EasyChair Preprint

№ 9087

Engineering of Hand Recognition and Control Co-Design Using Real Time Processing

Abdelhalim Hazzem, Lahcene Ziet, Rafik Torche and Fayçal Radjah

October 24, 2022

2nd International Conference on Engineering and Applied Natural Sciences

https://www.iceans.org/

October 15-18, 2022, Konya, Turkey

Engineering of Hand Recognition and Control Co-Design Using Real-Time Processing

Hazzem Abelhalim*[1], Ziet Lahcene[1], Torche Rafik [1], and Radjah Fayça1[1]

[1]*Electronics department/technology faculty, Ferhat Abbas Setif1 University, Algeria*

[*]*(abdelhalim.hazzem@univ-setif.dz) Email of the corresponding author*

Abstract – Gestures are a powerful communication channel that is an important part of information transfer in our daily life.With co-design hybrid systems engineering containing CPU and FPGA components is an exciting new development that offers customized, compact, lightweight and low-power (c-swap) solutions for the realization of control and control by hand becomes more exciting.

engineering reusable components (c-swap) in a wide range of real-time and embedded applications, particularly in real-time computer vision, is a challenge, in part because it requires the simultaneous satisfaction of a or more and sometimes the contradiction of one or more aspects, such as execution speed, completion time, cost price and c-swap.

Keywords – Co-design Fpga Pynq-ZU Zynq Recognition Hand Python Computer-Vision

I. İNTRODUCTİON

Gestures are a powerful communication channel that forms an important part of information transfer in our everyday life. Compared to traditional devices, gestures are less intrusive, simpler, more comfortable and natural way for users to interact.

Nevertheless, the expressiveness of gestures remains little studied to solve the problem of human-computer interaction. Modern real-time systems offer an opportunity to achieve these. With the rise of hybrid processors and the development of embedded systems engineering; Control by hand gestures offers an answer to various problems in all areas of everyday life.

In this work we made an overview of the technique used with a work platform of which we made a small illustration example.

II. CO-DESIGN HYBRID ARCHITECTURE

Co-design hybrid systems engineering containing CPU and FPGA components are an exciting new development that offers customized, compact small low-weight low-power (c-swap) solutions while supporting hardware customization.

Real-time computer vision and embedded systems engineers are continually challenged to deliver increased computing capabilities to meet the most stringent requirements with ever-improving performance-to-time ratios. Best practices have long encouraged the use of components (c-swaps) to provide significantly efficient solutions while controlling design costs and lead times.

The engineering of reusable components (c-swap) in a wide range of real-time and embedded applications, in particular in real-time computed vision, is a challenge, partly because it requires the simultaneous satisfaction of one or more and sometimes contradiction of one or more aspects,

such as execution speed, completion time, cost price and c-swap.

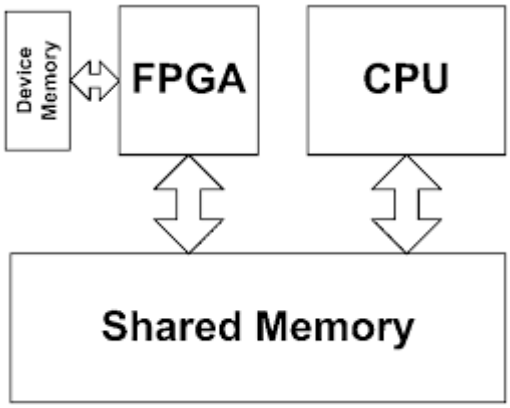

FIG. 1 CO-DESIGN HYBRID DIAGRAM

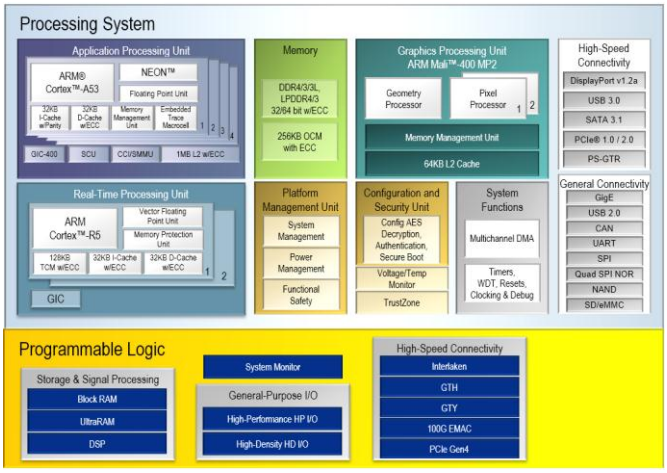

FIG. 2 ZYNQ-EG BLOCK DIAGRAM

(FPGA) are an exciting new development momentum that promises economies of scale (c-swap), while offering significant hardware customization.For instance.

Zynq® ultrascale+™ mpsoc devices offer 64-bit processor scalability while combining real-time control with software and hardware engines for graphics, video, waveform and packet processing. Built on a common real-time processor and programmable logic-equipped platform, three distinct variants include dual application processor (cg) devices, quad application processor and gpu (eg) devices and video codec (ev) devices, creating limitless possibilities for applications such as 5g wireless, next-generation adas, and ındustrial ınternet of things.

designers can program the free fpga gates with an expanded range of standard fpga i a intellectual property (IP) system library components, including serial and parallel ı/o interfaces, bus arbiters, controllers, and controllers. İnterrupt priority and dram controllers.Now have the freedom to select a set of ıp fpga components to create a specialized system-on-chip (SOC) solution.

- Quad arm cortex-a53
- Dual arm cortex-r5f
- 16nm finfet+ programmable logic
- Arm mali™-400mp2

III. FPGA (C-SWAP) AND VISION PROCESSING ENGINEERING :

In this part we see the final layout of the application that demonstrates the efficiency of high performance FPGAs. This design, which uses an FPGA-based vision processing application, because video engineering critically depends on the ability of human beings to analyse complex visual scenes. However, as humans are prone to errors. These critical tasks can be handled effectively if computer vision techniques are used. It is an established fact that computer vision is potentially capable of meeting the needs of a wide variety control tasks, remote monitoring and contactless command. However, in these modern applications, "embedded vision" is more relevant than existing vision systems. Embedded vision systems must be very compact (c-swap) and operate in very harsh and unstructured environments while providing high quality images. It should be mentioned that embedded vision is still an emerging technology, to date there are generally two main types of processors used in embedded systems - programmable gate arrays (FPGAs) and graphics processing units (GPUs). In recent years, FPGAs have gained in popularity as embedded vision processors compared to GPUs or general purpose processors. FPGAs are much faster than CPUs and hence they are gaining popularity due to their extremely low latency levels.

Similarly(c-swap) are also much more processing potential with much lower power consumption, size, occupied space, and they can speed up multiple portions of a computer vision pipeline.

A. *High performance vision processing using the ZYNQ-ZY FPGA:*

Architecture the new ZYNQ-ZU FPGA from Xilinx is based on the ZUNQ architecture, is a SOC designed by Xilinx. ZYNQ devices include a processing system (PS) and programmable logic (PL), the PL being equivalent to that of a field-programmable pre-broadcast array (FPGA), and the PS side contains quad core high-performance arm cortex A53 processor, 2 R5 as shown in figure 3.

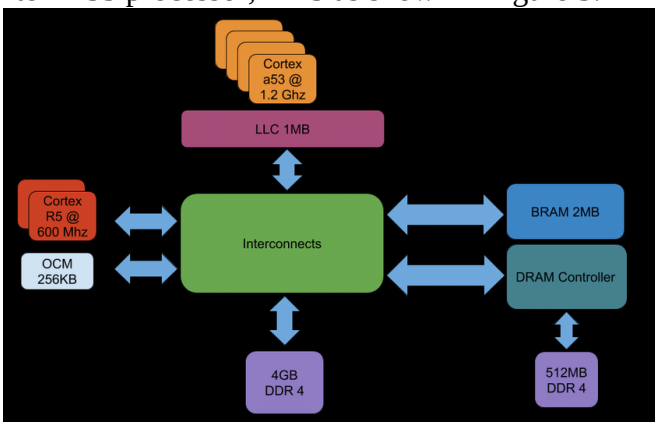

FIG. 3 SIMPLIFIED ZYNQ EG BLOCK DIAGRAM

We use PYNQ-ZU board as shown in figure 04

FIG. 4 PYNQ-ZU BOARD

PYNQ-ZU FPGA offers a tool to create and develop applications using python programming. It is designed on ZYNQ-EG FPGA. The name PYNQ is derived from python productivity for ZYNQ-EG. Three-layer PYNQ framework. The bottom layer represents the basic hardware design. This is normally created in VIVADO using IP integrator and related design tools and then output to a bit-stream (bit) file. The middle layers of the PYNQ-ZU consist of python software, the operating system, and low-level software drivers that can access low-level hardware. At the top level, user interaction is facilitated by the python development framework like JUPYTER. This framework is shown in figure 5.

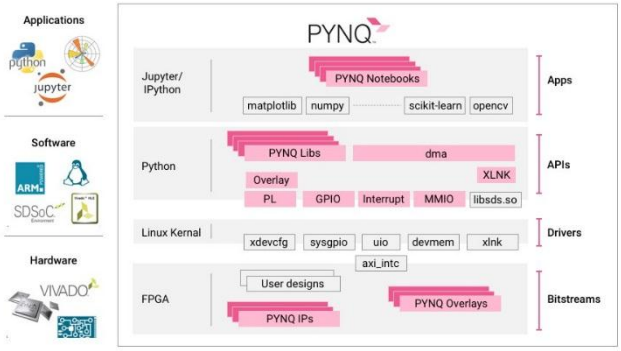

FIG. 5 PYNQ-ZU FRAMEWORK

B. *real-time video processing:*

Real-time video processing is used in a wide variety of applications ranging from remote monitoring and traffic management to medical imaging applications. These operations generally require very high computing power. However, the PYNQ-ZU used provides the performance needed for real-time processing.

Edge detection is a fundamental tool used in most image processing applications to obtain information from images as a preliminary step for feature extraction and object segmentation. This process detects the edges of an object and the boundaries between objects and the image background. An edge detection filter can also be used to improve the appearance of blurry or anti-aliased video streams. The basic edge detection operator is a matrix surface gradient operation that determines the level of variance between different pixels. This therefore requires intensive calculations.

Possible application hand recognition and control, this image acquisition and processing system will aid in hand identification and symbol recognition.

More precisely, the image stream of the video in real time. When the controller's hand appears the system detects the area of interest (roi); then it launches the process of detecting the symbol expressed by this one. This systems will be detailed in the third part.

C. *Pynq-zu based hardware design :*

In PYNQ, hardware system designs are referred to as overlays. They can be used in a manner

analogous to software libraries. Specifically, overlay represents complete hardware system that will be programmed onto the PL (fpga side), and it represents part of the hardware/ bottom layer of the pynq-zu framework. In this design we have used two overlays:

1. The pynq-zu base overlays for basic I/O communication and video interface.
2. The computer vision overlays to carry out video processing. This overlays are created using vivado hls software. The CV overlay is shown in figure 6.

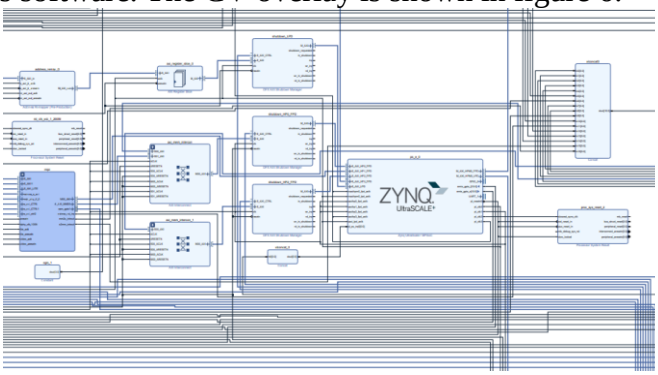

FIG. 6 PYNQ-ZU OVERLAY

Each vivado block of this design (as shown in figure 7) represents each computer vision functions. This functions are written in high level language and corresponding hardware version has been generated using vivado hls.

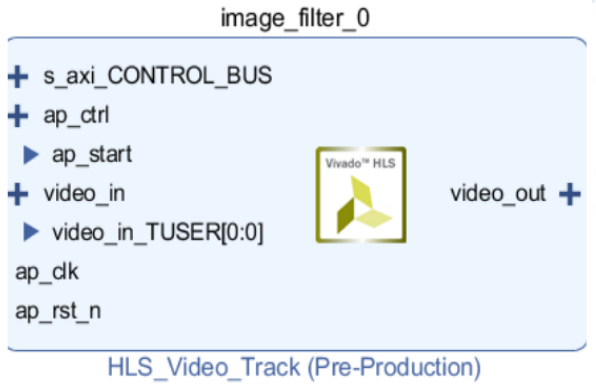

FIG. 7 CORE CREATED WITH VIVADO HLS

D. *Pynq-zu based software design :*

As a software development, the video processing is carried out using python. It has to be noted that the python coding is not used for hardware description and verification, i.e., to generate circuits for implementation in the pl. Section. Python is used for programming on the PS only, which includes interaction with hardware via the PS-PL interfaces. This means that the python interface executed on PS side and it communicates with PL. In the PL side, computation intensive processing is carried out.

E. *Using FPGAs and CPUs together*

Fpgas are very well suited for image processing due to their ability to exploit parallelism. They are capable of achieving real-time performance in many applications. Often, image processing algorithms on fpgas are implemented as unique designs designed to speed up a particular task. Additionally, edge and corner detection are two very popular operations to perform on fpgas. There have been many implementations of edge and corner detection on fpgas, but very few offer comparisons with other architectures. One of the few that makes this comparison is by possa, et al. They compared canny edge detection implementations on a CPU, gpu, and fpga. The results of the work are presented in the following two tables. This work shows an order of magnitude increase in frame rate from the CPU to the fpga, and a three orders of magnitude reduction in power consumption for the fpga below the CPU or GPU. The authors report a 544x speedup for an fpga implementation of the canny edge detection algorithm, as well as 9.72x and 1.57x speedup for two image registration algorithms. This is an exemplary study of how fpgas can dramatically increase performance, but only in certain situations.

Table 1. Canny edge result performance

Resolution H*W	**Cpu** mS	**Gpu** mS	**Fpga** mS
512*512	30	2.11	1.1
1024*1024	101	6.08	4.37
1476*1680	267	13.9	10.31
3936*3936	1497	59.94	64.16

Table 2. Energy usage for canny edge in mJ

Resolution	**Cpu**	**Gpu**	**Fpga**
512*512	4200	500	1.6
1024*1024	14800	1500	6.4
1476*1680	39800	3400	15
3936*3936	229000	15000	64.16

An alternative method to using fpgas is to use them as accelerators in tandem with at CPU. Using an fpga together with a CPU is a useful way to decrease the time cost of developing fpga firmware while still getting some of the benefits. In general, new algorithms and methods are developed first for CPUs, so it is relatively easy to find an existing CPU implementation of a method. Fpga implementations

of image processing algorithms, on the other hand, are very difficult and time consuming to develop. To

Alleviate the challenge of building an entire fpga image processing pipeline, a CPU and fpga can be used together. This allows the system to be built and deployed quickly, while time is spent to develop fpga modules only for the specific pieces of the algorithm that are most in need of acceleration.

IV. HAND RECOGNITION AND CONTROL

In this example, we use the Haar classifier to detect the hand applied to a real-time vision source. Hand segmentation offers the means to apply the algorithm in the region of interest and save time during treatment.

The algorithm sequence for the detection of the symbol is illustrated in order in all of the following figures.

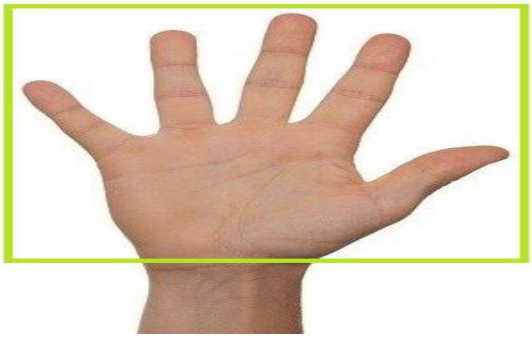

FIG. 8 HAND DETECT USING HAAR CLASSIFIER

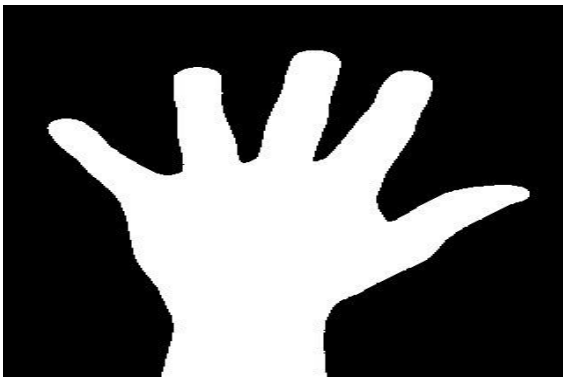

FIG. 9 HAND IMAGE SEGMENT BINARIZATION

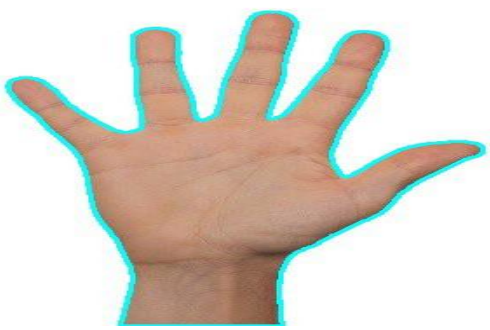

FIG. 10 CONTOUR AREA OF IMAGE SEGMENT

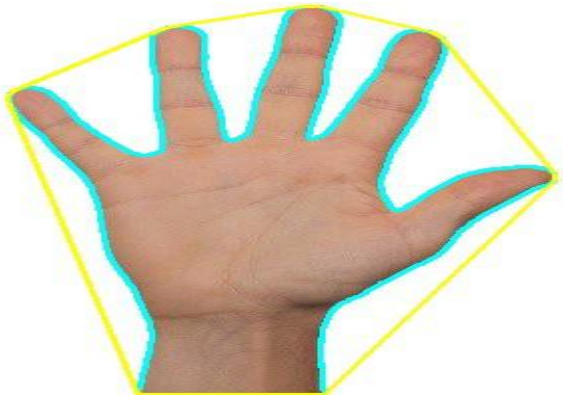

FIG. 11 CONVEX HULL AROUND MAX COUNTOUR

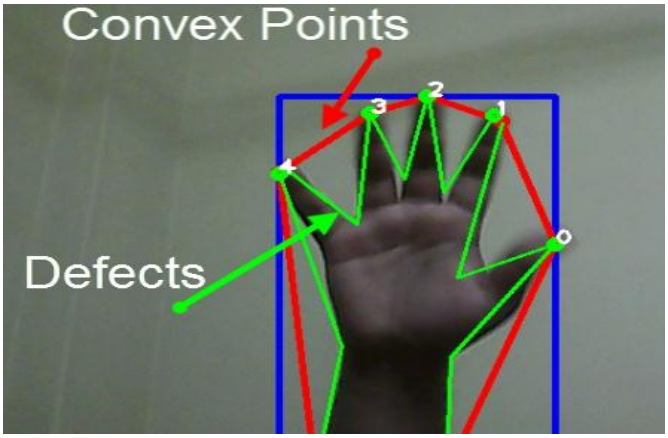

FIG. 12SYMBOL DETECT BY GEOMETRIC PROCESS

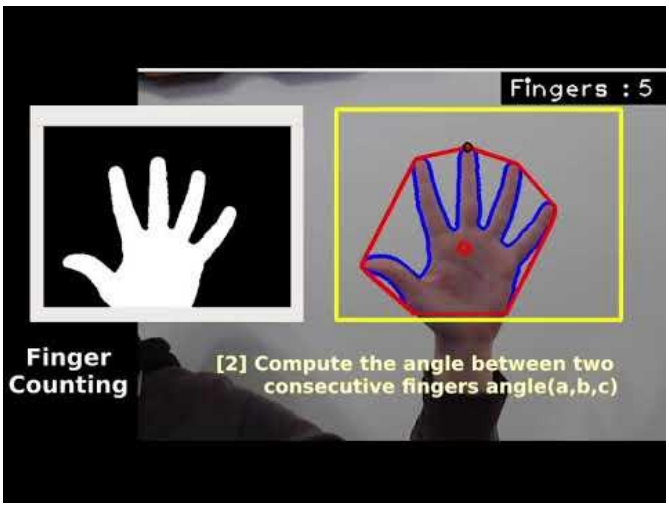

FIG. 13 DISPLAY OF SYMBOL CORRESPONDENCE

V. CONCLUSION

In this work we have seen control by hand in a hybrid environment. the new performances of co-design open up a new way of development. as a perspective for this work we propose the

exploitation of npu and cnn models in order to brings improvements.

REFERENCES

[1] Spinola CG, Canero J, Moreno-Aranda G, Bonelo JM, Martin-Vazquez M. Real-time image processing for edge inspection and defect detection in stainless steel production lines. In2011 IEEE International Conference on Imaging Systems and Techniques 2011 May 17 (pp. 170-175) IEEE

[2] JY Mon, 1. Arias-Garcia, C Sánchez-Ferreira, D. M. Muñoz, CH. Llanos, and J

[3] MS. T. Motta, "An FPGA-Based Omnidirectional Vision Sensor for Motion:

[4] Detection on Mobile Robots," Int. J. Reconfigurable Comput. pp. 1-16, 2012. J. Nikobe, J. Rehder, M. Bumn, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in 2014 IEEE International Conference on Robotics .

[5] and Automation (ICRA), 2014, pp. 431-437. M. Russell and S. Fischaber, "OpenCV based road sign recognition on Zynq." in 2013 11th IEEE International Conference on Industrial Informatics (INDIN), 2013, pp. 596-601.

[6] S. Neuendorifer, T. Li, and D. Wang, "Accelerating OpenCV Applications with

[7] Zynq-7000 All Programmable SoC using Vivado HLS Video Librarie," is W vol. 1167, p. 1, 2013.

[8] F. M. Siddiqui, M. Russell, B. Bardak, R. Woods, and K. Rafferty. "IPPro FPGA based image processing processor," in 2014 IEEE Workshop on Signal Processing Systema (SPS), 2014, pp. 1-6..

[9] Crockett, Louise, David Northcote, Craig Ramsay, Fraser Robinson, and Robert Stewart. "Exploring Zynq MPSoC: With PYNQ and Machine Learning Applications" (2019)

[10] Y.-S. Cheng, Z.-Y. Chen, and P.-C. Chang, "An H.264 spatio-temporal hierarchical fast motion estimation algorithm for high-definition video in 2009 IEEE International Symposium on Circuits and Systems, 2009, pp. 880-883.

[11] E Lee "Overview of the Ptolemy Project tech memo Mar 2001, http ptolemy on of FPGA eecs erkeley.edu.

[12] P. Alexander and C. Kong "Rosetta e scheduling Semantic Support for Model Centered Linux kernel, Systems Level Design," Computer, vol. 34. computational no. 11, Nov. 2001. pp. 64-70