquisition (task 1), processing (task 2), and reading (task 3). Task 2 is the association of all the processing tasks executed by the COP and the blobbing including all the data transfer. These 3 tasks are executed in parallel to gain time and increase the number of letters processed. In Figure 14, a sequential sequence (task 1 following 2 and 3) is shown and during the acquisition the processor and the coprocessor do not work. The same remarks are valid when the coprocessor or the processor works simultaneously.

The specificity of the platform is that the 3 principal actions can work in parallel (i.e., tasks 1, 2, and 3 simultaneously). When an image is grabbed at time T, then

COP processes the image T-1 and the processor reads the bar code of the image T-2. Only the transfer between the FPGAs prevents an acquisition or a processing into the coprocessor. Moreover, the processor is implemented using a full DMA mode and can work continuously. It receives data and at the same time it decodes the bar code. By using this configuration, the processing time is the same from acquisition to the output, but the number of processed letters is increased. Results are shown in the following section, and a comparison between a sequential mode, a parallel mode, and the PC performances is provided.

6.3. Results and comparisons

In this section, the results of the processing are provided. A comparison between the classical PC-based system and the
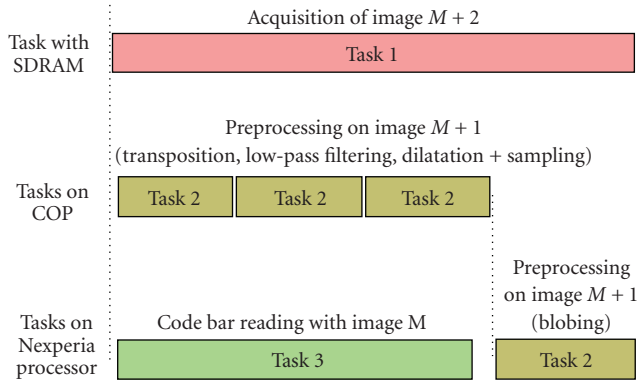
FIGURE 14: Scheduling of the 3 different tasks in parallel.

TABLE 6: Processing time.

Processing	Time (ms)
Acquisition	15.4
Transfer between the 2 FPGAs	4.6
Transposition	1.54
High pass filtering	1.54
Dilatation plus subsampling	1.54
Transfer in the processor of the subsampling image	0.15
Blobbing	4
Transfer in the processor of the bar code image	0.11
Reading the bar code image	12
Total	40.88

TABLE 7: Platform versus PC (approximative time).

	Sequential	Parallel	PC (ROI)
Time processing (ms)	40	40	40
Number of letter	15	30	15
Theoretical speed (m/s)	4	8	4

coprocessor platform is made. The PC platform is equipped with the camera (BCi4; Vector international/CCAM Technologies, available at http://www.vector-international.be/) associated with the compatible frame grabber. The coprocessor platform is obviously equipped with the same camera. The PC has a processor 3.2 GHz and 1 Go of RAM.

In Table 6, the necessary time for each processing needed to decode a bar code is shown. The tests were made with an image of 180 pixels width and 1712 rows captured. It is about a standard acquisition for a letter. Transfer is considered as a processing in the table. The transfer time from the coprocessor to the processor by the PCI is considerably reduced. The transfer of an entire image by the PCI takes 2.3 microseconds, but to transfer a subsampled image, it is 16 times lower (0.15 microsecond) and for the bar code it is 20 times lower (0.11 microsecond). This saving is a very important factor to speed up the overall processing performance.

(a) Coprocessor platform plus camera

(b) PC and coprocessor platform

FIGURE 15: Portability: coprocessor platform versus PC.

Table 7 presents the comparison between

(i) the coprocessor platform and a sequential reading,

(ii) the coprocessor platform and a parallel reading,

(iii) the current PC platform.

In the sequential and parallel mode, the processing time is approximately the same as a PC platform, but employing the parallelism, the number of the processed letters can be increased (i.e., number of images). In the case of the PC, the size of the processed image is reduced to a small ROI (around 512×70), against 1712×180 with the coprocessor platform. If the size is reduced to include correctly the bar code and not the image of the letter, the number of letters read can be increased up to 50. Moreover, the coprocessor platform is more efficient in hostile environment, small in size, and equivalent in terms of the percentage of bar codes correctly read. The portability of the two systems is illustrated in Figure 15. The size is reduced and results in being more appropriate for the integration in an industrial process.

7. CONCLUSION

Despite the increasing speed of PC processors and bus frequencies, the implementation of embedded coprocessor architectures expressly conceived for image sensors and inserted in the acquisition loop presents several advantages. Very high processing speed and reduced image data bandwidth are achievable. The architecture also provides

high degree of flexibility in the preprocessing stage for the different acquisition modes specific of CMOS imaging. Moreover, new processing tasks can be easily added in function of the application and the platform supports a wide panel of data interfaces. A complete test case application has been successfully implemented on this platform, with significant performance improvements when compared to a classical PC-based platform.

REFERENCES

[1] L. F. L. Y. Voon, G. Cathebras, B. Bellach, B. Lamalle, and P. Gorria, "Silicon retina for real-time pattern recognition," in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications II*, vol. 4306 of *Proceedings of SPIE*, pp. 168–177, San Jose, Calif, USA, January 2001.

[2] R. D. Burns, C. Thomas, P. Thomas, and R. Hornsey, "Pixel-parallel CMOS active pixel sensor for fast object location," in *Ultrahigh- and High-Speed Photography, Photonics, and Videography*, vol. 5210 of *Proceedings of SPIE*, pp. 84–94, San Diego, Calif, USA, August 2003.

[3] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10 000 fps CMOS sensor with massively parallel image processing," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, 2008.

[4] Y. Shi, R. Taib, and S. Lichman, "GestureCam: a smart camera for gesture recognition and gesture-controlled web navigation," in *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV '06)*, pp. 1–6, Singapore, December 2006.

[5] R. Y. D. Xu, "A computer vision based whiteboard capture system," in *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV '08)*, pp. 1–6, Copper Mountain, Colo, USA, January 2008.

[6] R. Mosqueron, J. Dubois, and M. Paindavoine, "High-speed smart camera with high resolution," *EURASIP Journal on Embedded Systems*, vol. 2007, Article ID 24163, 16 pages, 2007.

[7] M. McErlean, "An FPGA implementation of hierarchical motion estimation for embedded object tracking," in *Proceedings of the 6th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT '06)*, pp. 242–247, Vancouver, BC, Canada, August 2006.

[8] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-D convolvers for fast digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 3, pp. 299–308, 1999.

[9] C. W. Murphy and D. M. Harvey, "Reconfigurable hardware implementation of BinDCT," *Electronics Letters*, vol. 38, no. 18, pp. 1012–1013, 2002.

[10] N. W. Bergmann and Y. Y. Chung, "Video compression with custom computers," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, pp. 925–933, 1997.

[11] C. Hinkelbein, A. Kugel, R. Maenner, et al., "Pattern recognition algorithms on FPGAs and CPUs for the ATLAS LVL2 trigger," *IEEE Transactions on Nuclear Science*, vol. 47, no. 2, pp. 362–366, 2000.

[12] M. Gorgon and J. Pryzybylo, "FPGA based controller for heterogenous image processing system," in *Proceedings of Euromicro Symposium on Digital Systems Design (Euro-DSD '01)*, pp. 453–457, Warsaw, Poland, September 2001.

[13] Y. H. Jung, J. S. Kim, B. S. Hur, and M. G. Kang, "Design of real-time image enhancement preprocessor for CMOS image sensor," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 68–75, 2000.

[14] K. Tiri, D. Hwang, A. Hodjat, et al., "A side-channel leakage free coprocessor IC in 0.18 μm CMOS for embedded AES-based cryptographic and biometric processing," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 222–227, Anaheim, Calif, USA, June 2005.

[15] S. G. Smith, J. E. D. Hurwitz, M. J. Torrie, et al., "A single-chip CMOS 306 × 244-pixel NTSC video camera and a descendant coprocessor device," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, pp. 2104–2110, 1998.

[16] X.-R. Yu, Z.-B. Dai, and X.-H. Yang, "A parallel co-processor architecture for block cipher processing," in *Proceedings of the 7th International Conference on ASIC (ASICON '07)*, pp. 842–845, Guilin, China, October 2007.

[17] F. Paillet, D. Mercier, and T. M. Bernard, "Second generation programmable artificial retina," in *Proceedings of the 12th Annual IEEE International ASIC/SOC Conference*, pp. 304–309, Washington, DC, USA, September 1999.

[18] M. Bramberger, M. Quaritsch, T. Winkler, B. Rinner, and H. Schwabach, "Integrating multi-camera tracking into a dynamic task allocation system for smart cameras," in *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '05)*, pp. 474–479, Como, Italy, September 2005.

[19] J. Dubais and M. Mattavelli, "Embedded co-processor architecture for CMOS based image acquisition," in *Proceedings of IEEE International Conference on Image Processing (ICIP '03)*, vol. 2, pp. 591–594, Barcelona, Spain, September 2003.

[20] O. D. Trier and A. K. Jain, "Goal-directed evaluation of binarization methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1191–1201, 1995.

[21] S. V. Rice, F. R. Jenkins, and T. A. Nartker, "The fifth annual test of OCR accuracy," Tech. Rep. TR-96-01, Information Science Research Institute, Las Vegas, Nev, USA, 1996.

[22] R. Mosqueron, J. Dubois, and M. Mattavelli, "Smart camera with embedded co-processor: a postal sorting application," in *Optical and Digital Image Processing*, vol. 7000 of *Proceedings of SPIE*, pp. 1–12, Strasbourg, France, April 2008.