Design and Simulation of a Median Filter for a CubeSat Image Processing Application Using an FPGA Architecture

Mohammed Alae Chanoui[1,2*], *Issam* Bouganssa[1], *Mohammed* Sbihi[1,2], *Zine Elabidine* Alaoui Ismaili[2,3], and *Adil* Salbi[1]

[1]LASTIMI Laboratory, EST Salé, Mohammed V University in Rabat, Morocco
[2]CURTS, EMI, Mohammed V University in Rabat, Morocco
[3]ICES Team, ENSIAS, Mohammed V University in Rabat, Morocco

Abstract. CubeSats are small satellites that can perform space missions with the advantage of low cost and short development time. Earth observation is a well-known satellite use case that has found its place in the CubeSat community. To improve the quality and the number of images that can be received from the satellite, image processing techniques can be performed. Satellite images can be disturbed, and the median filter is a pre-processing technique usually used to remove impulse noise. The aim is to develop an architecture for CubeSat onboard image processing, starting with the design of a median filter. This paper presents the design and the simulation process of a 3x3 median filter based on the Spartan 6 FPGA architecture using software components. Simulation results are generated using a test bench algorithm and a visual comparison of both the input and output images is performed.

Keywords: CubeSat, image processing, median filter, FPGA.

1 Introduction

CubeSats are small satellites that regroup all major satellite parts while using COTS components. They have the advantage of being able to perform space applications in low earth orbit at a low cost and in a short development time. The CubeSat standard was created by Professor Jordi Puig-Suari at Cal Poly and Professor Bob Twiggs at Stanford. The standard specifies that a 1-unit (1U) CubeSat has a size of 10x10x10 cm^3 and a maximum weight of 1.33 kg [1]. The purpose of the standard is to provide design specifications for CubeSats designers, allowing launch vehicle manufacturers to use a common deployment system for all CubeSats.

A 1U CubeSat could either serve as a standalone satellite or could be combined with other units to build a larger spacecraft. For instance, a 3U CubeSat will have a form factor similar to three combined 1U CubeSats. The choice of the satellite dimensions depends on the mission that is conducted and its power budget.

Earth observation is one of the main satellite use cases. This can be supported by the fact that hundreds of earth observation satellites have been launched and provide useful measurements for all the disciplines of the earth sciences: hydrology, climatology, meteorology, aeronomy, atmospheric chemistry, oceanography, geology, biology, and so on [1]. Although CubeSats are developed for educational purposes or capability demonstrations, earth observation research can be conducted using a small spacecraft.

The amount of data that can be stored in a CubeSat is limited by its hardware specifications. Given that the spacecraft communicates with the ground station only a few times per day, the amount of data that can be transmitted to the user has to be optimized. Onboard image processing is a potential solution to achieve that. The aim is to perform onboard image processing and store only the relevant results instead of the original images, leaving space for more data to be stored.

When a satellite image is captured using a defective sensor or transmitted through a faulty channel, the result can be corrupted by salt and pepper noise. Applying processing algorithms to noisy images would give nonreliable results. Thus, image pre-processing is an important step, and the median filter is well-known for its good performance on random noise, such as salt and pepper.

2 Related theory and fundamentals

2.1 CubeSat architecture

To conduct a satellite mission, two segments are needed: the space segment, which is the satellite that will be in orbit for data acquisition; and the ground segment or the facilities accompanying the satellite during the mission.

The satellite architecture is a combination of two main parts: the structure, which contains the technologies that ensure the satellite's functions; and the payload, which regroups the tools needed for the experimentation.

The satellite structure is a combination of several subsystems, as shown in Fig.1.

* Corresponding author: chanouialae@gmail.com

- **OBC (On-Board Computer)** is the main part of the satellite. It provides the hardware and software platform needed to control and handle the data [2].
- **COM (Communication System)** is in charge of the communication with the ground station so that data can be sent to earth and orders can be received.
- **EPS (Electrical Power Supply)** conditions and distributes the power to all satellite subsystems. It is designed to generate power from the solar cells and store it in the batteries in order to deliver power during the eclipse and demand peaks.
- **ADCS (Attitude Determination and Control System)** has a support role through the CubeSat mission. It has to maintain all other modules in an operational situation by controlling the satellite attitude.

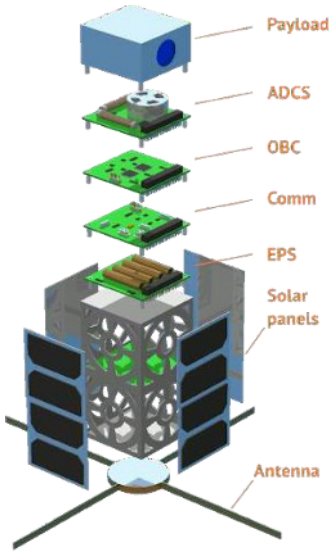

Fig. 1. Example of a CubeSat architecture.

The choice of the payload hardware and software depends on the conducted mission. For instance, in earth observation and onboard image processing, the payload hardware can be a single or multiple cameras combined with a processing unit. The processing unit can either be the satellite OBC or a payload dedicated controller that is in charge of image capture and processing.

2.2 Median filter

The median filter is a nonlinear image processing technique used to remove impulsive noise from images.

As illustrated in Fig.2, this spatial filtering operation applies a two-dimensional (2D) window mask to an image region, replacing the center pixel value with the median value of the pixels contained inside the window. After that, the window moves to the next image section, and the procedure continues until the entire image is processed. As a result, the ideal filtered image would have no impulsive noise and perfectly sharp edges [3].

2.3 Salt and pepper noise

Salt and pepper noise indicates an image with pixels resembling salt and pepper. These noisy pixels have values that are either near the maximum or minimum values that a pixel can take. The noise can be reduced by removing the maximum and minimum values and replacing them with a determined median value using a sorting approach [4].

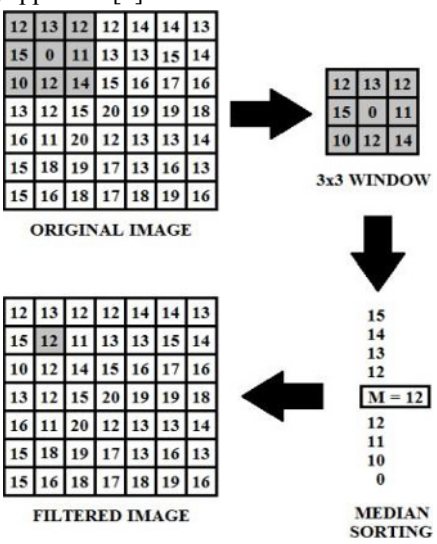

Fig. 2. Median filter process [3].

3 Materiel and methods

Based on benchmarks comparing several processing units in image processing applications, it is usually reported that FPGA gives good results in terms of energy and performance [5-7]. Also, given the fact that many image processing applications are conducted using FPGAs [8-11], it was the chosen technology for this system design.

3.1 System architecture

The designed architecture for this system is presented in Fig.3. It regroups the blocks that have been developed and those that are yet to be developed.

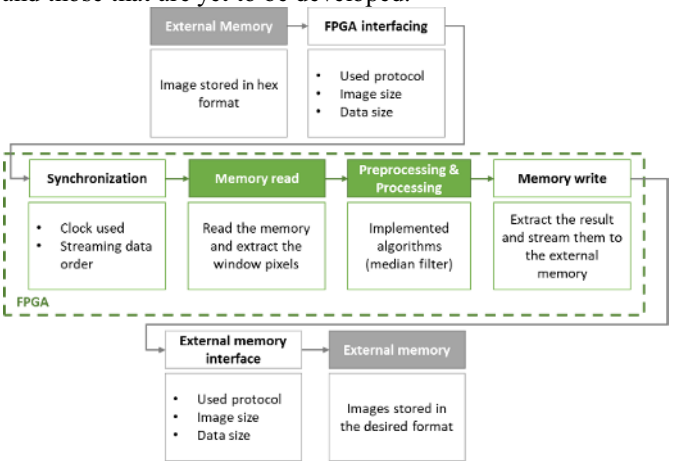

Fig. 3. System architecture.

The idea of the system is to have the image that requires processing stored in an external memory. The FPGA should retrieve the image pixels from the external memory, organize the pixels to generate the 3x3 window, apply the median filter algorithm, and then store the result in the external memory. This paper presents the work done for the FPGA design part.

To simulate the median filter results, a simulation-adapted architecture was designed as shown in Fig.4.

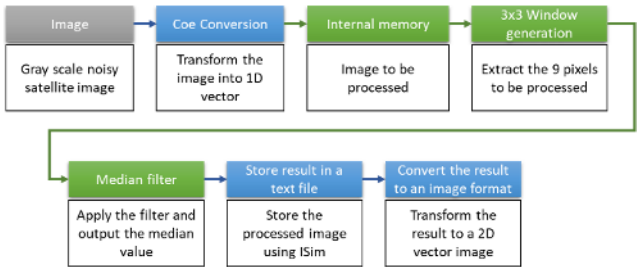

Fig. 4. Simulation architecture.

In this alternative architecture, the FPGA's internal memory is used to store the noisy image that needs processing. The image stored in the FPGA should be in a .coe file format, which regroups pixels values organized as a 1D vector, and in this case, the conversion was done using MATLAB. The "3x3 Window generation" block aims to read the image pixels from the internal memory and extract the window pixels. The "Median filter" block processes the 3x3 window and outputs the median value. The "Store result in a text file" block stores the median filter result in a text file, then the text file is converted to a 2D image and displayed using MATLAB.

3.2 Window generation

As shown in Fig.5, the window generator is made of a chain of nine flip-flops and two FIFO memories. The image pixels values are streamed through the chain, the nine pixels of the 3x3 window are stored in the flip-flop, and the other pixels of the processed row are stored in the FIFO memory. This block is designed for 200x200 images, so the FIFO memories are configured to store 197 pixels each.

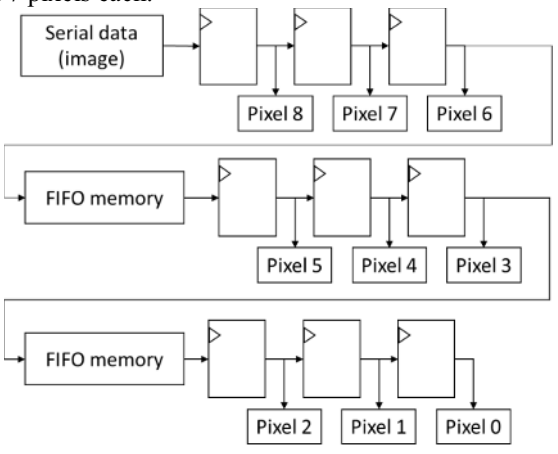

Fig. 5. Window generator architecture.

A "synchronization" block is used to synchronize the memory reading and the data streaming orders in the window generator. The block outputs the addresses of the pixels to the internal memory and enables read and write commands for the FIFO memories, as shown in Fig.6.

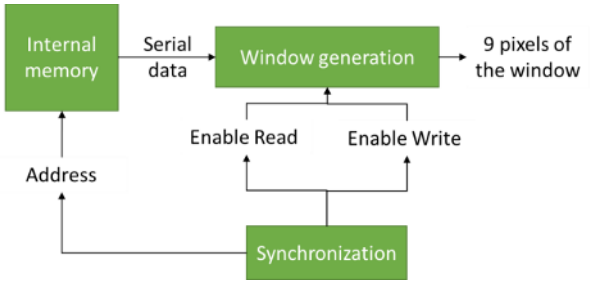

Fig. 6. Window generator synchronization

3.3 Median filter design

The designed filter works by comparing every two adjacent pixels using logic blocks at each processing step, as shown in Fig.7, so that the minimum value is buffered to the first position and the maximum value is buffered to the last position.

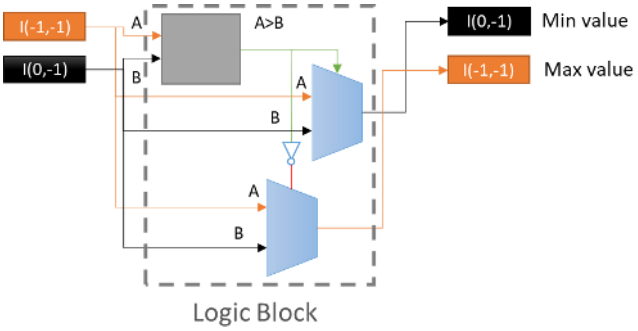

Fig. 7. Logic Block structure

Fig.8 presents the previous concept but applied to the nine pixels of the window. Four comparisons are performed at each sorting stage. In the first stage, eight pixels are being processed, and the last one is buffered to the second stage. In the second stage, the comparison is shifted so that the last pixel will be processed and the first pixel will be buffered to the third stage. This process is repeated until the median value reaches the middle pixel.

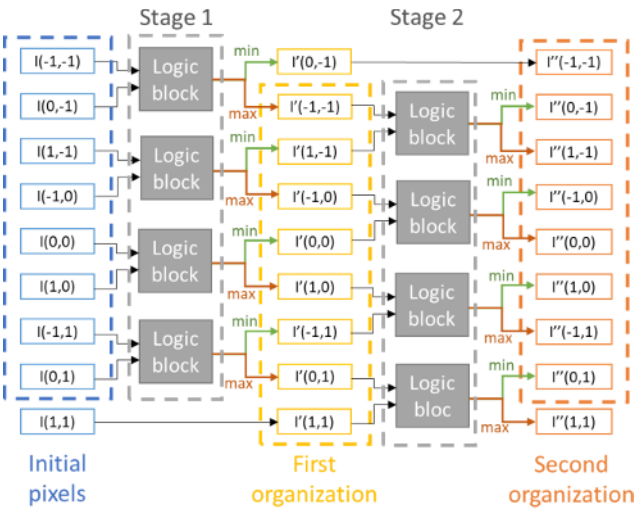

Fig. 8. Median filter design method

3.4 Hardware

An FPGA consists of a matrix of programmable logic cells with a grid of interconnecting lines and switches between them. I/O cells exist around the perimeter, providing an interface between the interconnect lines and the chip's external pins. Programming (Configuring) an FPGA consists of specifying the logic function of each cell and the switches in the interconnecting lines [12].

The design and simulation were done in the ISE Design Suite 14.7 software using a Spartan 6 FPGA architecture.

3.5 Software

In order to implement the design and do the simulation, two software were used:

- **MATLAB:** to convert the image into a standardized .coe file format so that the image can be stored in the FPGA board. After the processing, the software is used again to convert the text file that contains the median filtered image into a 2D image file.
- **ISE Design Suite 14.7:** the IDE that is used in the system design and simulations for the Xilinx FPGA.

4 Result and discussion

This part presents the results obtained during the design and the simulation.

The hardware design was done using the VHDL programming language, the simulation was done using a test bench algorithm, and the image output was displayed using MATLAB.

4.1 Architecture result

Fig.9 presents the synthesized architecture that regroups all the developed FPGA blocks for the median filter. The design is based on two main blocks: The first one is "Window3x3 Generation" as explained in section 3.2 and its RTL is shown in Fig.10, and the second one is "MedianFilter" as explained in section 3.3 and shown in Fig.11.

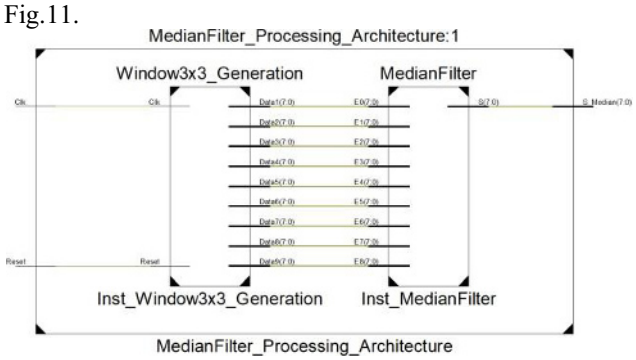

Fig. 9. RTL Schematic for the complete architecture

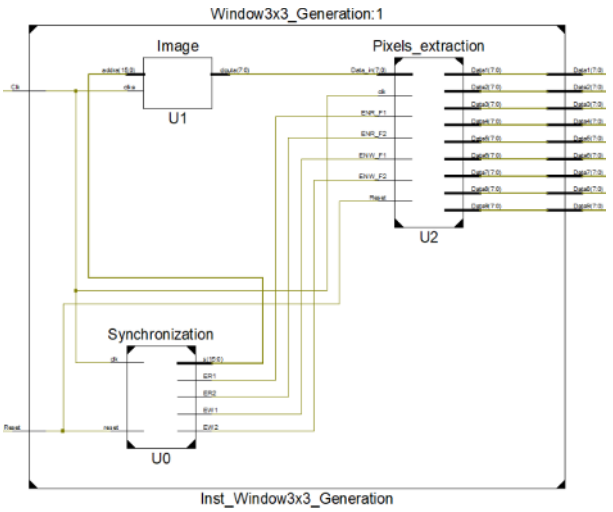

Fig. 10. RTL Schematic for the window generator

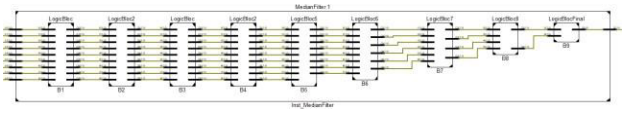

Fig. 11. RTL Schematic for the median filter

4.2 Synthesis report

A report was generated after the design was successfully synthesized. Table.1 presents a summary of the FPGA resources used, and Fig.12 gives more details about the device utilization.

Table 1. Summary report after synthesis

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	189	18224	1%
Number of Slice LUTs	672	9112	7%
Number of fully used LUT-FF pairs	66	795	8%
Number of bonded IOBs	10	232	4%
Number of Block RAM/FIFO	21	32	65%
Number of BUFG/BUFGCTRLs	1	16	6%

```
Advanced HDL Synthesis Report

Macro Statistics
# Counters                                      : 1
 16-bit up counter                             : 1
# Registers                                     : 76
 Flip-Flops                                    : 76
# Comparators                                   : 35
 16-bit comparator greater                     : 4
 16-bit comparator lessequal                   : 1
 8-bit comparator greater                      : 30
# Multiplexers                                  : 52
 8-bit 2-to-1 multiplexer                      : 52

Total REAL time to Xst completion: 12.00 secs
Total CPU time to Xst completion: 11.67 secs

-->

Total memory usage is 4570220 kilobytes
```

Fig. 12. Results from the design synthesis report

4.3 Simulation Results

The Xilinx Isim test bench tool was used to simulate the design.

In the first test, the input pixels and the output median value were displayed to check if the processing results were good, as shown in Fig.13. The simulation algorithm was designed using a 100Mhz clock, like the one in the Spartan 6 FPGA, and the result showed that the processing time needed for one 200x200 image is around 0.4ms.

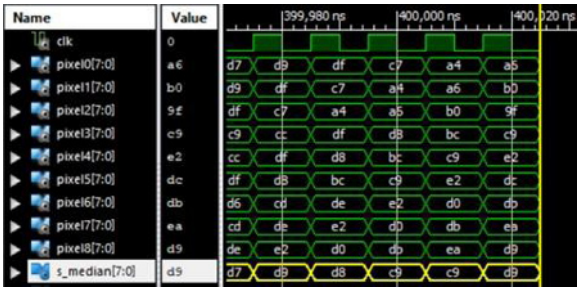

Fig. 13. Median filter test bench result

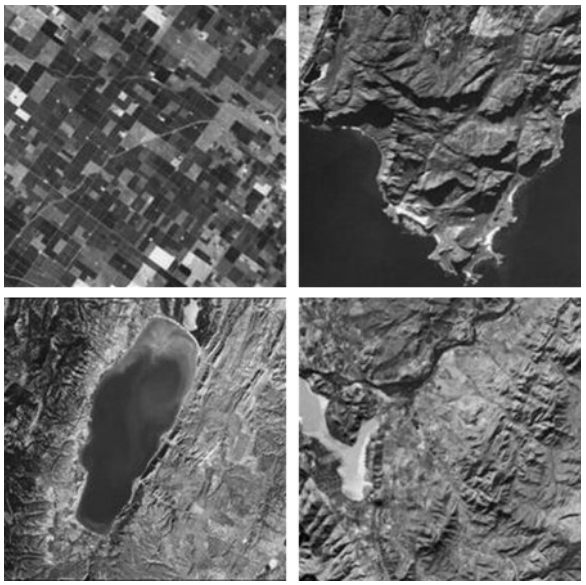

Fig. 14. Original images

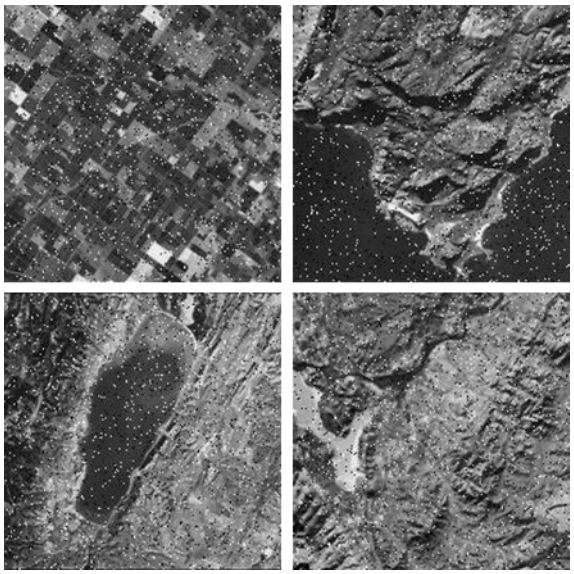

Fig. 15. Noisy images

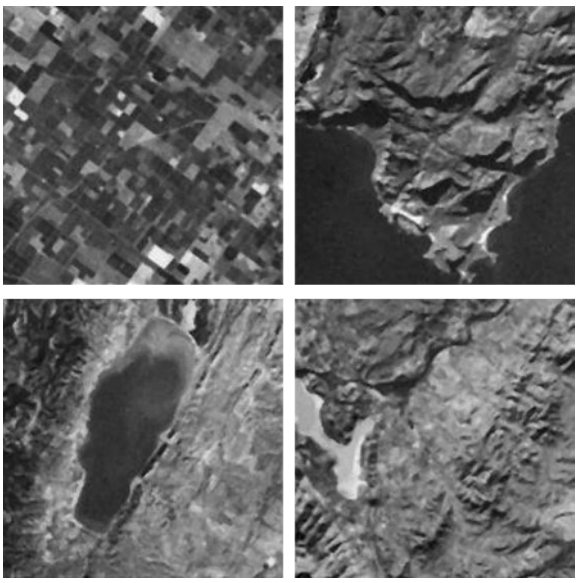

Fig. 16. Processed image

To test the efficiency of the algorithm, some CubeSat images [13] were used. The images were converted to grayscale (Fig.14), salt and pepper noise was applied to them (Fig.15), and they were input separately into the architecture. The test bench algorithm was designed to store the median filter output value in a text file. Once the text file was converted into a 2D vector, the processed images were displayed as shown in Fig.16.

After analyzing the test bench results (Fig.13) and comparing the input images (Fig.14) with the output images (Fig.16), it is noticed that the simulations gave satisfying results, so the system works well and the hardware implementation can be carried out.

5 Conclusion

This paper gives an overview of CubeSats and the proposed onboard image processing architecture. The median filter reliability was tested using images taken by a CubeSat [13], and the simulation results were satisfying.

In the future, a hardware implementation of the designed algorithm will be done using a Spartan 6 FPGA. The internal memory will be replaced by external ones so that larger images can be processed. The architecture will be optimized for better energy consumption results so that the design can be easily embedded. Once the work is complete, some advanced image segmentation algorithms will be developed.

This work is supported by Royal Center for Space Studies and Research (CRERS), National Center for Scientific and Technical Research (CNRST), Mohammed V University in Rabat (UM5) and University Center for Research in Space Technologies (CURTS).

References

1. D. Selva and D. Krejci, Acta Astronaut. **74**, 50 (2012)
2. A. Hanafi, M. Karim, I. Latachi, T. Rachidi, S. Dahbi, and S. Zouggar, *FPGA-based secondary on-*

board computer system for low-earth-orbit nano-satellite, in 2017 International Conference on Advanced Technologies for Signal and Image Processing, ATSIP, (2017)

3. L. A. Aranda, P. Reviriego, and J. A. Maestro, IEEE Trans. Nucl. Sci. **64**, 2219 (2017)
4. V. P. Korakoppa, Mohana, and H. V. R. Aradhya, *Implementation of highly efficient sorting algorithm for median filtering using FPGA Spartan 6*, in 2017 International Conference on Innovative Mechanisms for Industry Applications, ICIMIA, (2017)
5. C. Brugger, L. Dal'Aqua, J. A. Varela, C. De Schryver, M. Sadri, N. Wehn, M. Klein, and M. Siegrist, *A quantitative cross-architecture study of morphological image processing on CPUs, GPUs, and FPGAs*, in 2015 IEEE Symposium on Computer Applications & Industrial Electronics, ISCAIE, (2015)
6. P. Cooke, J. Fowers, G. Brown, and G. Stitt, ACM Trans. Reconfigurable Technol. Syst. **8**, (2015)
7. M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, *Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels*, in 2019 IEEE international conference on embedded software and systems, ICESS, (2019)
8. S. M. Raje, A. Goel, S. Sharma, K. Aggarwal, D. Mantri, and T. Kumar, *Development of on board computer for a nanosatellite*, in Proceedings of the International Astronautical Congress, IAC, (2017)
9. M. I. Alali, K. M. Mhaidat, and I. A. Aljarrah, *Implementing image processing algorithms in FPGA hardware*, in 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies, AEECT, (2013)
10. I. Bouganssa, M. Sbihi, and M. Zaim, *Laplacian edge detection algorithm for road signal images and FPGA implementation,* Int. J. Mach. Learn. Comput, (2019)
11. I. Bouganssa, M. Sbihi, and M. Zaim, *Implementation of Edge Detection Digital Image Algorithm on a FPGA*, MATEC Web Conf. **75**, (2016)
12. O. Al-khaleel, A. Idries, K. Mhaidat, and I. Aljarrah, *FPGA-based features extraction unit for arabic characters*, in Proceedings of the International Conference on Information and Communication Systems, ICICS, (2013)
13. P. Mhangara, W. Mapurisa, and N. Mudau, Aerospace **7**, (2020)