The Architecture of an Embedded Smart Camera for Intelligent Inspection and Surveillance

Michał Fularz, Marek Kraft, Adam Schmidt, and Andrzej Kasiński

Institute of Control and Information Engineering, Poznań University of Technology, Piotrowo 3A, 60-965 Poznań, Poland
michal.fularz@put.poznan.pl

Abstract. Real time video surveillance and inspection is complex task, requiring processing large amount of image data. Performing this task in each node of a multi-camera system requires high performance and power efficient architecture of the smart camera. Such solution, based on a Xilinx Zynq heterogeneous FPGA (Field Programmable Logic Array) is presented in this paper. The proposed architecture is a general foundation, which allows easy and flexible prototyping and implementation of a range of image and video processing algorithms. Two example algorithm implementations using the described architecture are presented for illustration – moving object detection and feature points detection, description and matching.

Keywords: Smart Camera, Embedded System, Computer Vision, Hardware Software Codesign, FPGA.

1 Introduction

Over the last years, an increased demand for vision-based inspection and monitoring can be observed. The most important reason behind this change is the fact, that visual data carries a lot of useful information. However, it also makes processing of visual data a non-trivial task. Extraction of desired information from such raw data requires significant computational power. At the same time, the number of cameras in a typical multi-camera system increases, to the point, at which human operators are unable to handle all the incoming information. A typical approach to the automated processing of data coming from a multi-camera system is the use of central server. However, such an approach is not without its limitations. The transfer of image data from the cameras to the central processing node requires appropriate infrastructure, which is oftentimes costly and may be cumbersome to deploy. Under extreme conditions, such as a very large number of cameras and/or the use of high resolution or high framerate video data, the bandwidth or processing power requirements may be impossible to satisfy. The most common way to deal with these limitations is to use more and more sophisticated compression algorithms. On the downside, the use of

R. Szewczyk et al. (eds.), *Progress in Automation, Robotics and Measuring Techniques*, Advances in Intelligent Systems and Computing 350, DOI: 10.1007/978-3-319-15796-2_5

such algorithms increases the computational power demand for video encoding and decoding, resulting in increased application cost and power consumption [15]. Furthermore, the compression degrades image quality which may impact the quality of processing results [18].

In the face of the problems and limitations given above, distributed smart camera networks are gaining more and more attention. The idea corresponds with the current Internet of Things trend[3]. In such networks, most of the processing is performed in the measurement node (a smart camera) with a certain degree of autonomy [5][2]. As only the information that is useful from the point of view of the application is transmitted to the central server or other cameras in the network, the load on communication infrastructure is reduced.

The autonomy of the smart camera allows to adjust the activity rate of the sensor to current conditions, which in turn reduces the power consumption. Furthermore, it also allows the collaborative, network-wide execution of designated tasks.

In this article, we present a smart camera architecture based on heterogeneous system-on-chip (SoC) Xilinx Zynq platform [27]. The device integrates a dual-core microprocessor, a pool of programmable logic resources and other peripherals in a single chip, enabling the implementation of highly integrated, low power, high performance embedded systems.

2 An Overview of the Existing Solutions

Depending on the perceived role of the smart camera in the network, existing solutions can be divided into three distinctive categories:

2.1 Semi-disposable, Widely Deployed Sensor Nodes

The main focus of the design of the smart cameras in this category is their low cost and low power consumption. As they are intended for possibly long battery operation, they are equipped with low power, yet relatively slow microprocessors or microcontrollers and rather slow communication interfaces. The processing power allows to perform simple operations on low resolution images (CIF, QCIF, QVGA) with the speed of a few frames per second or less. The interface is in most cases based on a low power, low bandwidth solution, e.g. ZigBee [14] or 6LoWPAN. The use of such an interface improves battery life, but on the other hand it is too slow for the transmission of images at high framerates. Moreover, the computational platform is usually not fast enough to compress the acquired images or video stream. The data transmitted in such sensor networks is usually constrained to observed scene metadata [22].

2.2 Application Processor and DSP-Based Smart Cameras

As the low power consumption is not always the primary concern and more sophisticated image and video processing requires more computational power,

smart camera designs are usually powered with application processors [9][10], digital signal processors (DSPs) [11][16] or a combinations of both for enhanced flexibility [25]. Such solutions are capable of performing video analysis in real-time. In many cases, they are also capable of on-the-fly video coding, so beside the scene metadata they can transmit the live video stream for viewing, logging and archiving. Video streaming requires a relatively high speed communication interface, contributing to the overall power consumption of the system [24]. The most common approach is using existing Ethernet or WiFi infrastructure.

2.3 Smart Cameras Using FPGAs

As shown in [4], computer vision algorithm implementations based on FPGAs can reach the performance beyond the capabilities of modern microprocessors. This is especially true for image processing operations based on pixel-level or neighbourhood-level access, as the FPGAs are inherently capable of massively parallel processing. Parallelism can also be achieved using GPUs, but such approach is usually not feasible for smart cameras because of the power constraints. Dedicated, application-specific integrated circuits like the one presented in [1] are another alternative. However, as the demands on computational power increase, the cost of development and implementation of specialised integrated circuits using state of the art fabrication process node becomes increasingly prohibitive. FPGAs are easily reconfigurable and offer the capability of connection with multiple external devices, e.g. CMOS imagers, other sensors, communication interfaces. However, dedicated programmable logic coprocessors are not well suited for high-level processing tasks that exhibit data-dependancy and irregular control flow. Thus, programmable processors are the prime choice for these tasks. On the other hand, the amount of logic resources in modern FPGAs is enough to implement a multicore microprocessor system along with specialised image processing hardware. In the light of the above, FPGAs are often viewed as an interesting computational platform for smart camera implementation, either standalone [7][6], or as a part of a system [19][23].

3 Architecture of the Proposed System

The opposing requirements of high computational capabilities and low power usage make choosing the platform for smart camera a complicated task. In our previous work [17], [13] we implemented image processing algorithms using FPGA devices (Xilinx Spartan 6 and Virtex 6) as the computational platform. These solutions were leveraging the reprogrammable logic for accelerating the image processing task while using soft processor cores (MicroBlaze) for communication and control functions. The algorithms that were difficult to implement in hardware had to be partitioned between many soft processors cores. The perfomed tasks required the implementation of a few of such processors due to their relatively low computational performance. This in turn resulted in the increased use of FPGA resources.

The advent of Xilinx Zynq devices, which combine high-performance dual ARM Cortex A9 microprocessor cores and a large pool of programmable logic resources, opens the way to the implementation of mixed, heterogeneous, integrated architectures, in which hardware accelerators can work alongside the general purpose processors. This provides a significant advantage over the solutions presented in section 2.3, as such architecture combines the best of microprocessor and programmable logic worlds in a single chip and provides mechanisms for high-throughput communication between these two domains. On the other hand, taking advantage of both parts of the device and dividing the workload to achieve the optimal performance is not an easy task.

The presented solution is a general framework for Xilinx Zynq devices, allowing seamless cooperation of the hardware coprocessors and the high level software-based algorithms (e.g. algorithms from the OpenCV library) running on the Cortex A9 cores. The architecture provides means for transporting the data between the processing elements in the system and the external memory. In addition, it offers an easy way of setting the parameters of hardware coprocessors. The schematic of proposed architecture is presented in Fig. 1.

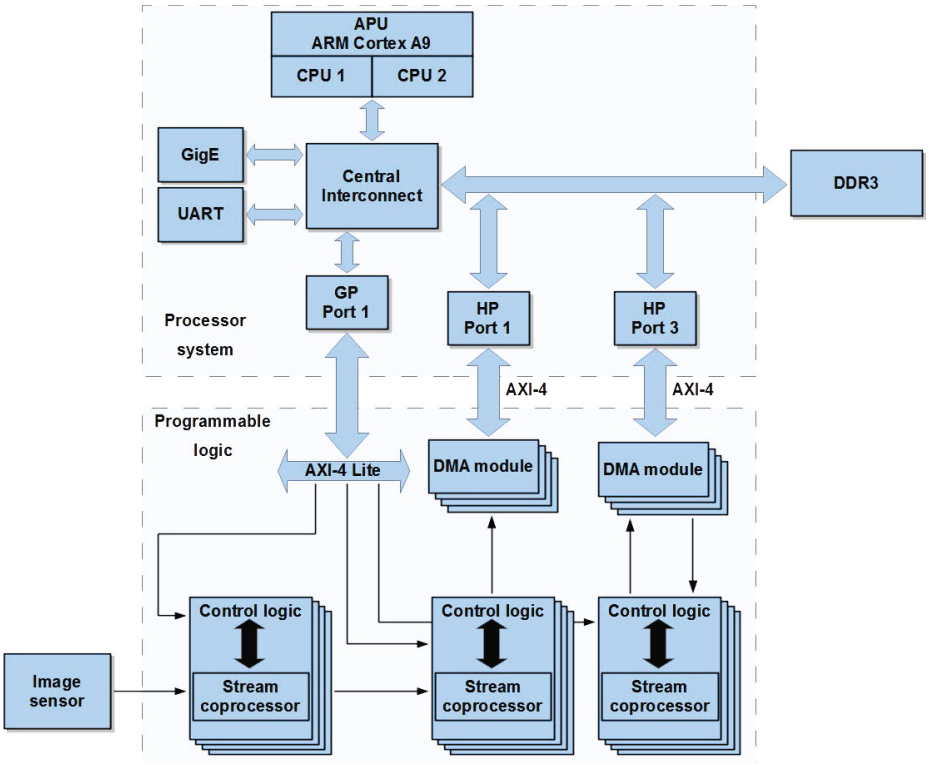

Fig. 1. The block diagram of the proposed system architecture

The architecture is based on the assumption that most of the computationally intensive operations should be performed by specialised coprocessors implemented in programmable logic. Such an approach is suitable for achieving high performance and low power as long as the algorithm being implemented is receptive to parallelization and operates on continuous stream of data instead of requiring random access to memory. Exemplary system presented in Fig. 1 acquires the input data directly from the image sensor or from a dedicated hardware block. Using such dedicated functional blocks, the data can be provided by e.g. GigE Vision IP camera connected to Ethernet port in the processor subsystem or read from the external memory. In each case, the image has to be converted to a stream of pixels encapsulated in AXI4-Stream interface [26]. Data in this format is provided to one or multiple processing elements (PE). Each PE consist of some control logic and stream coprocessor. The detailed structure of PE is shown in Fig. 2.

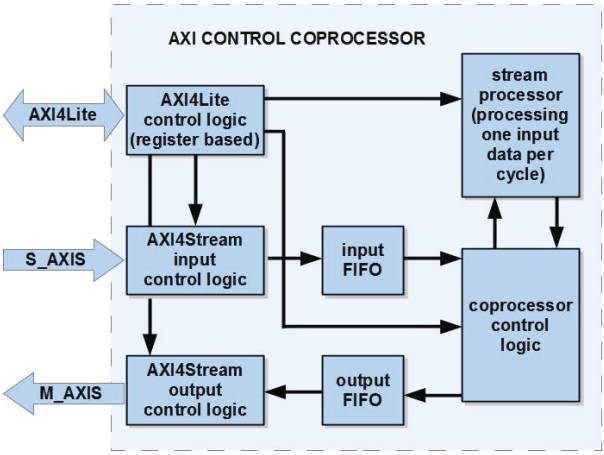

Fig. 2. The block diagram of the coprocessor control IP-core

The incoming data provided in AXI4-Stream format is buffered by input FIFO. The stream processor operates only when the input data (stored in input FIFO) is available and the output FIFO is not full. This design makes it possible to use different clock domains for the communication and the processing part, which is handy when e.g. the data is provided as 256-bit wide vectors while the stream processor input accepts 8-bit data. The provided coprocessor control logic block is very flexible, which makes it easy to implement various image processing algorithms without any concerns about communication and data transfers. In addition a dedicated AXI4-Lite interface is used for configuring parameters of stream processor (e.g. filter coefficients or threshold values).

As shown in Fig. 1, the PEs can be linked to each other provided the output of the preceding one is compatible with the input of the following PE. This can be

used for linking subsequent filtering or morphological operations or connecting the detector of feature points to the descriptor without storing the intermediate results in external memory. In each case, the final outcome of processing are transferred to external memory using DMA (Direct Memory Access) modules. Using the DMA modules is beneficial for the overall system performance, as it assures, that all the memory transfers from and to the PEs are handled by the memory controller exclusively. The processing results stored in in the memory can be fetched by other PE or by the software running on ARM cores. In such a solution, the processor cores act as a class of PE and can be used to implement algorithms that are otherwise difficult to implement in programmable logic. It should be noted, that unlike the PEs implemented with programmable logic, the ARM cores are limited to two instances and cannot be replicated. Additionally, in case of the smart camera, the processor is used for the communication with external world usnig the Ethernet or WiFi interface and for the control of the PEs.

The presented architecture of a smart camera can be applied in various use cases, such as automated surveillance or product inspection. The process of adapting the framework to the task should start with pure software implementation, followed by the transfer of the most computationally intensive parts to the hardware. Algorithms that perform pixel-level or neighbourhood-level operations are the best candidates for that, and can be placed in the system just after sensor acquisition module. Provided coprocessor controller simplifies the hardware implementation and hides the communication details from the user. The high level image processing algorithms (e.g. object recognition) can be converted to dedicated hardware coprocessors or, if it is not a viable solution, run on the processor cores. The standard ARM processor implemented in Zynq devices can allow the use of Linux operating system alongside popular image processing libraries (e.g. OpenCV). Modular architecture of the presented solution can be tailored to the application by replicating the PEs for higher degree of parallelism.

4 Example Results

To prove its feasibility and usefulness, the presented smart camera architecture was tested using two different vision applications. The first one was moving object detection and labelling [12], while the other one was point feature detection, description and matching. Schematics for both of this systems are shown in Fig. 3. In case of the solutions, only the coprocessor part was presented, as the processor subsystem was identical.

The moving object detection system uses approximate median algorithm presented in [20] with additional filtering and morphological operations implemented in hardware. The labelling operation is performed by ARM cores, which allows for parallel operation. The system achieves over 300 frames per second for VGA images and over 20 for frames per second for 1920x1080 images. The detailed information about processing speed and comparison to PC is presented in table 1, while resources usage is shown in table 2.

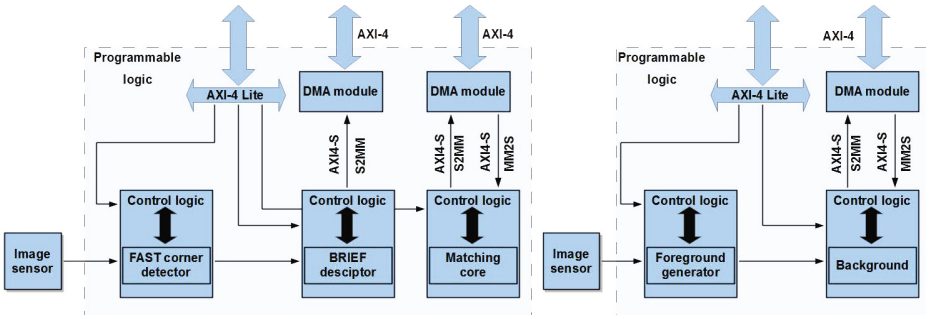

Fig. 3. On the left - block diagram of feature point detection, description and matching system. On the right - block diagram of moving object detection system.

The feature detection, description and matching system performs each of its primary operations in hardware. The image registered by the sensor is passed through AXI4-Stream interface to the first processing blocks - the FAST detector [21] and then to BRIEF descriptor [8]. For each incoming pixel, the coprocessors check if it is an interest point and calculate its descriptor. The descriptor is only saved if the pixel is considered to be a feature point. The results provided by this IP cores are sent to the external DDR RAM memory using the DMA engine. After two consecutive images are processed, the processor requests the matching module to find best matches between vectors of descriptors from both images. The appropriate data (descriptors) is send from DDR RAM memory to dedicated hardware matcher block and as a result, the best match for each interest point is found. Further processing can be done using general purpose ARM cores. The system can achieve over 300 VGA frames per second, while the matcher with four matching cores can keep up with detection and description of up to approximately 1100 feature points found in each matched image. Resources usage by this system is shown in table 2.

Table 1. Processing times for different resolutions and processing platforms given in milliseconds

Resolution		640x480		1920x1080	
Platform		Zynq	PC	Zynq	PC
image processing	software	42,45	2,80	285,53	18,15
	hardware	3,10	N/A	20,94	N/A
labeling	software	3,27	1,05	21,86	7,10

Table 2. Resource usage of the implemented design (designations: FFs - flip-flops, LUTs - lookup tables, BRAMs - block RAM memory blocks). The values in percent are given with respect to all corresponding resources available in XC7Z020 device.

	FF	**LUT**	**BRAMs**
approx. median system	8612 (16,19)	9745 (9,16)	20 (14,29)
det. desc. match. system	18314 (34,42)	27422 (25,77)	55 (39,29)
XC7Z020	53200	106400	140

Both of the described test systems were implemented using the low-cost Zedboard evaluation board. It hosts Xilinx Zynq-7000 SoC (XC7Z020-CLG484-1), 512 MB of DDR3 RAM, Gigabit Ethernet port, USB host, HDMI output and FMC connector for image sensor connection and has the ability to boot Linux from the SD card.

5 Conclusions and Future Work

Using hybrid heterogeneous reprogrammable devices allows for the integration of the low-level, pixel-based and neighbourhood-based image processing algorithms alongside the more complex and sophisticated ones (like object detection and matching) with a single device. Such solution offers high performance enabling real-time image processing with a relatively low power consumption. The presented architecture makes seamless integration of various processing elements and microprocessor cores in a single device possible. It is achieved by laying down the high-performance on-chip communication infrastructure based on the dedicated DMA channels and interfaces allowing the configuration of processing elements. The infrastructure was designed to use a standard streaming interface, which simplifies the implementation its components.

Future work will focus on broadening the range of available hardware coprocessors and analysing of performance in more complex applications.

Acknowledgment. This research was financed by the Polish National Science Centre grant funded according to the decision DEC-2011/03/N/ST6/03022, which is gratefully acknowledged.

References

1. Abbo, A., Kleihorst, R., Choudhary, V., Sevat, L., Wielage, P., Mouy, S., Vermeulen, B., Heijligers, M.: Xetal-II: A 107 GOPS, 600 mW massively parallel processor for video scene analysis. IEEE Journal of Solid-State Circuits 43(1), 192–201 (2008)

2. Aghajan, H., Cavallaro, A.: Multi-camera networks: principles and applications. Academic Press (2009)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks 54(15), 2787–2805 (2010)
4. Bailey, D.G.: Design for embedded image processing on FPGAs. John Wiley & Sons (2011)
5. Bhanu, B., Ravishankar, C., Roy-Chowdhury, A., Aghajan, H., Terzopoulos, D.: Distributed Video Sensor Networks. Springer (2011)
6. Birem, M., Berry, F.: DreamCam: A modular FPGA-based smart camera architecture. Journal of Systems Architecture 60(6), 519–527 (2014)
7. Bourrasset, C., Serot, J., Berry, F.: FPGA-based smart camera mote for pervasive wireless network. In: Proc. of International Conference on Distributed Smart Cameras (ICDSC), pp. 1–6 (October 2013)
8. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a local binary descriptor very fast. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(7), 1281–1298 (2012)
9. Chen, P., Ahammad, P., Boyer, C., Huang, S.I., Lin, L., Lobaton, E., Meingast, M., Oh, S., Wang, S., Yan, P., et al.: CITRIC: A low-bandwidth wireless camera network platform. In: Proc. of International Conference on Distributed Smart Cameras, pp. 1–10. IEEE (2008)
10. Chen, P., Hong, K., Naikal, N., Sastry, S.S., Tygar, D., Yan, P., Yang, A.Y., Chang, L.C., Lin, L., Wang, S., Lobatón, E., Oh, S., Ahammad, P.: A low-bandwidth camera sensor platform with applications in smart camera networks. ACM Transactions on Sensor Networks 9(2), 21:1–21:23 (2013)
11. Di Caterina, G., Hunter, I., Soraghan, J.: An embedded smart surveillance system for target tracking using a PTZ camera. In: 2010 4th European Education and Research Conference (EDERC), pp. 165–169 (December 2010)
12. Fularz, M., Kraft, M., Kasinski, A., Acasandrei, L.: A hybrid system on chip solution for the detection and labeling of moving objects in video streams. In: Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), pp. 94–99 (September 2013)
13. Fularz, M., Kraft, M., Schmidt, A., Kasiński, A.: FPGA implementation of the robust essential matrix estimation with RANSAC and the 8-point and the 5-point method. In: Keller, R., Kramer, D., Weiss, J.-P. (eds.) Facing the Multicore - Challenge II. LNCS, vol. 7174, pp. 60–71. Springer, Heidelberg (2012)
14. Hengstler, S., Prashanth, D., Fong, S., Aghajan, H.: Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In: 6th International Symposium on Information Processing in Sensor Networks, IPSN 2007, pp. 360–369 (April 2007)
15. Jin, X., Goto, S.: Encoder adaptable difference detection for low power video compression in surveillance system. Signal Processing: Image Communication 26(3), 130–142 (2011)
16. Kandhalu, A., Rowe, A., Rajkumar, R.: Dspcam: A camera sensor system for surveillance networks. In: Third ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2009, pp. 1–7 (August 2009)
17. Kraft, M., Fularz, M., Kasiński, A.: System on chip coprocessors for high speed image feature detection and matching. In: Blanc-Talon, J., Kleihorst, R., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2011. LNCS, vol. 6915, pp. 599–610. Springer, Heidelberg (2011)

18. Ma, T., Hempel, M., Peng, D., Sharif, H.: A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. IEEE Communications Surveys Tutorials 15(3), 963–972 (2013)
19. Maggiani, L., Salvadori, C., Petracca, M., Pagano, P., Saletti, R.: Reconfigurable FPGA architecture for computer vision applications in smart camera networks. In: 2013 Seventh International Conference on Distributed Smart Cameras (ICDSC), pp. 1–6 (October 2013)
20. McFarlane, N., Schofield, C.: Segmentation and tracking of piglets in images. Machine Vision and Applications 8(3), 187–193 (1995)
21. Rosten, E., Drummond, T.W.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
22. Seema, A., Reisslein, M.: Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design. IEEE Communications Surveys Tutorials 13(3), 462–486 (2011)
23. Tomasi, M., Pundlik, S., Luo, G.: FPGA–DSP co-processing for feature tracking in smart video sensors. Journal of Real-Time Image Processing, 1–17 (2014)
24. Winkler, T., Rinner, B.: Pervasive smart camera networks exploiting heterogeneous wireless channels. In: IEEE International Conference on Pervasive Computing and Communications, PerCom 2009, pp. 1–4 (March 2009)
25. Winkler, T., Rinner, B.: Trustcam: Security and privacy-protection for an embedded smart camera based on trusted computing. In: 2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 593–600 (August 2010)
26. Xilinx Inc.: AXI Reference Guide, Version 13.4 (2012)
27. Xilinx Inc.: Zynq-7000 all programmable SoC overview (August 2012)