Design of Embedded Vision System based on FPGA-SoC

Ahmed Alsheikhy[1], Yahia Fahem Said[2]
Electrical Engineering Department, College of Engineering, Northern Border University, Arar, Saudi Arabia[1, 2]
Laboratory of Electronics and Microelectronics (LR99ES30), Faculty of Sciences of Monastir, University of Monastir, TUNISIA[2]

***Abstract*—The advanced micro-electronics in the last decades provide each year new tools and devices making it possible to design more and more efficient artificial vision systems capable of meeting the constraints imposed. All the elements are thus brought together to make artificial vision one of the most promising, even unifying scientific "challenges" of our time. This is because the development of a vision system requires knowledge from several disciplines, from signal processing to computer architecture, through theories of probability, linear algebra, computer science, artificial intelligence and analog and digital electronics. The work proposed in this paper is located at the intersection of embedded systems and image processing domains. The objective is to propose an embedded vision system for video acquisition and processing by adding hardware accelerators in order to extract some image characteristics. With the introduction of reconfigurable platforms, such as new All Programmable System on Chip (APSoC) platforms and the advent of new high-level Electronic Design Automation (EDA) tools to configure them, FPGA-SoC based image processing has emerged as a practical solution for most computer vision problems. In this paper, we are interested in the design and implementation of an embedded vision system. This design facilitates video streaming from the camera to the monitor and hardware processing over real-time FPGA-SoC.**

Keywords*—Embedded vision; video processing architecture; real-time; All Programmable System on Chip (APSoC)

I. Introduction

The discipline that aims to automate the understanding of images, computer vision, is a branch of artificial intelligence whose objective is precisely to enable a machine of perception and understand what it sees when it is connects to one or more cameras. With the emergence of increasingly powerful processors, it becomes practical to incorporate computer vision algorithms to embedded systems, to analyze their environments through video inputs [1]. Products like Mobileye's Helper System and Nest Cam surveillance camera are raising awareness of the incredible potential of embedded vision technology. As a result, many embedded system designers are beginning to implement computer vision algorithms.

Embedded vision is the integration of computer vision in embedded systems, which use an infinity of algorithms to decode the scene and obtain meanings. In fact, this type of artificial vision is developing little by little and becomes the major and most used solution in the field of industry, medical, etc [2]. It can be defined as the application of computer vision to problems. Its principle is to equip machines with the ability to see in order to automate control tasks. This automation makes it possible to increase production performance and production rates in the industry, for example, to make production more reliable, to improve the quality of products, to ensure their traceability, and to guarantee safety.

Today's progress has successfully integrated specially dedicated processors for the embedded vision to facilitate the processing task and improve the result, which helps the embedded vision to replace the majority of human functions.

Thanks to its learning and treatment capabilities, a trend towards miniaturization has been established in many areas of electronics. Embedded vision technology will be the technological future, which will be deployed widely in the fields: medical, industrial, agricultural, maritime, military and become among the important necessities in all technological industries.

Relatively good performance processors can be implemented by the users in the FPGA and speak of complete System on Chip SoC. The new FPGAs now have processor cores. FPGA-SoC based architecture with processor cores is the most appropriate solution for a context requiring to constantly evolve image processing applications, with the development of computing units of increasing complexity. In addition, a SoC platform allows different variations of computational architectures, which makes it possible to address different families of vision systems that are defined, according to constraints of performance and consumption. Although generating a natural extra cost in terms of silicium surface, an FPGA-type component makes it possible to produce low-cost vision systems while choosing the right partitioning of software and hardware operation.

An embedded vision system based on FPGA-SoC is proposed in this paper. Images captured by the sensor, are processed internally by integration of hardware and software processing in the same platform. This paper is organized as follows. Related works on embedded vision systems are presented in Section 2. Then, the proposed system architecture is described in Section 3. Section 4 details experiments and results. Finally, Section 5 concludes the paper.

II. Related Works

Computer vision is the use of computers to extract meaningful meaning from images, such as those from real-time photographs, videos, and camera streams. With the profusion of low-power parallel processors, increasing sensor availability, and an active ecosystem of algorithms, it is now

possible for many embedded devices to analyze their environments.

Embedded vision is the application of computer vision to problems. Its principle is to equip machines with the ability to see in order to automate control tasks. Recently there has been a significant increase in research in building embedded vision systems [7, 9, 10, 11, 12, 13, 16].

Bravo et al. [4] proposed an intelligent ad-hoc camera with embedded processing. A Virtex 4 FPGA device for internal image processing was used. However, the amount of resources available on the chosen device was not sufficient. An FPGA platform with more slices is needed for implementing complex image processing algorithms.

A high-speed camera with high-resolution based on a CMOS image sensor and a Xilinx Virtex-II FPGA platform was proposed by Mosqueron et al. [3]. Images with SXGA resolution (1280×1024) were acquired and processed at 500 fps. Real-time image processing algorithms (Sobel filter, fast marker extraction, etc.) were implemented.

Lee et al. [5] proposed an embedded implementation for efficient SLAM algorithm for low-cost indoor navigation robots. The NXP4330Q platform equipped with forward-viewing mono-camera was used for experiments and testing. The proposed architecture achieved high performance under various challenging constraints.

An embedded system with hardware/software co-design for stereo vision is proposed by Stefania et al. [6]. Disparity maps are optimized by custom hardware module and software routine controls data streaming and communication. High performances and accuracy was achieved by the proposed architecture.

A Computer-on-Module (CoM) for embedded vision applications based on FPGA SoC is proposed in [8]. The platform integrated a Xilinx Zynq SoC combined with Adapteva Epiphany processor. Indoor robot navigation and collision avoidance applications have been implemented and performed increased energy efficiency and accuracies.

Ismail et al. [14] proposed a real-time FPGA-based tracking system. The system is composed by DE1-SoC platform and D5M camera. Particle filter algorithm has been applied. By the proposed implementation, multiple objects with multiple colors can be tracked up to 30 meters for 15×15 cm size.

A video streaming module for smart camera system is implemented on a Xilinx Virtex-5 platform in [15]. Hardware/software co-design was performed using the Xilinx EDK design tool targeting real-time video processing applications.

Bonny introduced in [17] an algorithm for similarity measurement between two sequences in a database. The proposed method is validated on Zedboard FPGA device using custom HW/SW partitioning and achieved an acceleration time of 60% compared to other strategies.

Abdelkader et al. [29] implemented an embedded architecture for corner detection and tracking system in video sequence, using the HW/SW partitioning flexibility of Zynq platform. They used Harris algorithm for corner detection and Kalman filter for feature tracking. The processing time of the proposed system was improved by 50% using several approximations and innovations.

III. Proposed Architecture

An embedded system with a camera and a monitor, together form a complete vision system, which allows the acquisition of video data and, depending on the application that will run, the processing can be real-time. In this paper, using a platform-based design approach, an embedded architecture for the acquisition of video and processing modules for the design of a vision system is proposed. This design facilitates real-time video streaming from the sensor to the monitor via DDR3 memory and hardware processing over FPGA. The proposed system architecture is shown in Fig. 1. It consists of a VITA-2000 image sensor [18], the 'FPGA Mezzanine Card' (FMC) module [19], the Xilinx ZedBoard platform [20] and a HDMI monitor for viewing the output video.

The VITA-2000 is an Ultra eXtended Graphics Array (WUXGA) CMOS camera configurable in HD (1920 x 1080) or (1600 x 1200) format [18]. The FMC module [19] provides several high definition video interfaces. It features on-board HDMI I/O interfaces, LCEDI interface to connect the VITA-2000, as well as an LPC interface to connect to the FPGA. The FMC module is ideal for developing vision, motion monitoring and security applications.

The ZedBoard development platform (XC7Z020 CLG484-1), is a low cost Zynq-7000 platform whose FPGA is based on Artix-7, with a capacity of 13300 Logic Blocks, 220 DSP48E1 and 140 RAM Blocks (32K) [20]. ZedBoard provides an ideal platform for the implementation of flexible SoCs: 'All-Programmable SoC (AP SoCs)'. It consists of two main parts: a Processing System (PS) built around a dual core ARM Cortex-A9 processor and Programmable Logic (PL) equivalent to that of an FPGA. It also has built-in memory, a variety of peripherals, and high-speed communication interfaces.

The PL section is ideal for implementing logical subsystems, arithmetic and high-speed data flow, while the PS supports routines and/or operating systems, which means that the functionality overall system can be properly partitioned between hardware and software. The links between PL and PS are made using the Advanced eXtensible Interface (AXI) interface [21].

Fig. 2 shows the Vita-2000 camera that is connected to the FMC module via the LCEDI port and coupled with the ZedBoard platform via the LPC port. In this design, the ARM processor is used for interfacing the different FPGA-based IPs used for streaming, as well as configuring platform peripherals. The software environment of the system consists of a software application and drivers of peripherals. The hardware part of the system includes configurable logic blocks, AXIs interfaces and different memories used such as DDR3. This integration of software and hardware provides the complete functionality of the proposed system.

Xilinx provides in its tool pack, various software for the creation of on-chip embedded systems. These tools include Vivado Design Suite, Vivado HLS and SDK. These three tools offer us the possibility to have a Bitstream for FPGA programming according to the targeted application.

The Vivado Design Suite is a Xilinx tool, provides a design flow that revolves around the philosophy of IP design, reuse and integration. It makes it possible to create IP modules, but also to pack IP modules for repeated use and to easily integrate third party applications [22]. The custom IP is packaged for use, according to IP-XACT before being available in the IP catalog.

Vivado High-Level Synthesis HLS [25] is part of the Xilinx tools suite. The tool is based on an Eclipse integrated development environment (IDE) [24] and uses precompile directives to generate RTL code from C code. The IDE also includes RTL compilation and simulation of the component. During the transcription process, the tool generates representations in SystemC, VHDL and Verilog. It is possible to directly export a component as an IP block from the HLS, easily integrated into a project on Xilinx AP SoC devices. It speeds up the creation of intellectual property (IP) without having to RTL manually.

The Software Development Kit (SDK) is an Xilinx tool that provides an environment in which fully functional software applications can be created, compiled, and debugged into a single tool. It is used for the C code development of the ARM processor included in the Zynq SoC. It includes Board Support Package (BSP), drivers for IPs, the GNU compiler database, and libraries for application-specific functions [23].

At first, the proposed design is used to display in real time on a monitor the scene observed by the camera without any processing that is to say: the video data is captured by the camera with a resolution of 1920 x 1080P at 60MHz. Then, this data is sent to the FPGA, where there is a chain of IPs to bring this data to the output: the 'VITA Receiver' IP is responsible for receiving the video data that will subsequently undergo operations processing pipeline to correct defective pixels with the IP 'DPC' (Defective Pixel Correction), and apply the interpolation filter with the IP 'CFA' (Color Interpolation Filter) which is used to interpolate the missing colors for each pixel; followed by another treatment with the IP 'TPG' (Test Pattern Generator) which is used to evaluate the color performance and the quality of movement (see Fig. 3). After all these treatments, the video data will be stored in a 'VDMA' (Video Direct Memory Access) memory.

Image processing pipeline will be controlled by the PS, which synchronizes the different IP blocks and manages the different clocks to each operating domain via AXI interfacing, as well as initializing platform devices and controlling video processing by reading and writing control registers in the system. After recording to the VDMA, the video data will undergo the last operation with the IP 'Xylon Controller' before being displayed on the screen. Among these operations that contain the display IP 'Xylon Controller', we note the conversion RGB to YUV, 444 to 422 (in IP LOGICVC_0) to transfer this data to IP 'HDMI OUTPUT' which ensures the output of data to the monitor.

Fig. 3 shows a screenshot of the Co-designed system including different IPs that are connected with the ARM processor (Zynq PS).

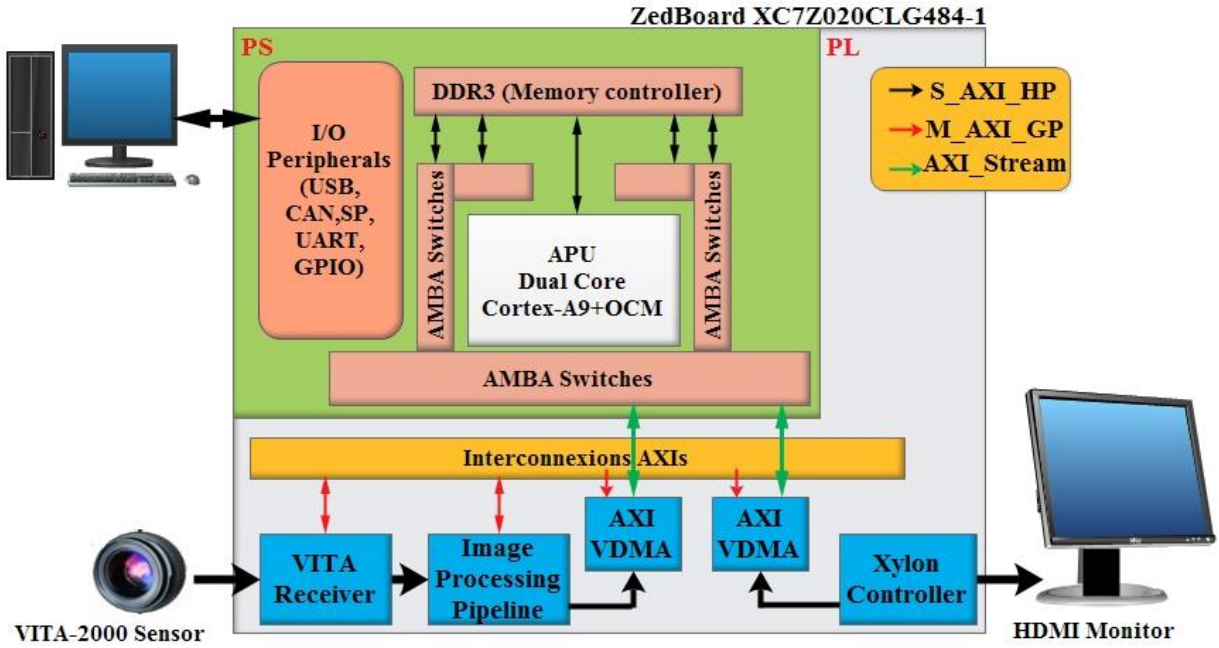

Fig. 1. Proposed Embedded Video Processing Architecture.

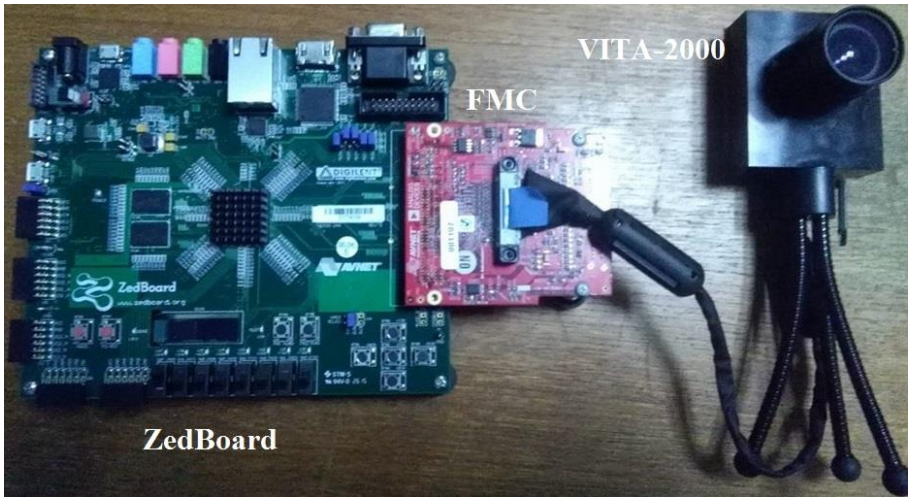

Fig. 2. The ZedBoard Platform Coupled with the FMC Module and the VITA-2000 Camera.

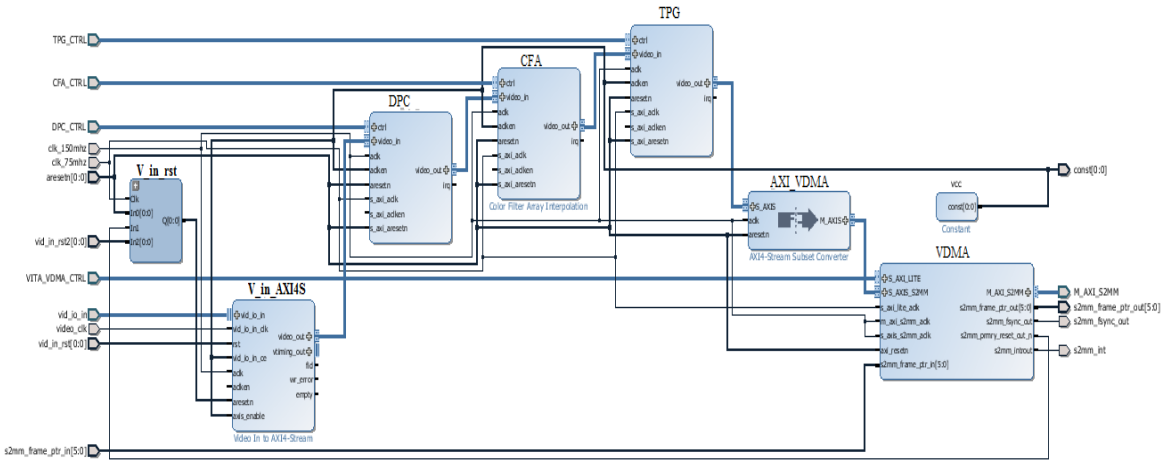

Fig. 3. Screen Capture of IPs that Constitute the Image Processing Pipeline.

IV. Experiments and Results

The previous proposed system is none other than an architecture that can process video data in real time without any operation to be applied to the captured images. In the following, this architecture must keep the processing in real time, but this time with a processing operation on the images, in the form of hardware accelerators.

To validate the proposed embedded video processing architecture, two algorithms were added: the 'FAST Corners' filter and the 'Sobel' filter, which are designed from the OpenCV library and the Vivado HLS tool, which is used to synthesize the C codes of the library used and make it in the form of a hardware accelerator, which is subsequently going to be exported as an IP block usable in the architecture.

These accelerators (FAST/Sobel) must work in synchronization with the other IP blocks and process the data in pipeline so that the architecture remains functional in real time. The general concept is that the added processing IP is connected with the VDMA via the AXI VDMA (see Fig. 4). A 'Video Processing' block is added which contains the IP filter as well as an AXI interface which guarantees the communication of the ARM processor with the proposed accelerator and the communication of the latter with the VDMA.

A. Design of the IP Fast_Corners Filter and Implementation

In computer vision and image processing, the detection of areas of interest of a digital image (feature detection) consists in highlighting areas of this image deemed "interesting" for the analysis, that is, with remarkable local properties. Such zones can appear, according to the methods used, in the form of points, of continuous curves, or incurring connected regions rectangular or not and which constitute the result of the detection.

Points of interest detection algorithms generally focus on particular selected contour points, according to a specific criterion. Thus the corners are the points of the image where the contour (of dimension 1) abruptly changes direction.

The 'Feature from Accelerated Segment Test' (FAST) method is a corner detection algorithm introduced by Cambridge University researchers for the first time in 2006 [26]. Angle detection or 'Fast Corners' algorithm is frequently used in computer vision systems for object detection, image recording and 3D reconstruction.

The algorithm works in two steps: in the first step, a segment test based on the relative luminosities, which are applied to each pixel of the processed image. The second step allows to refine and limit the results by the method 'non-maximum suppression'. The whole process can be summarized mathematically as follows: Where 't' is the threshold for detection.

$$V = \max \begin{cases} \sum (\text{pixel values} - p) \text{if}(\text{value} - p) > t \\ \sum (p - \text{pixel values}) \text{if}(p - \text{value}) > t \end{cases}$$

The design of an IP block requires source codes. The OpenCV library offers these codes in C or C ++. These source codes are the essential and the basis of the design, which will be synthesized by the HLS Vivado in order to implement it as a hardware accelerator.

Vivado HLS also provides the simulation of a test Bench code: it requires a single image data. Vivado HLS treats both the synthetic code (hardware) and the OpenCV version (software) and he looks for differences between these two approaches and reports them as a failure (because these functions should be identical). The result of Co-simulation is given in Fig. 5. The summary report gives us an idea of the performance such as the frequency of the IP block, the hardware resources used on the target platform. All these results are presented in Table I, compared with the design proposed by [27], implemented on a ZedBoard FPGA card.

As this table shows, the design of [27] requires 4 BRAMs, 5963 LUTs and 8281 Flip Flops, with a maximum frequency of 134 MHz. On the other hand, the proposed IP required about 11 BRAMs, 8425 LUTs and 4252 Flip Flops with a working frequency of 150 MHz. With this implementation, this hardware architecture achieves high-speed performance with fairly large hardware resources than the design of [27], which is explained by the increased degree of parallelism, due to the numbers of nested loops added in the proposed design.

The integration of the IP block is done by exportation in the IP Integrator (IPI) library. Once the filter block is added, the design should be validated without error, and the IP address should be checked and updated. Subsequently, the implementation of the proposed architecture with IP FAST_Corners on the ZedBoard platform is completed by Xilinx Vivado Design Suite. Fig. 6 shows the experimental result of the proposed architecture with IP FAST_Corners. Table II shows the hardware resources required and the maximum operating frequency of the complete system (proposed architecture and IP FAST_Corners).

To calculate the video data processing rate, the programmable logic design contains three submodules that implement video pipelines in the design: Video Capture, Video Processing, and Video Display. These submodules communicated with VDMA memory with a 1080p60 video resolution with an image quality of 1920x1080 i.e. that the video rate is 60 frames/second and with a 32bit memory bus width, the broadcast rate of the video data in the memory can be calculated: 1920 * 1080 * 60 /s * 32 = 4 GB/s.

TABLE. I. QUANTITY OF HARDWARE RESOURCES USED BY IP FAST_CORNERS COMPARED WITH OTHER DESIGN

Resources		**Proposed**		**[27]**	
	Available	**Used**	%	**Used**	**%**
BRAMs	140	11	7	3	2
LUTs	53200	8425	15	5963	11
Flip Flops	106400	4252	4	8281	7
Maximum frequency		150 MHz		134 MHz	

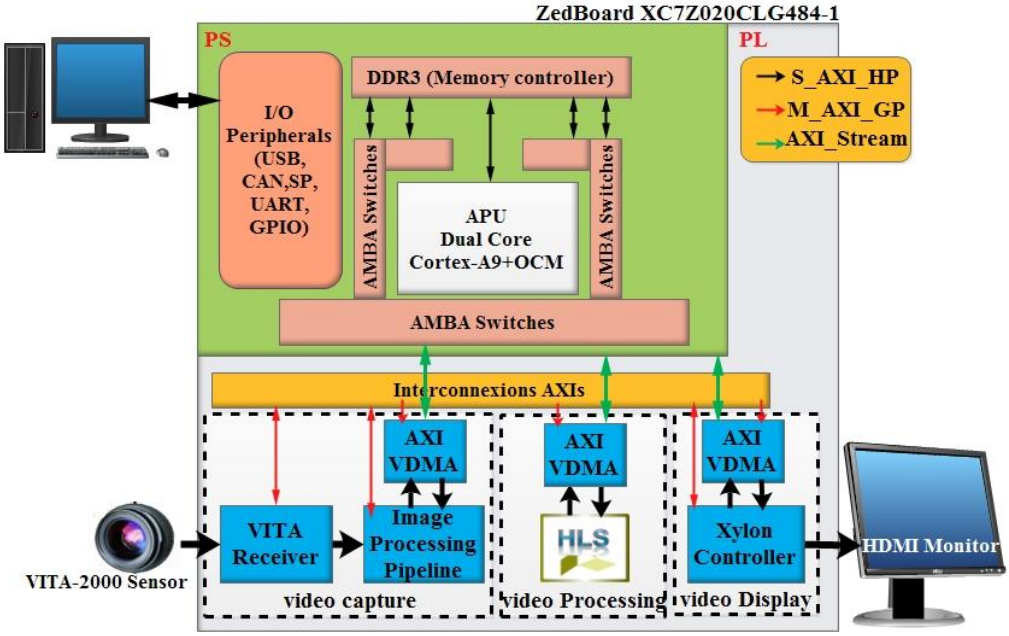

Fig. 4. Embedded Vision System Architecture with Added Video Processing.

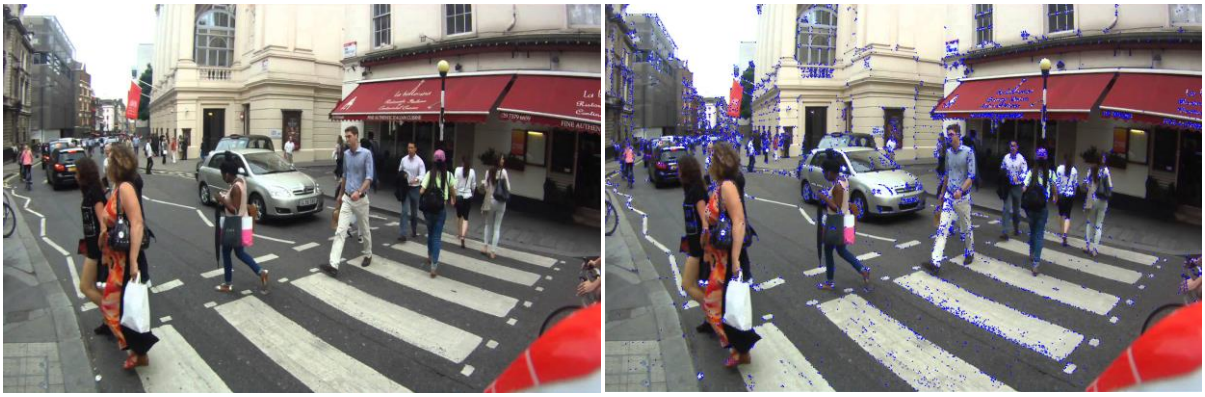

Fig. 5. Test Image (Left) and the Result after the FAST Corners Filter (Right).

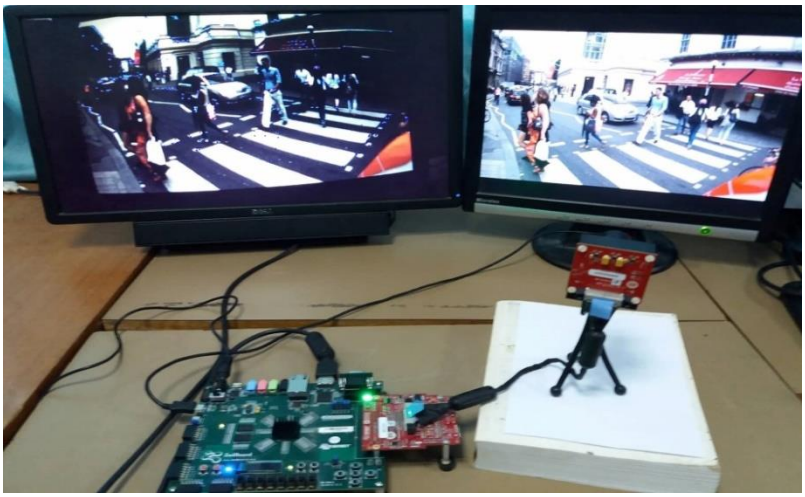

Fig. 6. Experimental Result of the Proposed Architecture with IP FAST_Corners.

TABLE. II. RESULTS OF THE SYNTHESIS OF PROPOSED SYSTEM WITH IP FAST_CORNERS

Resources	**Available**	**Used**	%
BRAMs	140	90	64
LUTs	53200	32770	62
Flip Flops	106400	34306	32
DSP48E1	220	44	20
Maximum frequency	150 MHz		

B. Design of the IP Sobel Filter and Implementation

Edge detection is the process of locating the image points that correspond to a sudden change in light intensity. It is a basic operation in many image processing applications. Sobel Filter is one of the efficient algorithms for edge detection. It uses convolution matrices. The 3×3 matrix is convolved with the image to calculate approximated horizontal and vertical gradients Gx and Gy as follows:

$$Gx = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \quad \text{and} \quad Gy = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

Then, the gradient magnitude G is calculated:

$$G = \sqrt{Gx^2 + Gy^2}$$

The Sobel edge detector is built as a hardware video processing accelerator, using Xilinx Vivado HLS and the OpenCV library, by applying the same procedures that were applied to FAST_Corner IP. The result of the Co-Simulation of the Sobel filter is given in Fig. 7. Table III shows the amount of hardware resources used by the proposed IP, compared with the design of [28], implemented on a Zynq 7000 FPGA-SoC card. The proposed Sobel edge detection IP occupies 6% of RAM blocks, 1% of Flip Flops, and 4% of LUTs, with a maximum frequency of 150MHz. On the other hand, the design of [28] requires 2% BRAMs, 1% LUTs and 1% Flip Flops with the same 150MHz working frequency.

The integration of IP Sobel into the design is done first by exporting the IP from the HLS to the IP Integrator IPI library in RTL form, afterwards just add in the design diagram and make connections, in the same way as IP FAST_Corners. The IP Sobel will communicate with the VDMA memory via AXIs interfaces with a bit rate of 4Gb/s. The result of the complete system implementation (proposed architecture and IP sobel) also allows us to determine the numbers of material resources consumed (see Table IV). The experimental result of the IP Sobel implementation is shown in Fig. 8.

Fig. 7. Test Image (Left) and the Result after the Sobel Filter (Right).

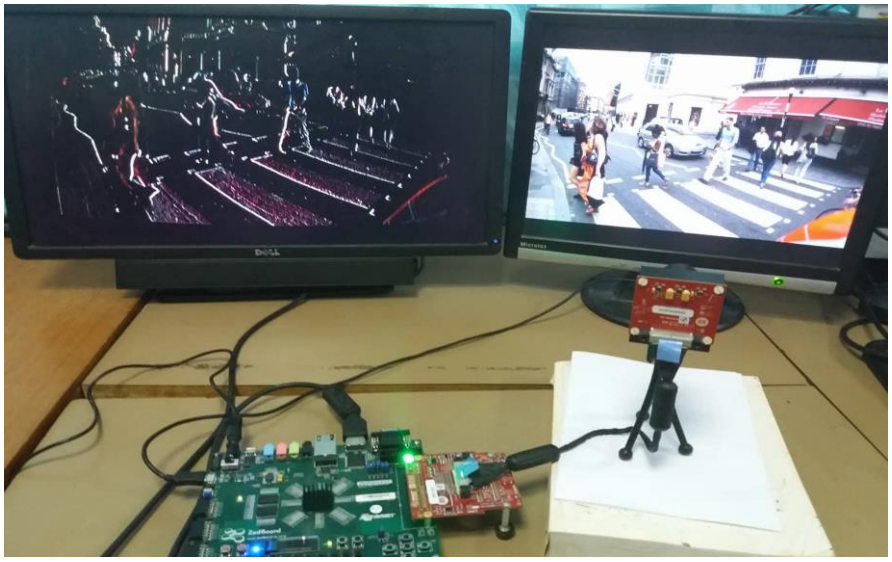

Fig. 8. Experimental Result of the Proposed Architecture with IP Sobel.

TABLE. III. QUANTITY OF HARDWARE RESOURCES USED BY IP SOBEL COMPARED WITH OTHER DESIGN

Resources		**Proposed**		**[28]**	
	Available	**Used**	%	**Used**	**%**
BRAMs	140	9	6	3	2
LUTs	53200	1946	4	746	1
Flip Flops	106400	1445	1	579	1
Maximum frequency		150 MHz		150 MHz	

TABLE. IV. RESULTS OF THE SYNTHESIS OF PROPOSED SYSTEM WITH IP SOBEL

Resources	**Available**	**Used**	%
BRAMs	140	65	46
LUTs	53200	29951	56
Flip Flops	106400	31989	30
DSP48E1	220	39	18
Maximum frequency	150 MHz		

V. CONCLUSIONS

In this paper, an embedded vision architecture implemented on a FPGA-SoC platform was proposed. Firstly, a system that allows the acquisition of video data in real time at 60 fps and displays it on a monitor was designed. We subsequently integrated two hardware accelerators that are designed using the OpenCV and the Vivado HLS. The first accelerator is the FAST_Corner which is used to detect corners in an image/video, the second accelerator is the Sobel which is used to detect the edge of objects. The sequence is obtained from a VITA_2000 sensor and the output is delivered through an HMDI monitor to check the video processing results in real time.

ACKNOWLEDGMENT

The authors wish to acknowledge the approval and the support of this research study by the grant N° ENG-2018-3-9-F-7705 from the Deanship of the Scientific Research in Northern Border University, Arar, KSA.

REFERENCES

[1] Embedded Vision Alliance: https://www.embedded-vision.com.

[2] B. Deepayan, and K. Appiah. "Embedded vision systems: A review of the literature," In International Symposium on Applied Reconfigurable Computing, pp. 204-216. Springer, Cham, 2018.

[3] R. Mosqueron, J. Dubois, M. Paindavoine, "High-Speed Smart Camera with High Resolution," J. Embed. Syst, 2007.

[4] I. Bravo, J. Balinas, A. Gardel, J. L. Lazaro, F. Espinosa, and J. Garcia, "Efficient smart CMOS camera based on FPGAs oriented to embedded image processing," Sensors, vol.3 , pp.2282- 2303, 2011.

[5] L. Tae-Jae, C. H. Kim, and D. D. Cho. "A monocular vision sensor-based efficient SLAM method for indoor service robots," IEEE Transactions on Industrial Electronics vol.66, no .1, pp.318-328, 2019.

[6] P. Stefania, et al. "Design of Real-Time FPGA-based Embedded System for Stereo Vision," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2018.

[7] D. T. Tin, et al. "Evaluation of Embedded Systems for Automotive Image Processing," 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2018.

[8] K. Daniel, et al. "Resource-efficient Reconfigurable Computer-on-Module for Embedded Vision Applications," IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2018.

[9] E. Osman, and A. M. Mutawa. "Embedded vision system with hardware acceleration," IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), IEEE, 2018.

[10] G. Li, et al. "Lossy Compression for Embedded Computer Vision Systems," IEEE Access, vol. 6, pp. 39385-39397, 2018.

[11] C. Bui et al., "A Hardware/Software Co-Design Approach for Real-Time Object Detection and Tracking on Embedded Devices," IEEE SoutheastCon 2018, pp.1-7, 2018.

[12] P. J. Gopal, et al. "Platform-Based Design Approach for Embedded Vision Applications," Journal of Image and Graphics, vol. 1, no. 1 ,pp.1-6, 2013.

[13] B. Deepayan and A. Kofi (2018). "Embedded vision systems: A review of the literature," 14th International Symposium on Applied Reconfigurable Computing (ARC), Santorini, Greece, 2-4 May 2018.

[14] S. N. Ismail, H. S. Muataz, and Y. Wahab, "Design and implementation of embedded vision-based tracking system for multiple objects using FPGA-SOC," ARPN Journal of Engineering and Applied Sciences, Vol. 13, No. 3, February 2018.

[15] J. G. Pandey, S. Purushottam, A. Karmakar, and C. Shekhar "Platform-Based Extensible Hardware-Software Video Streaming Module for a Smart Camera System," International Journal of Modeling and Optimization, vol. 2, no. 4, August 2012.

[16] A. Oetken, S. Wildermann, J. Teich, D. Koch, "A Bus-based SoC Architecture for Flexible Module Placement on Reconfigurable FPGAs," International Conference on Field Programmable Logic and Applications, 2010.

[17] B. Talal, "Heterogeneous HW/SW FPGA-Based Embedded System for Database Sequencing Applications," International journal of advanced computer science and application, IJACSA, vol. 9. no 10, pp. 244-251, 2018.

[18] VITA2000: CMOS Image Sensor, Global Shutter, 2.3 Megapixel: http://www.onsemi.com/PowerSolutions/product.do?id=VITA2000.

[19] FMC cards: https://www.xilinx.com/products/boards-and-kits/fmc-cards.html.

[20] Avnet zedboard: http://www.zedboard.org/overview-zedboard-kit.

[21] Xilinx, Inc., 'AXI Reference Guide', UG761, v14.3, November 2012: http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug761_axireference_guide.pdf.

[22] Vivado Design Suite User Guide UG910 (v2014.1) April 2, 2014: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug910-vivado-getting-started.pdf.

[23] Xilinx, Inc., "Zynq-7000 All Programmable SoC Software Developers Guide", UG821, v9.0: http://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf.

[24] Eclipse desktop IDEs: https://eclipse.org/ide/.

[25] Vivado High-Level Synthesis: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html.

[26] OpenCV FAST Algorithm for Corner Detection: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html.

[27] F. Brenot, P. Fillatreau, qnd J. Piat, "FPGA based accelerator for visual features detection," 2015 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), IEEE, pp. 1-6, 2015.

[28] A., Haythem, A. Helali, H. Maaref, and A. Youssef, "Implementation of real time edge detection system for HD video on Zynq," 3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16), Proceedings of Engineering and Technology (PET), pp. 41-45, 2016.

[29] A. B. amara, M. Atri et al., "Zynq FPGA based and Optimized Design of Points of Interest Detection and Tracking in Moving Images for Mobility System," International Journal of Advanced Computer Science and Applications, IJACSA, vol. 9, no. 10, pp. 430-437, 2018.