

Optimizing Resources of an FPGA-based Smart Camera Architecture

A. W. Azman^{1,2} A. Bigdeli² Y. M. Mustafah^{1,2} B. C. Lovell^{1,2}

¹ School of ITEE,
The University of Queensland, Brisbane Qld 4072, Australia.
{amelia,yasir,lovell}@itee.uq.edu.au

² National ICT Australia
Queensland Research Laboratory, Brisbane QLD 4000, Australia.
Abbas.Bigdeli@nicta.com.au

Abstract

The acceptance of reconfigurable platforms specifically FPGAs in embedded system design is becoming more apparent. While there are varieties of platforms available for smart camera implementation, our interest is mainly on reconfigurable platforms, specifically FPGAs. In this paper we discuss the research opportunity and the importance of hardware/software partitioning on FPGA-based smart camera platform. An overview of our current research work on the above mentioned problem is describe in this paper.

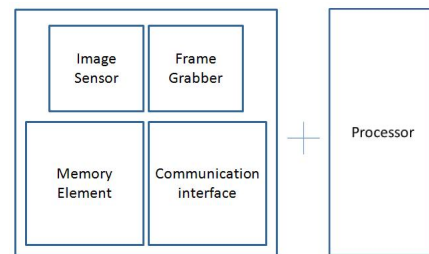


Figure 1. Basic representation of smart camera concept

1. Introduction

The role of a smart camera has developed rapidly over the years in both commercial as well as in academic research area. At one end, the smart camera benefited from the advances in computer vision coupled with the improvement in semiconductor technology. On the other, there has been increasing demand of smart camera especially in advanced video surveillances as well as in the industry automation. The general concept of a smart camera is to embed a processing element into or close to a standard camera architecture as illustrate in Figure 1.

In most cases particularly where smart cameras are used for surveillance purposes, the smart camera will employ certain degree of software components, for example, face detection algorithm. However, unlike a normal PC, the processor of a smart camera usually runs at significantly lower processing speed and has limited resources. While a PC can perform the face detection in fraction of a second, depending on the processing power, it might take much longer on a smart camera. This is because pure software implementation although offers high degree of flexibility, it

is usually slower when compared to hardware (gate-level) implementation. This could pose a problem since meeting certain time deadline is an important criterion of a smart camera. This paper will investigate the importance of hardware/software co-design for optimal solution for a Field Programmable Gate Array (FPGA)-based smart camera. Later in this paper, a proposed architecture for a smart camera prototype is presented. The prototype is customised for automated face detection for crowd surveillance.

The structure of this paper is as the following: section 2 will cover the related work on FPGA-based smart cameras. Hardware/software co-design for smart camera design is discussed in section 3. In section 4, we will review some of the partitioning techniques and modeling tools in hardware/software co-design. Section 5 presents the proposed smart camera architecture and our prototype architecture. Our future plans will be explained in Section 6 followed by concluding remarks in section 7.

2. Reconfigurable platform for a smart camera system

One of the earliest publications on smart cameras was by Wolf et al. [15] where they introduced a system that can build a complete model of the torso and recognizes various human gestures. The work started with research on a human activity recognition algorithm which soon evolved to hardware implementation. Their smart camera system is implemented by connecting a camera to a host PC. In this paper, we identified such architecture as PC-based smart camera. Another well-known smart camera project was by Bramberger et al [2]. They built a prototype camera called SmartCam which is a fully embedded smart camera system targeted for various surveillance applications such as traffic control. Kleihorst et al. [9] proposed a smart camera system that is able to detect and recognize faces. The system was implemented on a Phillips's INCA+ camera. Table 1 summarises a comparison of the three mentioned research groups on smart camera system design. It should be noted that they have all utilised DSP as their main processor for their smart camera design.

As reconfigurable platform emerged as an alternative to the conventional use of ASIC and general-purpose processor in embedded system design [16] [7], this has also affected research in the smart camera realm. A reconfigurable processor offers greater advantage in terms of flexibility and rapid conversion to mass market products. At present, the FPGAs dominate the sale of reconfigurable devices. In 2004, Chalimbaud and Berry [3] presented a smart camera with a template tracking capability. Their camera stabilizes and tracks Window-of-Interest (WOI) to reduce the data transfer between the image sensor and the processor. The camera is a pure software running on a Nios soft core processor on the Stratix FPGA. The overall architecture is as shown in Figure 2. In their smart camera design, the Nios acts as a master (labeled as M0) which control the different modules (for eg. P0-P5) of their smart camera.

Table 1. Leading research group on smart camera.

Research Group	Smart Camera Architecture	Application
Wolf [15]	PC-based, DSP Platform	Gesture Recognition
Bramberger [2]	Prototype, DSP Platform	Traffic Surveillance
Kleihorst [9]	INCA+ Camera, DSP Platform	Face Recognition

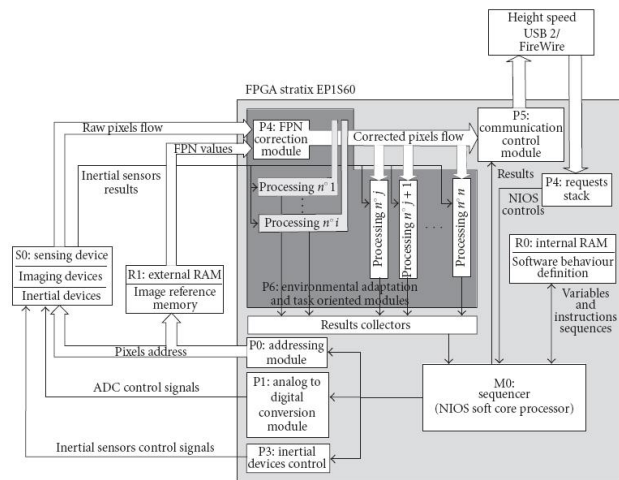


Figure 2. Block diagram of architecture adopted by Chalimbaud and Berry [3]

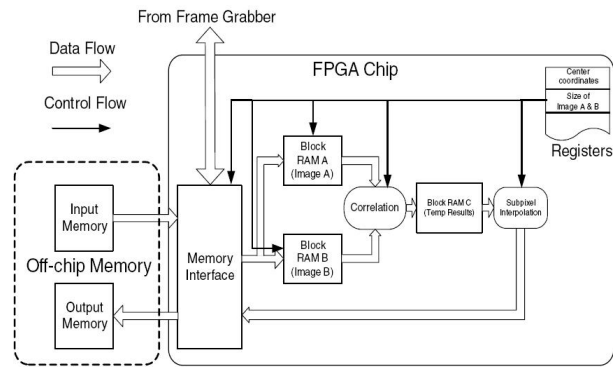


Figure 3. Block diagram of image processing architecture for Leeser et al. [11] smart camera

Leeser et al. [11] proposed a PC-based smart camera design for medical image processing. A video camera is connected to an Annapolis Microsystems Firebird board which has a Virtex 2000E FPGA and an external frame grabber. The FPGA processing board interfaces to a host PC via PCI bus. The camera FPGA processing board was designed to run in two parts concurrently. The first part reads pixels data and memory management while the second part runs application specific algorithm. Figure 3 illustrates the second part of the FPGA implemented by Leeser et al. In their case they have chosen two medical applications algorithm, Retina Vascular Tracing and Particle Image Velocimetry. In the same paper, they have explained that by implementing the cross-correlation

algorithm (since the cross-correlation computation are highly parallelisable) in hardware, the camera manages to improve data transfer rate by 20 times.

The two research groups described above are examples taken from the many FPGA-based smart camera research area. In most of the reported works on the FPGA-based smart camera, the researchers discussed their hardware architecture designs as well as elaborating their design methodologies. However, currently we are not aware of research work that focuses on hardware/software resource partitioning on reconfigurable smart camera platform. In the next section, we will discuss on development of hardware/software partitioning of embedded system design.

3. Hardware/software co-design in distributed embedded system

The smart camera system is an example of the growing trend in distributed embedded systems. A distributed system is a system that is built by cooperating heterogeneous elements [10]. For a smart camera system, the term 'elements' not only refers to the hardware and software components, it also can refer to the communication and network protocols. Figure 4 shows an example of the heterogeneous elements that can be combined to build a specific smart camera system.

As different elements have different characteristics and performance values, incorporating these components together can be an issue especially between the hardware and software components. While software implementation is cheaper and offers high degree of flexibility, hardware logic has better performance in terms of processing speed. Wolf in his paper [14] describes the idea of hardware/software co-design as to ensure that the design between the hardware and software component not only operate as required, but also meets **performance, cost and reliability goals**. In other words, hardware/software

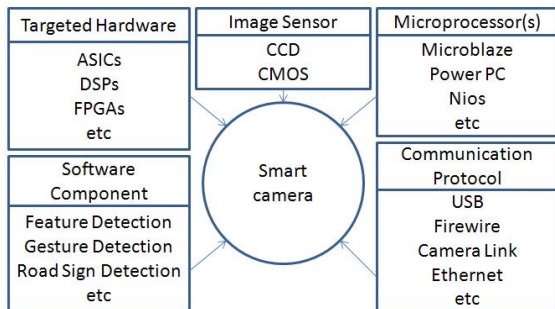


Figure 4. Example of heterogeneous elements of smart camera architecture

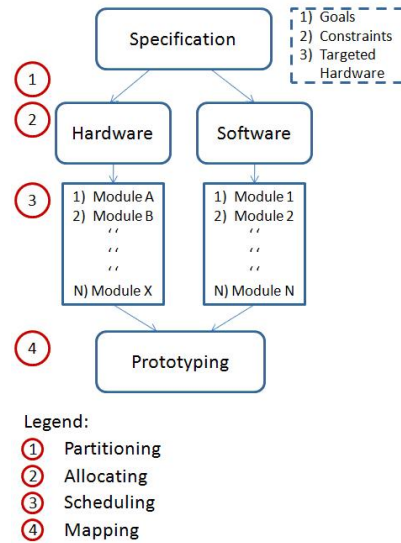


Figure 5. Example of heterogeneous elements of smart camera architecture.

co-design is a method to provide an optimum trade-off solution between hardware and software implementation while meeting system specification.

The major problem in hardware/software co-design is identifying the balance between the hardware and software component i.e. finding the optimal partitioning solution. Poor partitioning solution will result in excessive usage of resources and low performances. Another problem in hardware/software co-design is developing an efficient communication link between the partitioned hardware and software solution.

A general picture of a complete hardware/software designing process is illustrated in Figure 5. The first step of hardware/software **partitioning** is to consider the overall system requirements. These include the system goals and constraints (timing, power consumption and etc.) as well as selected target device. The early partitioning process will result in the system to be modeled in smaller functional modules. There are two widely used partitioning strategies [12]: 1) the heuristic approach and 2) decomposition strategy. Later, in the **allocating** process, the functional units are allocated into either hardware (gate-level) or software (microprocessor) implementation. Then there are the **scheduling** and **mapping** process. Scheduling is the process of assigning order to each functional modules based on data dependencies, timing constraints and etc while mapping maps the final chosen modules into microprocessor or logic implementations.

4. Partitioning techniques and modeling tools

The key feature that made FPGA an excellent hardware/software target platform is due to the fact that modern FPGAs allow embedded microprocessor architecture to be implemented on the same chip [13] [12]. The internal structure of an FPGA is the other positive aspect that indirectly affects hardware/software solution on FPGA.

The three common parallelism that are exploited are data-level parallelism, instruction-level parallelism and task-level parallelism. However, there is yet a rule of thumb to effectively partition an application on FPGA. For an FPGA, there are two partitioning domains that demand attentions [16]: the **spatial** and the **temporal** domains. Spatial partitioning is referring to the physical implementation of different functionality at different location of the hardware resources. On the other hand, temporal partitioning allows the FPGA to be reconfigured at different phases. In other word, reconfiguration at run-time.

As discussed in [13], hardware/software partitioning could improve the application performance of an FPGA while reducing its power consumption. Due to that, many researchers have been devising ways for optimal hardware/software partitioning approach on reconfigurable platforms. Among them are Dave et al. [4] who developed a heuristic-based algorithm called COSYN. The COSYN algorithm was developed based on task-level parallelism. Another reserch group that also exploit task-level is Dick and Jha [5]. They developed a system that could dynamically reconfigure an FPGA called CORDS. Yanbing et al. [16] on the other hand proposed a hardware/software partitioning algorithm that exploits the instruction-level parallelism aspect.

While the above mentioned groups have focused on reconfigurable platforms, there have been many important works on hardware/software partitioning that features different architectures. For example, Ernst et al. [6] proposed Cosyma, an automatic software-oriented approach for hardware/software partitioning on microprocessor. The POLIS approach was proposed by Balarin et al. in [1] also targeted microprocessor architecture. Gupta and Micheli [8] presented a partitioning template for digital systems. Several other important partitioning techniques are elaborated in [7] [14]. From our studies, we have noticed that the heuristic partitioning approach (iterative method, constructive method, simulated annealing and etc.) has been applied to some extent in every partitioning design.

Besides the increase of interest in the partitioning research area, there has also been the same pattern in hardware/software modeling tools growth. Traditionally,

the standard way of describing hardware is by using VHDL or Verilog language. However, the two languages are not suitable for embedded system design that involve both hardware and software components. Because of that, developers are designing alternative hardware description language tools that are C or C++ language based to ease modeling hardware/software co-design. Currently, there are Celoxica Handel-C and Impulse-C which are C language based while SystemC which utilised C++ language. Besides that, there are UML, SysML, Simulink by MathWorks and Ptolemy which are graphical modeling tools for hardware/software co-design.

Since a smart camera architecture has constraints in both processing speed and resources, applying an efficient hardware/software partitioning could help to increase the performance of a smart camera. Besides that, since hardware/software partitioning reduces the overall system power consumption, it would provide an optimal solution to power management in smart camera design.

5. Overview of NICTA smart camera architecture

In this section we will elaborate the specification of our smart camera system. This section will cover our research goals, proposed smart camera design, constraints in our design as well as the chosen hardware for our smart camera prototype.

5.1. Research goals

Our aim in this project is to design and build a high speed and high resolution FPGA-based smart camera. Having that said, the main challenge is to build a standalone, high performance with low cost and low power consumption smart camera system that integrates a face detection algorithm for crowd surveillance as a case study on a low resource Spartan FPGA-based series.

5.2. Proposed smart camera architecture

By examining previous work, we realized that most image resolution on existing smart camera ranges from 192x124 pixels to 640x480 pixels (VGA standard). In our smart camera design, we have decided to extend the idea of smart camera specifically for face detection in crowd surveillance. Crowd surveillance usually surveys a wide area with many of objects of interest in view. Thus, we believed that having high resolution image as a crucial point in order to provide detailed information regarding objects in view. Figure 6 shows an example of how higher resolution image could affect face detection result.

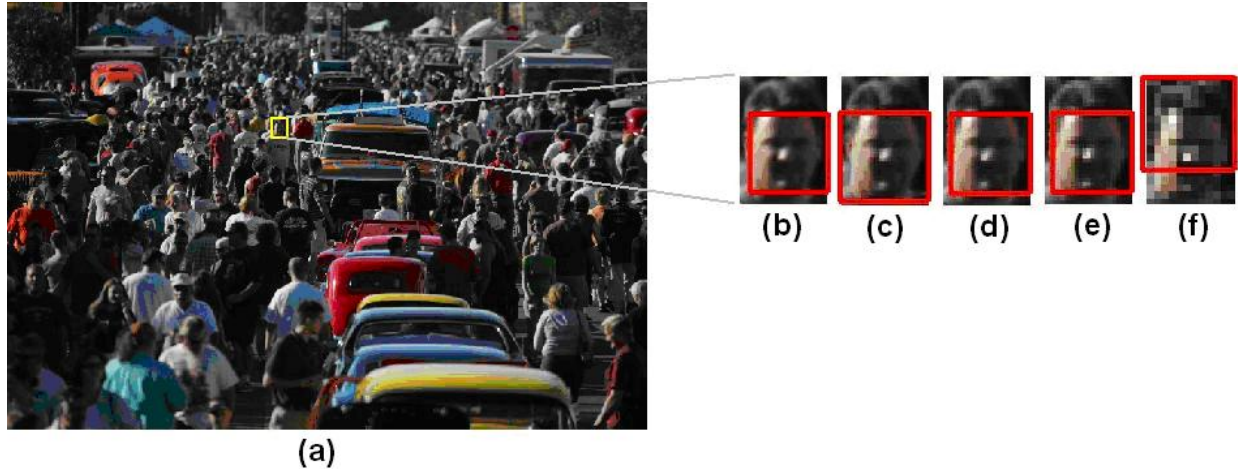


Figure 6. Overall scene (a) ROI extracted from scene with resolution of 7Mp(b), 5Mp(c), 3Mp(d), 1Mp(e) and VGA(f).

In the example, the region of interest (ROI) is extracted from an image of a crowd of people (a). The face (b) extracted from a 7 MP (MegaPixel) high resolution image is much more recognizable than (f) extracted from the lower resolution (VGA) image. The extracted faces were also tested for suitability for automatic detection using a Viola-Jones face detection module. The images (c), (d) and (e) taken with 5, 3, and 1MP sensors were suitable for face detection. However, face cannot be correctly detected in the VGA image (f) because the image does not have enough details for the face detection module to work correctly.

5.3. System design constraints

There are several constraints that we have taken into consideration in designing our smart camera the main ones being:

1. Real-time constraint: Meeting real-time constraint is an important issue since our smart camera targeted application in the surveillance area with real-time response requirement.
2. Hardware resources: In our smart camera design, we have chosen the Spartan FPGA-based series. A Spartan FPGA is a low cost version of the high performance Virtex family. Hence, the Spartan series has reduced hardware resources.
3. Bandwidth constraint: Higher resolution image would require higher data transfer rate. In our current design, our smart camera has a limited communication bandwidth to a host PC of 800Mbps.

4. Memory capacity: The memory storage is highly dependable on the image sensor. For example, a 5Mp image sensor has a total raw pixels value of 5Mp times Bit depth.

5.4. Hardware specification of NICTA smart camera prototype

As detailed in section 5.2, we have decided to use a high resolution image sensor in our smart camera design. Figure 7 shows an overview of our proposed smart camera architecture design. To meet our design requirement, we have chosen a 5 Megapixel (2592x1944 pixels) CMOS image sensor headboard manufactured by Micron. This sensor headboard could operate up to 14fps (frame-per-second) at full resolution. The main reasons to choose the CMOS image sensor are because unlike CCD image sensor, CMOS image sensor has parallel data access for faster data manipulation, low power consumption and the on-chip functionality. The main board of our smart camera has a Spartan-3 series FPGA chip (XS3C5000). For this project, Spartan FPGA is chosen as the processing target device primarily because of its low cost and low power consumption. The main board also has a DDR SDRAM slot. To ensure that our smart camera system could handle the high data rates (from the high resolution image sensor), we have installed a 1GB DDR SDRAM as the main frame buffer of the camera. As for the camera communication interface to the host PC, we have decided to use a FireWire 800 (1394b) communication protocol. A FireWire board that consists of Texas Instrument's 1394b Link Layer and Physical Layer controller chips and 3 FireWire 800 ports is used in our design. All three boards were interfaced

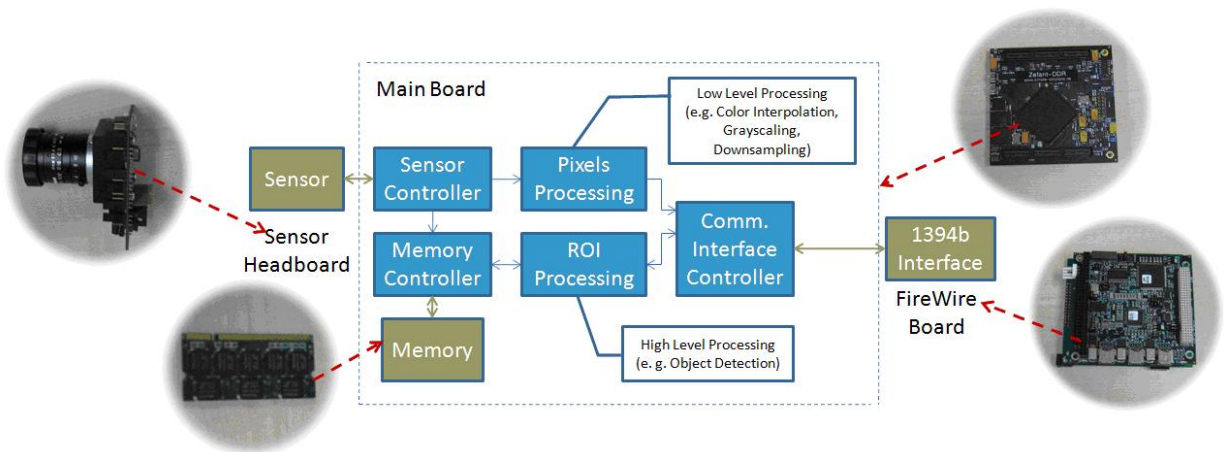


Figure 7. Proposed smart camera system architecture (photos are not to scale).

together and powered using a custom-designed PCB board (interface board). Figure 8 shows a picture of our smart camera prototype while Table 2 summarises the basic specification of our prototype.

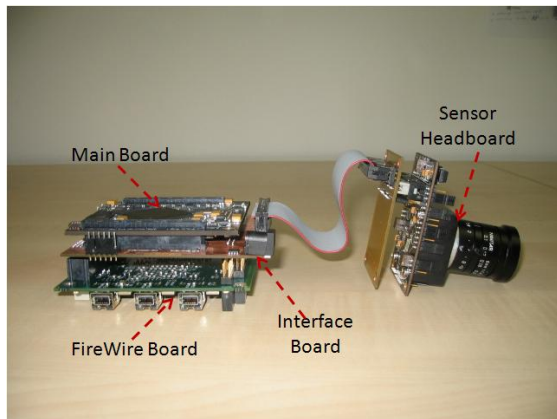


Figure 8. NICTA smart camera prototype

Table 2. NICTA smart camera specification

Parameter	Value)
Sensor Type	CMOS
Resolution	2592 x 1944
Processing Element	Spartan-3 FPGA
Comm. Interface	FireWire800
Physical Dimension	90 x 90 x 150 mm ³

6. Future plan

In the near future, we will explore the possibility of methodically hardware/software partitioning of the smart camera. Image processing on high-resolution image requires higher processing load which makes software implementation alone inadequate for our smart camera to achieve real-time performance. By having a hardware/software partitioned processor, an effective face detection and data-transfer can be achieved on our proposed smart camera. Our target is to develop a generic hardware/software partitioning template specifically for FPGA-based smart camera for different software applications. Unlike other embedded system, a smart camera system captured images data in a continuous manner. However, since it is impossible to buffer all the images, a smart camera system would require a good memory management to reduce the expensive back-to-back memory access. Hence, our hardware/software partitioning will also be focusing on the memory management aspect of a smart camera.

7. Conclusions

Lately, the acceptance of reconfigurable platforms specifically FPGA in embedded system design is becoming more apparent. With the advancement in FPGA technology, incorporating microprocessor on a same FPGA chip is made possible. This has opened the door to efficient hardware-software co-design implementation. The smart camera realm is one of the research areas that is also affected by this changes. In this paper, we have presented the idea of hardware-software co-design for a distributed FPGA-based smart camera system. We also presented

our idea on improving the existing smart camera design especially cameras that are used for surveillance purposes.

8. Acknowledgements

This project is supported by a grant from the Australian Government Department of the Prime Minister and Cabinet and by the Australian Research Council through the Research Network for Securing Australia. NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

- [1] F. Balarin, M. Chiodo, P. Giuto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincetelli, E. Sentovich, K. Suzuki, and B. Tabbara. *Hardware-Software Co-Design of Embedded Systems: The POLIS approach*. Kluwer Academic Publisher, United States, 1997.
- [2] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2):68–75, 2006.
- [3] P. Chalimbaud and F. Berry. Design of an imaging system based on fpga technology and cmos imager. In *IEEE International Conference on Field-Programmable Technology*, pages 407–411, 2004.
- [4] B. Dave, G. Lakshminarayana, and N. Jha. Cosyn: Hardware-software co-synthesis of embedded systems. pages 703–708, 1997.
- [5] R. Dick and N. Jha. Cords: Hardware-software co-synthesis of reconfigurable real-time distributed embedded systems. In *IEEE International Conference on Computer-Aided Design*, pages 62–68, 1998.
- [6] R. Ernest, J. Henkel, and T. Benner. Hardware-software cosynthesis for microcontrollers. *IEEE Design and Test of Computers*, 10(4):64–75, 1993.
- [7] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu. An overview of reconfigurable hardware in embedded systems. *EURASIP Journal on Embedded Systems*, 2006, 2006.
- [8] R. Gupta and G. D. Micheli. Hardware-software cosynthesis for digital systems. *IEEE Design and Test of Computers*, 10(4):29–41, 1993.
- [9] R. Kleihorst, M. Reuvers, B. Krose, and H. Broers. A smart camera for face recognition. In *International Conference on Image Processing 2004*, pages 2849–2852, 2004.
- [10] K. Kuchcinski. Synthesis of distributed embedded systems. *25th Euromicro Conference*, 01:1022–1030, 1999.
- [11] M. Leeser, S. Miller, and Y. Haiqian. Smart camera based on reconfigurable hardware enables diverse real-time applications. In *Field-Programmable Custom Computing Machines*, pages 147–155, 2004.
- [12] G. D. Michell and R. Gupta. Hardware/software co-design. In *Proceeding of the IEEE*, pages 349–365, 1997.
- [13] G. Stitt, R. Lysecky, and F. Vahid. Dynamic hardware/software partitioning: A first approach. *DAC*, 00:250–255, 2003.
- [14] W. Wolf. Hardware-software co-design of embedded systems. In *Proceeding of the IEEE*, pages 967–989, 1994.
- [15] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *Computer*, 35(9):48–53, 2002.
- [16] L. Yanbing, T. Callahan, E. Darnell, R. Harr, U. Kurkue, and J. Stockwood. Hardware-software co-design of embedded reconfigurable architectures. In *37th Conference on Design Automation*, pages 507–512, 2000.