2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT) | 979-8-3503-0356-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICT59525.2023.10313150

Performance Evaluation of Real-Time Video Processing Edge Detection on Various Platforms

Guner TATAR
Dept. of Electrical-Electronics Eng.,
Fatih Sultan Mehmet Vakıf University
Istanbul, Turkey
Orcid: 0000-0002-3664-1366

Salih BAYAR
Dept. of Electrical-Electronics Eng.,
Marmara University
Istanbul, Turkey
Orcid: 0000-0002-4600-1880

Ihsan CICEK
Dept. of Electronics Eng.,
Gebze Technical University
Kocaeli, Turkey
Orcid: 0000-0002-7881-1263

***Abstract*—As real-time video processing applications grow in complexity, they demand higher performance. Achieving such a performance must involve a delicate balance between design constraints and optimization of performance criteria. A vital aspect of this balance is the integration of application-specific accelerator designs to boost computational efficiency. To illustrate this, we applied Laplacian High-Pass filtering operations on real-time video signals across three hardware platforms an ARM processor, an ARM+FPGA-based SoC, and a single-core Intel i7 processor. We further analyzed these platforms' price-performance ratios. Our research revealed that the ARM+FPGA-based SoC executed the filtering algorithms 23.124 times faster than the ARM processor and 1.969 times faster than the Intel i7 processor. Additionally, the ARM+FPGA-based SoC also showed the highest price-performance efficiency. To offer readers a more visual understanding, we include a resource utilization graph for the SoC hardware accelerator development board, thus demonstrating the efficiency of each platform tested.**

***Index Terms*—Video processing, OpenCV, PYNQ-Z1 SoC, FPGA Vision, Overlay design, Pipeline architecture, Hardware accelerator**

I. Introduction

Real-time video processing, prevalent in numerous sectors such as surveillance, traffic control, and medical imaging, necessitates high computational power and speed. The progression in semiconductor technology has created digital circuits that can function at these high speeds. This advancement is particularly evident in digital image and video processors. Digital filtering techniques are favoured in current applications because they offer high performance and can be adjusted while the system operates. To prevent delays caused by filtering in high-speed applications, utilising both software enhancements and hardware accelerators is crucial. Hardware accelerators usually function as peripheral interfaces, alleviating the processing load from the connected processor by taking on heavy calculations. These accelerators can take many forms, including multi-core central processing unit (CPU) architectures, graphic processing units (GPUs), application-specific integrated circuits (ASICs), and field programmable gate arrays (FPGAs). Given cost, performance, energy usage, and design flexibility [1], [2], FPGAs, which can be programmed to serve as hardware accelerators, are extensively employed in digital video processing applications [2]–[5].

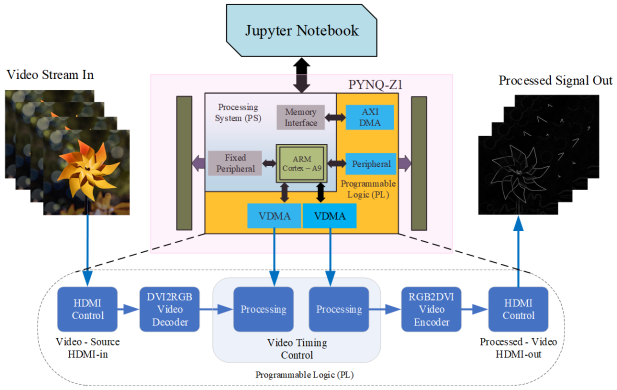

Fig. 1. Real-time video processing block diagram

In this research, we utilized FPGAs for their superior performance in executing real-time video edge detection through Laplacian high-pass filtering. The general framework of the study is illustrated in Figure 1. Edge detection is a critical process in feature extraction for various video and image processing applications. It identifies the object's outline in an image and the borders between the objects and the image background. It minimizes the low-frequency components of an image while emphasizing its high-frequency elements. Edge detection primarily relies on a gradient operation within the matrix domain, which gauges the variation between two distinct pixels. Consequently, it entails intense mathematical computations. Our research shows that combining hardware accelerator platforms with software optimizations yields beneficial results. In this study, our objective is to showcase the effectiveness of hardware accelerators when utilized in conjunction with software on different platforms.

The rest of this paper is structured as follows: Section II offers an overview of Laplacian high-pass filtering and its practical application. Section III delves into high-performance FPGA-Vision processing. Section IV provides a review of related scholarly literature. Section V proposes a hardware design, while Section VI examines the corresponding software design. Section VII focuses on the steps involved in testing and

validating our methods. Section VIII wraps up with the results and discussions. Finally IX prospects conclusions drawn from our research and discusses potential avenues for future work.

II. Laplacian High-Pass Filtering

The Laplacian of an image serves as a representation of a second-order or second-derivative enhancement method [7]. This approach is specifically designed to detect minute details in images. A Laplacian operator will accentuate any sharp, abrupt feature. Commonly used in image processing, the Laplacian is a differential operator that approximates the second derivative, as given by Eq 1. Thus, it offers the Laplacian F(x,y) value for an image with pixel density values denoted by I(x,y). In simple terms, it helps enhance an image's features by calculating the sudden changes in pixel values, which often represent the edges or other essential details in an image.

$$F(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} \tag{1}$$

In technical terms, our primary focus is on grey-scale images, which involves examining changes in the density of the image's pixels. A sudden increase or 'spike' in pixel density typically indicates the presence of an edge. Mathematically, this situation can be represented by taking the derivative of pixel intensity; a rise in intensity signifies an edge. The Laplacian filter identifies edges in both x and y dimensions, indicating that we are essentially taking the second derivative of the pixel densities. As shown in Fig 3, the graph hits zero when we take this second derivative. Consequently, our approach is to search for zero-pixel points in the image, marking these as edge points. Simply put, this method enables us to highlight significant features in an image at the edges.

In order to utilize the Laplacian operator in digital signal processing, it is necessary to interpret the equation in discrete time since real-world digital signals are represented as a series of discrete values over time. Given that the Laplacian operator is a two-variable partial differentiation operation, we will first represent it as a single-variable function and then extend it to a two-variable formulation. Like Eq 1, the Laplacian operator for a single variable continuous function can be expressed as shown in Eq 2. We are reformatting the Laplacian equation to suit the discrete nature of digital signals, enabling its practical application in digital signal processing.

$$\nabla f(x) = \frac{\partial^2 f(x)}{\partial x^2} \tag{2}$$

We should utilize approximate discretization, as shown in Eq 3, to represent it in discrete time.

$$Lf(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \tag{3}$$

Where, L is the discrete Laplacian operator, while h is a constant. This discretization implies that the function $f(x)$ is sampled evenly at each distance h. As shown in Eq 4, we may apply this approach as software or hardware convolutionally.

$$Lf(x) = l * f \tag{4}$$

In this case, l is a filter, and the Laplacian in (4) may be expressed as a matrix operator depending on the assumed boundary condition.

Laplacian filter kernels typically display a distinct pattern: they have negative values along a cross shape centred within the array (resembling a plus sign), while the corners of the array feature either zero or positive values. The value at the very centre of this array can be either negative or positive. When it comes to a five-by-five kernel for a Laplacian filter, it has a specific layout, as shown in Fig. 4. Simply put, a kernel is a small matrix utilized in processes like blurring, sharpening, edge detection, end so on. The pattern of negative and positive values within the Laplacian kernel allows for highlighting rapid intensity changes in images indicative of edges or detailed features.

III. High Performance FPGA-Vision Processing

Embedded vision technology, still in its developmental stages, heavily relies on FPGA and GPU processors integrated into embedded systems [6]. In recent years, FPGA architectures have gained prominence over GPUs and CPUs for embedded vision applications [8]. Although ASICs perform well in power and speed, their high production costs limit their use. FPGAs excel in speed, have low latency, consume less energy than GPUs, and possess inherent parallel processing capabilities. This has led to FPGAs being favoured for embedded vision, establishing a unique niche known as 'FPGA-vision' distinct from the broader computer vision landscape.

The growth of FPGA-vision technology, especially with the introduction of ZYNQ architectures, has been remarkable. ZYNQ integrates ARM-based processor programmability with FPGA hardware programmability, as shown in Fig. 5. ZYNQ architectures are also compatible with high-level languages like C/C++ and Python. While hardware description languages (HDL) are familiar with FPGAs, they can slow design time. High-level synthesis approaches are increasingly needed to overcome this bottleneck. High-level languages have been introduced into hardware design and synthesis to address this. In our study, we applied this by creating a Laplacian high-pass filter design on the ZYNQ-7020 SoC for real-time video edge detection using the PYNQ-Z1 development board. We tested our approach on different platforms: the SoC (hardware-assisted software), dual-core ARM Cortex-A9 (software only), and a personal computer (also software only) to evaluate effectiveness and system response speed.

IV. Related works

Extensive research and numerous academic publications focus on digital video processing. This field demands high computational density, necessitating hardware architectures

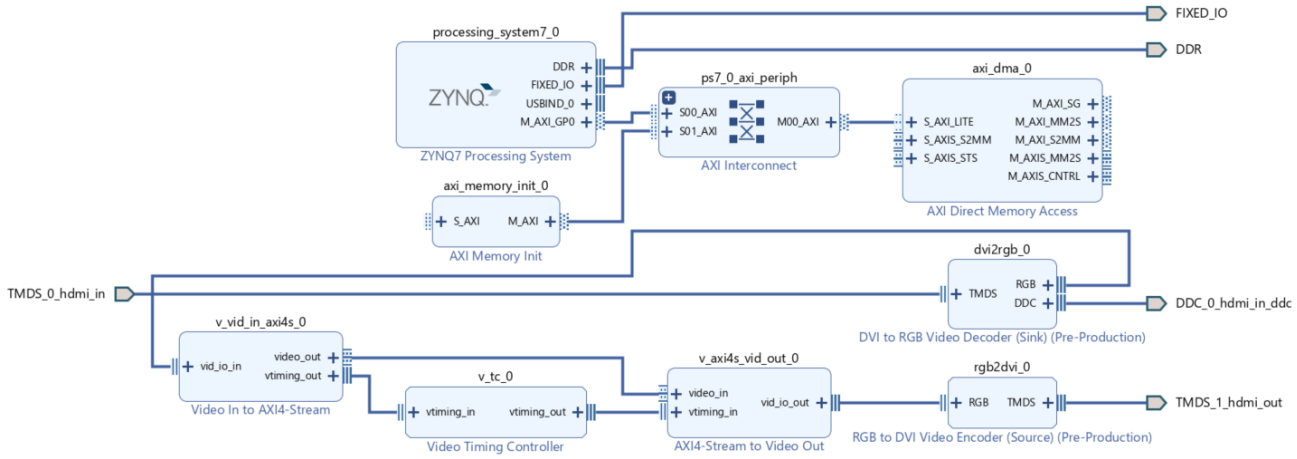

Fig. 2. Basic interface view with the pipeline of computer vision overlay

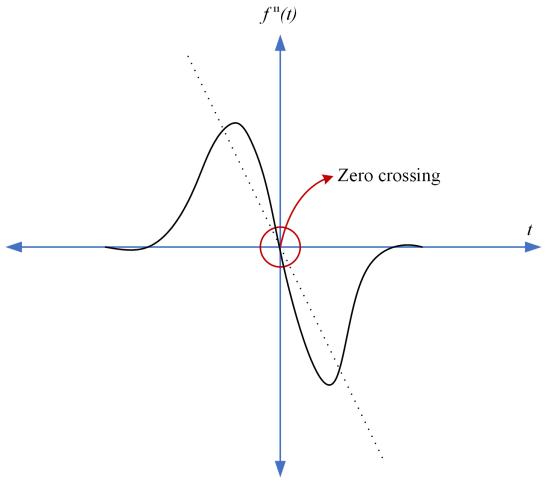

Fig. 3. Reference graph of the pixels which goes to zero

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Fig. 4. A 5 by 5 kernel for a Laplacian filter

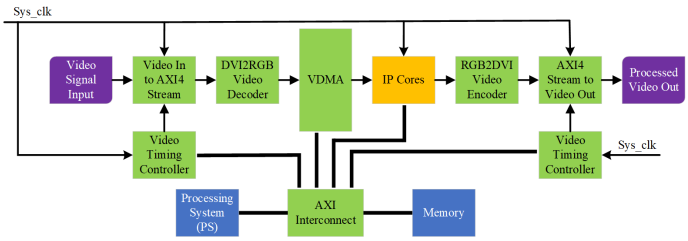

Fig. 5. Simplified flow chart of video streaming steps

offering high speed and low power consumption. As a result, FPGA, GPU, and DSP architectures are commonly preferred in these studies. Conversely, ASICs are less popular due to their expensive production costs. For instance, Daniel C.K. Kho et al. utilized a Xilinx Artix 7 (XC7A200T) FPGA in their study [9], achieving nearly 60 frames per second (FPS) processing speed in full HD video processing. They employed a 3x3 kernel for video image processing and attributed their success to the FPGA's ability to directly output within kernel data structures, providing efficient access to algorithm blocks.

In contrast to the referenced study, our research employed a different development board, achieving 45 FPS using a 5x5 kernel size. Another study focused on a pipeline architecture and a CNN-based FPGA-assisted video transcoding system [10]. This system converted 2K to 4K video, with their pipeline design and SoC-based FPGA crucial for determining the video stream's frame rate. Their approach enabled 10 FPS processing of ultra-high-definition video through hardware acceleration and software optimization.

Additionally, another study introduced a noise reduction filter for videos [11]. Executed on a Xilinx ZedBoard (ZYNQ-7020), the authors reported a speed of approximately 40 FPS for a video resolution 720x576. In a study referenced in [12], the authors executed a different type of filter application on a platform based on the Xilinx ZYNQ-7020 FPGA. During an edge detection application, they reported reaching 30 FPS for a video image with a 1280x720 resolution. We used a similar platform in our study and achieved approximately 44 FPS. Unlike the referenced study, we incorporated a pipeline architecture in our application and modified the threshold value and kernel size within the software.

Additionally, the authors of a study in [13] suggested a Sobel edge detection design application for video surveillance. Their research claimed to have achieved a refresh rate of around 24 FPS for a 4K resolution image. They documented

the application's resource usage and advocated for the Xilinx ZC702 SoC as the preferred hardware. We examined the resource usage of this study, among others, and provided a comparative analysis of our study, as presented in Tab II.

V. Hardware Design

The ZYNQ architecture is a programmable System-on-Chip (SoC) that pairs a dual-core ARM Cortex-A9 processing system (PS) with a conventional programmable logic (PL) integrated circuit (IC). This combination is characterized by high-bandwidth and low-latency connections, utilizing the advanced extensible interface (AXI) standard [15]. In the design process, tasks can be allocated between the processor and FPGA sections, termed PS and PL, respectively, according to the system's needs. The importance of this stage is highlighted by its impact on both the speed and functionality of the program being executed and the overall system performance. These outcomes are directly influenced by the division of tasks between the PS and PL. We assigned components that required high speed and computational density for video processing to the PL while the PS handled other parts. We defined the functional blocks in the hardware design and assembled them as Intellectual Properties (IPs), creating a connection between PL and PS using suitable AXI-DMA interfaces. We employed the Vivado 2020.1 integrated development environment for digital hardware development and high-level synthesis integrated system design in the PL.

The hardware accelerator design created within the PL environment is the overlay composed of IP blocks. The PYNQ-Z1 development board, designed specifically for FPGA-vision applications, has HDMI-in and HDMI-out interfaces directly connected to PL pins. Thus, the pipeline architecture (referenced in Fig.2), comprised of these IP blocks, serves as a hardware accelerator for the video processing application.

Digilent offers some IP cores, compatible with Xilinx Vivado IP, free of charge [16]. However, as these IP cores are not built into Vivado, the designer must manually integrate them. The pipeline architecture was constructed and incorporated into the machine-vision overlay following these integrations. This facilitated the acceleration and parallel execution of the video stream. The top layer of the pipeline design includes a 100MHz system clock, video decoder/encoder, video timing controller, high-performance AXI stream, and other supplementary IP blocks.

The FPGA portion is preferred for matrix operations in convolution due to the pipeline's ability to complete tasks quickly. Given the use of a 5x5 kernel in this study, the convolution process has a high computational density. The calculation time is longer since the image is in Full HD. Consequently, the computationally intense portion was executed on the PL side. The convolution operations are crucial for video processing and are executed rapidly in PL and transferred to the output via AXI-DMA. The time elapsed during the convolution process and data transfer represent the processing time of the PL.

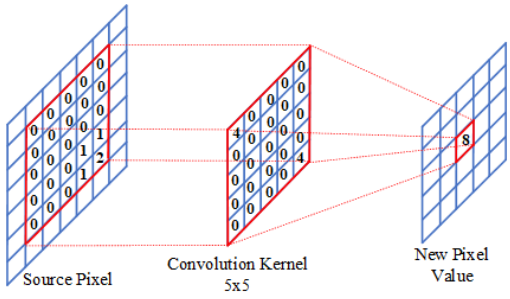

Fig. 6. Convolution operation with 5x5 kernel

VI. Software Design

The PYNQ-Z1 System on a Chip (SoC) development board, which we utilized in our study, was programmed using Python, a high-level language. The PYNQ-Z1 directly connects to a Jupyter Notebook via a static IP address. It is essential to clarify that Python coding is not applied to the IP blocks created for hardware description and verification. Instead, it is used exclusively on the Processor System (PS) side. Python facilitates the interaction between the PS and PL via interfaces.

For the video processing accelerator in the PL, a frame buffer stream architecture is defined using Python. The video pixel information is initially buffered and processed at the necessary bandwidth before being transferred by the processor to the video processing pipeline architecture using AXI interconnects. The connection between the ZYNQ PS and the AXI stream is established with Direct Memory Access (DMA). At the outset, the maximum DMA transaction size is 14 bits, or 16KB. For more extensive DMA operations, the bit width must be increased (up to 26 bits for ZYNQ-7000) during the design of Vivado IP.

In order to compare performances, we employed three different software algorithms: standalone ARM (PS), a combination of FPGA and ARM (PL+PS), and a multi-core CPU. Fig. 5 provides a flow chart of the operations carried out on both the PS and PL sides. The specifications for the multi-core CPU are provided in Tab. I. However, it is worth mentioning that, despite the computer having four cores, only one is used. Software parallelization techniques such as Pthreads, MPI, or OpenMP are not implemented.

VII. Test and Evaluation

The PYNQ-Z1 platform is built upon the ZYNQ-7000 SoC architecture, an integrated circuit that unites an ARM processor with an FPGA. This design allows for separate or hybrid usage of the FPGA and ARM processor. For our study, we evaluated the performance of the ARM processor alone,

TABLE I
COMPARISON OF THE 5X5 KERNEL LAPLACIAN HIGH-PASS FILTER'S REAL-TIME VIDEO PROCESSING EDGE DETECTION PERFORMANCE ON VARIOUS HARDWARE PLATFORMS

Platform	Embedding Methods	Hardware Architecture	FPS Value	Acceleration Rate	Cost ($)	Price / Performance ($ /×)
CORA Z7	Software	ARM Cortex-A9 Dual-Core 667 MHz	1.912	1×	147[1]	147
PYNQ-Z1	Software + Hardware	ZYNQ - Z7020	44.214	**23.124**×	299[2]	**12.921**
Personal Computer	Software	i7 7700 HQ 16 GB RAM Quad-Core 2.86 GHz	22.456	11.744×	∼1000	85.149

[1] https://digilent.com/shop/cora-z7-zynq-7000-single-core-and-dual-core-options-for-arm-fpga-soc-development/
[2] https://digilent.com/shop/pynq-z1-python-productivity-for-zynq-7000-arm-fpga-soc/

TABLE II
POST IMPLEMENTATION OF ZYNQ ARCHITECTURE RESOURCE USAGE IN TABLE

Resources	PYNQ Z1 Available	Utilization (%)	ZYNQ ZC702 Available	Utilization [13] (%)	ZYNQ Z-7010 Available	Utilization [14] (%)
LUT	53200	2.29	53200	10.6	17600	41.48
LUTRAM	17400	0.14	17400	N/A	N/A	N/A
FF	106400	1.64	106400	7.44	35200	34.97
BRAM	140	1.43	140	5	60	20
IO	125	16	200	26	100	32
BUFG	32	15.63	N/A	N/A	N/A	N/A
MMCM	4	50	N/A	N/A	N/A	N/A
PLL	4	25	N/A	N/A	N/A	N/A

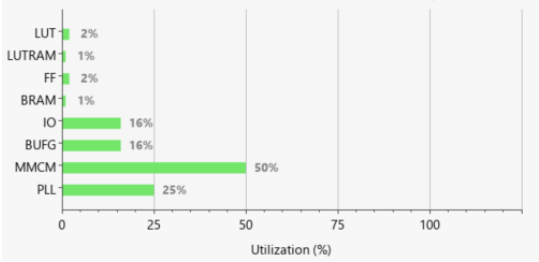

Fig. 7. Post implementation of ZYNQ architecture resource usage in graph

the hybrid ARM+FPGA, and a Multi-core CPU for a high-pass filtering sample problem. Our performance comparisons across these three different platforms are presented in Table I. We first conducted our experiment using only the ARM processor as a software accelerator. With this setup, we achieved an FPS value 1.912 following the convolution process using a 5x5 kernel (e.g., see Fig. 6). When we employed the ARM and FPGA architectures together under identical conditions, we saw a significant increase in the FPS value to 44,214.

Figure 8 illustrates the output from the real-time video processing application on the ZYNQ SoC (FPGA+ARM). We inverted the image to make it easier for the reader to understand and analyze. Similar results were observed on other hardware accelerator platforms, but only the best-performing platform's results are showcased here. High workload operations, such as convolution, were performed on the FPGA (PL), while the ARM (PS) processor executed other parts. The AXI interconnect provided data communication between the PS and PL. Despite AXI's high bandwidth and low latency connectivity, a certain amount of delay was observed in the data transmission and reception process. Finally, our study was conducted on a personal computer equipped with a quad-core 2.8 GHz Intel i7 processor and 16 GB of RAM to exhibit its performance in a CPU environment. Here, we measured an FPS value of 22.456. The ZYNQ 7020 SoC environment, which utilizes both software and hardware, outperformed the other environments regarding price and performance, as seen from Table I.

VIII. RESULTS AND DISCUSSION

Video processing applications can involve tasks like edge detection, object recognition, motion tracking, and more. These tasks require analyzing and manipulating large amounts of data in real-time. The increasing complexity of video processing tasks directly amplifies the computational load due to the involvement of intricate algorithms and calculations, necessitating greater processing power and memory resources.

Many video processing applications, including surveillance, medical imaging, and autonomous vehicles, operate under real-time constraints, demanding rapid frame processing for timely decisions; tasks with higher complexity may require enhanced processing capability to meet these demands. Such complex tasks further demand heightened accuracy and precision, particularly in tasks like fine-grained edge detection and object recognition, driving the need for increased computational resources to attain precise outcomes. Furthermore, energy efficiency is impacted by complex

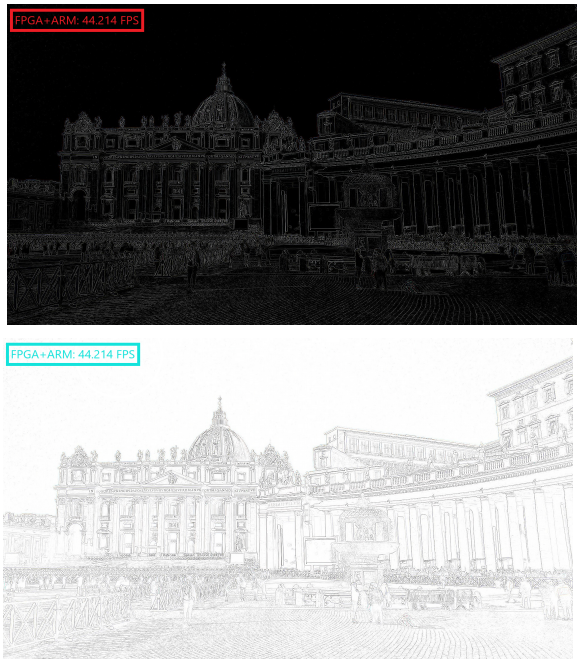

Fig. 8. ARM+FPGA-based accelerator for a Laplacian high-pass filter a) filtered image, b) inverse image

video processing, as heightened computations correlate with increased power consumption—a critical concern for battery-constrained mobile and edge devices.

In our research, we implemented real-time video processing across different hardware platforms and compared their performance. The detailed results are outlined in Table I. The selection of hardware for any project should be driven by speed, cost, and performance requirements. The ZYNQ-7000 integrated circuit offers the flexibility to use ARM and FPGA architectures separately or in combination. Any code intended for the ARM processor needs to be software-based and requires optimization for acceleration. On the other hand, the FPGA architecture provides inherent parallel processing capability, which enables simultaneous and faster execution of the code, hence the term ”hardware accelerators.” Although general-purpose CPUs can operate at high frequencies, they often need to catch up in applications requiring high computational density since they serially allow code execution. The performance of hardware accelerators can vary depending on the type of application. When considering FPS and cost, our results demonstrated that the optimal setup involves a combination of FPGA and ARM. The resource usage required for this FPGA+ARM combination is depicted in Fig. 7 and Table II. These findings can guide designers in their choice of hardware, ensuring a balance between performance and cost.

IX. Future work

Edge detection in a frame forms the foundation of an image or video processing applications. The rapid advancements in autonomous vehicle technology today are mainly due to sophisticated image-processing techniques. Identifying the edges within an image is an essential first step. This endeavour is the launching point for our future research. Our goal is to enhance the functionality of the A/DAS tasks for both driver-operated and autonomous vehicles by applying our real-time video processing application.

References

[1] D. -M. Ngo, A. Temko, C. C. Murphy and E. Popovici, ”FPGA Hardware Acceleration Framework for Anomaly-based Intrusion Detection System in IoT,” 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 2021, pp. 69-75, doi: 10.1109/FPL53798.2021.00020.

[2] A. Kübra ERENOĞLU and G. TATAR, ”Real-Time Hardware Acceleration of Low Precision Quantized Custom Neural Network Model on ZYNQ SoC,” 2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Istanbul, Turkiye, 2023, pp. 1-6, doi: 10.1109/HORA58378.2023.10155783.

[3] G. Tatar, S. Bayar, and I. Cicek, “FPGA Design of a High-Resolution FIR Band-Pass Filter by Using LabVIEW Environment,” European Journal of Science and Technology Special Issue, vol. 29, no. 29, pp. 273–277, 2021, doi: https://doi.org/10.31590/ejosat.1016363).

[4] G. Tatar, I. Cicek, and S. Bayar, “FPGA Design of a Fourth Order Elliptic Band-Pass Filter Using LabVIEW,” European Journal of Science and Technology Special Issue, vol. 26, no. 26, pp. 122–127, 2021, doi: https://doi.org/10.31590/ejosat.951601).

[5] C. Soubervielle-Montalvo et al., “Design of a Low-Power Embedded System Based on a SoC-FPGA and the Honeybee Search Algorithm for Real-Time Video Tracking,” Sensors, vol. 22, no. 3, p. 1280, Feb. 2022, doi: 10.3390/s22031280.

[6] M. Qasaimeh et al., “Benchmarking vision kernels and neural network inference accelerators on embedded platforms,” Journal of Systems Architecture, vol. 113, p. 101896, Feb. 2021, doi: https://doi.org/10.1016/j.sysarc.2020.101896.

[7] S. S. Bhairannawar, “Chapter 4 - Efficient Medical Image Enhancement Technique Using Transform HSV Space and Adaptive Histogram Equalization,” ScienceDirect, Jan. 01, 2018. https://www.sciencedirect.com/science/article/abs/pii/B9780128130872000038 (accessed Sep. 15, 2023).

[8] X. Jiang, “Human tracking of track and field athletes based on FPGA and computer vision,” Microprocessors and Microsystems, vol. 83, p. 104020, Jun. 2021, doi: https://doi.org/10.1016/j.micpro.2021.104020.

[9] D. C. K. Kho, M. F. A. Fauzi and S. L. Lim, ”Hardware Parallel Processing of 3×3-pixel Image Kernels,” 2020 IEEE REGION 10 CONFERENCE (TENCON), Osaka, Japan, 2020, pp. 1272-1276, doi: 10.1109/TENCON50793.2020.9293914.

[10] Y. Wei, L. Chen, R. Xie, L. Song, X. Zhang and Z. Gao, ”FPGA Based Video Transcoding System with 2K-4K Super-Resolution Conversion,” 2019 IEEE Visual Communications and Image Processing (VCIP), Sydney, NSW, Australia, 2019, pp. 1-2, doi: 10.1109/VCIP47243.2019.8966063.

[11] Madhura Shivaraju and Suresh Krishnappa, “A new parallel DSP hardware compatible algorithm for noise reduction and contrast enhancement in video sequence using Zynq-7020,” International Journal of Computer Aided Engineering and Technology, vol. 13, no. 1/2, pp. 14–14, Jan. 2020, doi: https://doi.org/10.1504/ijcaet.2020.108101.

[12] X. Wei et al., ”FPGA Implementation of Hardware Accelerator for Real-time Video Image Edge Detection,” 2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2021, pp. 16-20, doi: 10.1109/ASID52932.2021.9651710.

[13] S. Eetha, S. Agrawal and S. Neelam, ”Zynq FPGA Based System Design for Video Surveillance with Sobel Edge Detection,” 2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Hyderabad, India, 2018, pp. 76-79, doi: 10.1109/iSES.2018.00025.

[14] A. Ben Amara, E. Pissaloux and M. Atri, ”Sobel edge detection system design and integration on an FPGA based HD video streaming architecture,” 2016 11th International Design & Test Symposium (IDT), Hammamet, Tunisia, 2016, pp. 160-164, doi: 10.1109/IDT.2016.7843033.

[15] “Zynq-7000 SoC,” Xilinx. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[16] “Vivado Library - Digilent Reference,” digilent.com. https://digilent.com/reference/vivado:library (accessed Sep. 15, 2023).