Design and Implementation of a Real-Time Image Processing System Based on Sobel Edge Detection using Model-based Design Methods

Taoufik Saidani[1]*, Refka Ghodhbani[2], Mohamed Ben Ammar[3], Marouan Kouki[4], Mohammad H Algarni[5], Yahia Said[6], Amani Kachoukh[7], Amjad A. Alsuwaylimi[8], Albia Maqbool[9], Eman H. Abd-Elkawy[10]

Department of Computer Sciences-Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia[1, 2, 9, 10]

Department of Information Systems-Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia[3, 4, 7]

Department of Computer Science, Al-Baha University, Saudi Arabia[5]

Department of Electrical Engineering-College of Engineering, Northern Border University, Saudi Arabia[6]

Department of Information Technology-Faculty of Computing and Information Technology, Northern Border University, Rafha 91911, Saudi Arabia[8]

***Abstract*—Image processing and computer vision applications often use the Sobel edge detection technique in order to discover corners in input photographs. This is done in order to improve accuracy and efficiency. For the great majority of today's image processing applications, real-time implementation of image processing techniques like Sobel edge detection in hardware devices like field-programmable gate arrays (FPGAs) is required. Sobel edge detection is only one example. The use of FPGAs makes it feasible to have a quicker algorithmic throughput, which is required in order to match real-time speeds or in circumstances when it is critical to have faster data rates. The results of this study allowed for the Sobel edge detection approach to be applied in a manner that was not only speedy but also space-efficient. For the purpose of actually putting the recommended implementation into action, a one-of-a-kind high-level synthesis (HLS) design approach for intermediate data nodes that is based on application-specific bit widths was used. The high-level simulation code known as register transfer level (RTL) was generated by using the MATLAB HDL coder for HLS. The code written in hardware description language (HDL) that was produced was implemented on a Xilinx ZedBoard with the aid of the Vivado software, and it was tested in real time with the assistance of an input video stream.**

Keywords*—Image processing; sobel edge detection; high level synthesis; model based design; Zynq7000 MATLAB HDL coder

I. Introduction

The evolving requirements and technological advancements have led to a growing demand for real-time image processing systems, widely utilized across several industries. Real-time embedded system designs prioritize completing tasks within a certain timeframe overachieving high speed. These systems are commonly utilized in driving support systems, driverless vehicles, flight control, security, and military systems. Predictability characteristics and time stability are essential requirements for real-time systems. [1].

The majority of contemporary image processing and computer vision systems still struggle with the basic challenge of identifying the area of interest in a picture. This is necessary for a wide range of applications, including advanced driving assistance systems (ADAS), which identify things like pedestrians, traffic signals, and blind spots; lane departure warning systems; video surveillance applications; and simultaneous localization and mapping (SLAM) [1]. One example of this kind of feature in a picture is a corner, which is the place at where two distinct edges meet. Image corner detection often involves the employment of many algorithms. The Moravec method [2], the Susan algorithm, and the Sobel edge detector are just a few examples of the kinds of corner extraction techniques that see regular application. The Sobel edge detector is one of the corner detecting algorithms that has the highest level of accuracy. The method is quite computationally demanding, despite the fact that its operation is remarkably simple. It is frequently utilized in systems that demand data processing in real time; as a result, traditional CPUs are unable to fulfill the requirements of these systems. CPUs are only useful when there are big amounts of data involved or when we need to execute calculations using floating point numbers. As a result, field-programmable gate arrays, often known as FPGAs, are great candidates for implementing such algorithms in real time as a result of their rapid processing rates and parallel implementations. It's possible that corner detection will need to be developed on the FPGA in addition to the other algorithms if you're going to be using it for sophisticated computations. Some examples are non-maxima suppression, matrix computation, and triangulation [3]. Other examples include matching by utilizing the sum of absolute differences. As a result of this prerequisite, it is essential to enhance the effectiveness as well as the space needs of the FPGA implementations for these algorithms [4].

A large number of academics have recently released original work on innovative implementations of the Efficient implementation of Sobel edge detection on FPGAs in terms of both space and time. Liu and colleagues proposed a method capable of processing RGB565 video at 640x480 resolution with a frame rate of 154 frames per second. Liu and colleagues

implemented the idea using a Xilinx ZedBoard. Xu et al. [5] introduced a modified approach that incorporates a pre-filter and use a simplified matrix instead of the original Gaussian kernel matrix. The researchers devised this novel algorithm. As a result, the design complexity was reduced, allowing robotics applications utilizing a Spartan 3 FPGA to efficiently utilize their hardware resources. In the experiments, a 256 x 256 pixel input picture was processed in 2.3 milliseconds. Chao et al. [6] utilized the Sobel edge detector to simplify the maximum suppression technique. They achieved a data rate of 144 frames per second in their simulations with a design specifically tailored for ZedBoard. Research by Lee and colleagues [7], who created a modified Sobel edge detector, focused on breast cancer identification utilizing MRI and x-ray images. An automated method was employed for adaptive radius suppression to mitigate corner clustering. They were thus able to prevent the loss of important corners that oversuppression would have brought about. John and his colleagues devised a universal picture feature extractor technique and implemented it on a Cyclone 4 FPGA for real-time processing. They succeeded in obtaining a frame rate of 70 frames per second as a consequence. Hisham and colleagues developed a self-adaptive system on a chip for the Sobel edge detection technique using dynamic partial reconfiguration. Their solution consumed less electricity and had a little discernible effect on performance [8, 9, 10].

This article presents a real time implementation of the Sobel edge detector on a Xilinx ZedBoard and demonstrates that the implementation is better to earlier implementations in terms of performance and area utilization on the FPGA. The study also offers a real time implementation of the Sobel edge detector on a Xilinx ZedBoard. The design was created with the use of an innovative high-level design process that synthesizes the design using intermediate signal widths that are restricted based on the application (the input stimuli). The remaining portions of this document are structured as follows: In the next section, "Section II," you will be introduced to high-level synthesis. The Sobel edge detector's internal workings are broken out in detail in Section III. The technique that was employed in the suggested design is broken forth in Section IV. The results of the simulations and synthesis are reported in Section V, along with a comparison to the findings that were uncovered by other researchers. The final observations may be found in Section VI.

II. High-Level Synthesis based on Model-based Design

A technique that's gaining popularity is called high-level synthesis, or HLS, and it allows designers to continuously validate their designs at every stage of the design process while describing behaviors at high abstraction levels. Examples of HLS utilities include Vivado HLS, MATLAB HDL Coder, and various more open-source tools. These are frequently employed by researchers and digital designers to develop and run algorithms for a wide range of applications including fields like deep learning, neural networks, image processing, communications, and aerospace [11]. The code written on the skin can be simplified by a factor of eight using HLS technology. It enables the reuse of behavioral intellectual property in various projects and allows validation teams to employ abstraction-level modeling methods like transaction-level modeling [12].

Most modern chip systems utilize integrated CPUs. The microprocessors digital signal processors (DSPs), custom logic, and memory must all coexist on a single chip. In order for this to happen, the design process must include the creation of additional software or firmware. An automated HLS method enables designers and architects to explore different algorithmic and implementation options based on a single functional specification, thereby investigating space, power, and performance tradeoffs [13].

Because of recent developments in register transfer level (RTL) synthesis methods, the industrial deployment of high-level synthesis (HLS) tools is becoming an increasingly viable option. Companies that are considered to be industry leaders in semiconductor design, such as IBM [13], Motorola [14], Philips [15], and Siemens [16], have created their own proprietary tools. Major EDA (Electronic Design Automation) suppliers have also begun to commercialize various HLS products. For instance, Synopsys developed a tool known as the "Behavioral Compiler" [17] in 1995. This tool generates RTL implementations from behavioral HDL code and links to tools farther down the production line. Tools such as "Catapult HLS" by Mentor Graphics [18] and "Stratus High Level Synthesis" by Cadence [19] are examples of similar programs. A proposed methodology of a typical flow for HLS in VLSI designs based on MATLAB Simulink HDL Coder is shown in Fig. 1.

The industrial deployment of high-level synthesis (HLS) technology has become more realistic as a result of advancements in record transfer level (RTL) synthesis methods. Specialized tools have been developed by major semiconductor design firms such as IBM [12], Motorola [14], Philips [15], and Siemens [13]. Electronic Design Automation (EDA) industry leaders have also started selling High-Level Synthesis (HLS) solutions. For example, in 1995 Synopsys created the "Behavioral Compiler" tool [15], which generates RTL programs from behavioral HDL code and links to secondary tools. This was accomplished via the use of behavioral HDL. The "Catapult HLS" [18] and "Stratus High Level Synthesis" [19] tools produced by Mentor Graphics and Cadence respectively are similarly comparable. The HLS flow that is often used in VLSI is shown in Fig. 1.

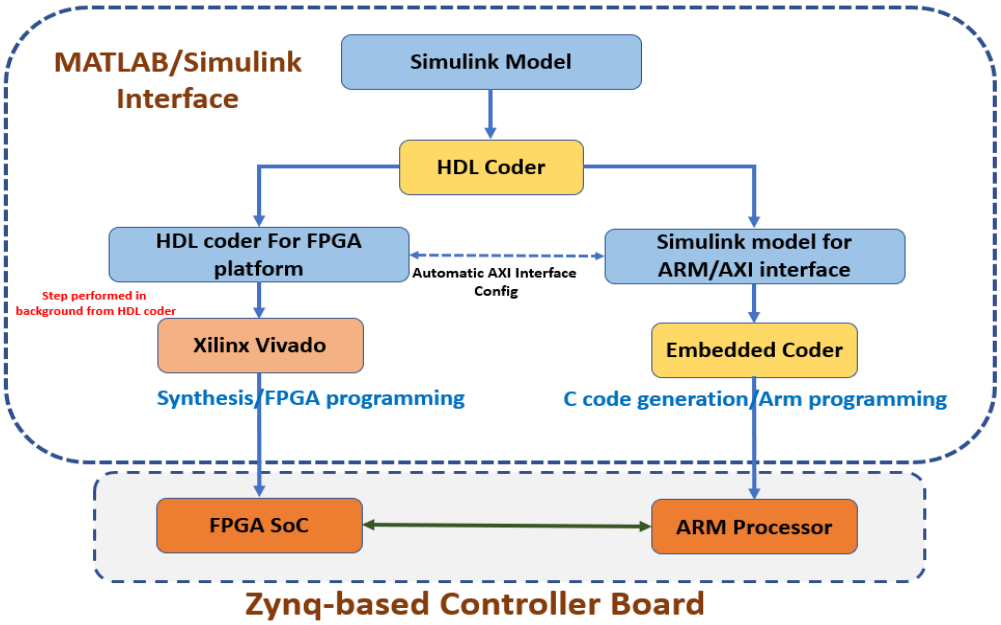

Fig. 1. High level synthesis flow based on MBD.

III. Sobel Edge Detector

The Sobel operator is utilized in the processing of images and computer vision, namely in edge recognition algorithms to emphasize edges in a picture. It is named after Irwin Sobel and Gary Feldman. Sobel and Feldman proposed a proposal of a "Isotropic 3x3 Image Gradient Operator." It is a discrete operator used to produce gradient estimations of the intensity function of an image. Every point within the image represents a gradient vector produced by the Sobel operator. The Sobel operator is computationally efficient since it convolves the image in both vertical and horizontal axes using a small, separable, integer-valued filter. The color gradient estimate it produces lacks precision, particularly for high-frequency variations in the image.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Sobel is a primary edge detection operator that relies on gradients. On an image, it applies a spatial gradient analysis in two dimensions. When trying to estimate derivatives, the operator convolves the original image with two 3x3 kernels. One kernel is used for horizontal alterations, while the other is used for vertical alterations. Each point includes estimations of the both vertical and horizontal derivatives. Here are the kernels:Rotating a kernel through 90 degrees yields a different kernel, as seen in Fig. 1. Gx is used to detect horizontal edges, while Gy is used to detect vertical edges. The two gradients Gx and Gy are utilized to calculate the orientation and magnitude of the edge at a certain location in the image. By combining gradient approximations, an absolute gradient magnitude may be determined at every location in the image. Just square the total value of the squares of the horizontal and vertical components to get the gradient's magnitude: G= $\sqrt{(Gx^2 + Gy^2)}$

IV. Design Methodology

The Full Sobel edge detector (see Fig. 2) for a streaming video was created in MATLAB/Simulink using HDL coder and the required toolbox for modeling. For this experiment using 240 × 320 x 3 input video frames representing each color. Since the HDL implementation is pixel-based rather than frame-based, the input frames need to be transformed to pixels before each pixel can be entered into the project on a clock cycle. Both the Sobel method edge finder and the Simulink library block were simultaneously utilized to process identical input images from their respective hardware implementations. The results of the two were subsequently evaluated against each other.

The length of the video stream, as well as the absolute minimums and maximums for the project's inputs, outputs, and intermediate nodes, were all recorded throughout the period that the simulation ran for, which was one hundred seconds. Later simulation tests were expanded to incorporate this basic and maximum database in order to take into account any and all possible visual signal inputs. After that, the amplitude of each and every signal was determined by taking the range of values for each input, output, and intermediate node into account.

The RTL was then built by the HLS tool with the limitations for all of the data nodes as well as the important inputs and outputs being changed accordingly. Following that step, the optimized RTL was programmed into the FPGA. Because the FPGA output is expressed in pixels, MATLAB was used to convert it to a picture. The finished product consisted of a picture with an acute angle superimposed over

the original. The total approach, which may be seen shown in Fig. 3, consists of HDL code, behavioral implementation, and a common picture source. As shown in Fig. 3, latency components have been added to the input and behavior display channels in order to compensate for the delay caused by the actual hardware implementation of the loop that is executing on the FPGA.

The input picture, the image produced by the MATLAB model, and the output produced by the HDL FPGA program are all shown in Figure 4. As can be seen in the picture, the input image source, the MATLAB behavior model, and the HDL FPGA model all functioned autonomously to process various images taken from the input video stream.

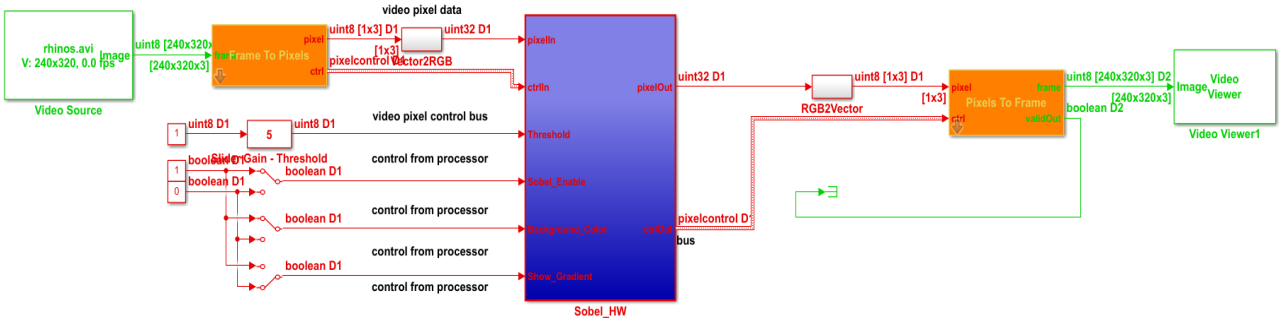

Fig. 2. Sobel edge system: full system.

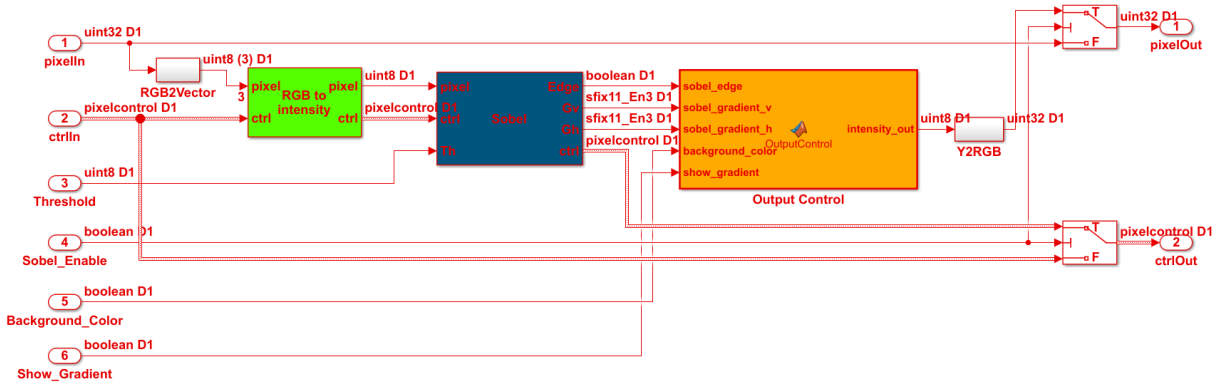

Fig. 3. HDL coder model for Sobel edge detector system.

V. Design Synthesis and Results

Simulations of designs on Simulink After completion, each filter system is separately converted to HDL code. This process is Matlab/Simulink HDL Coder and HDL workflow advisor with add-on realized through. HDL workflow consultant, Convert systems on Matlab and Simulink to HDL code make the necessary settings during the conversion process. It provides an interface. After this stage, a standard system converted to IP block is in Vivado Suite Camera reference with IP integrating system attached to the design. Thus, filter systems were implemented sequentially on the Zedboard. More In later studies, the system will be less on-chip for space coverage and simplification of the software. The designed IP blocks are combined on Simulink was redesigned.

Accordingly, the grayscale conversion and edge detection systems are combined into a single IP block. The median and sharpening filters formed the second IP block. Necessary interconnections were made again and the system was synthesized again. The block design of the final implemented system on Vivado Suite is given in Fig. 4. IP blocks designed in the study are marked on the diagram.

A. Simulation Results

In order to simulate the resulting VHDL RTL code with a testbed that could not be synthesized, the Vivado xSim software was utilized. Figure 5 illustrates the results of the functional simulation of the Vivado xSim simulator. As can be seen in the picture, the reference pixel values are very comparable to the pixel output generated by the technique that was advised. In addition to this, they were the same as the findings of the high-level MATLAB simulation that was carried out using the model with the ideal bit-width.

When the output photographs from both channels were compared to the same input image, this was confirmed. In this investigation, the quantization error that was brought about by choosing narrower "optimum" signal widths was evaluated and compared to the MATLAB-based double-precision model. This was accomplished by using the "FPGA in the loop" co-simulation feature that is available on the MathWorks HDL verifier. According to the results of the root mean square test (RMS), the error in the quantization was less than 1%.

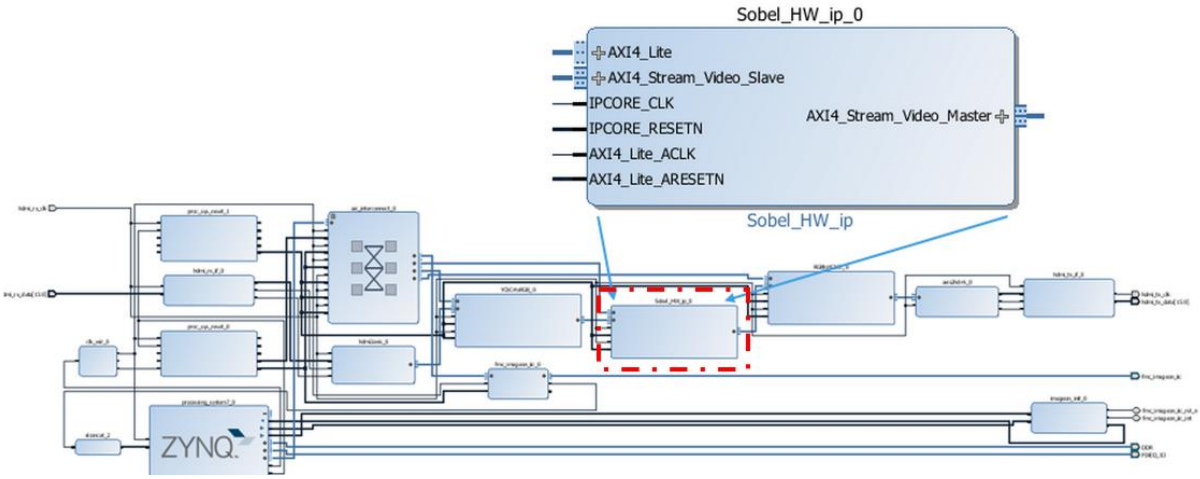

Fig. 4. RTL design for full edge detector.

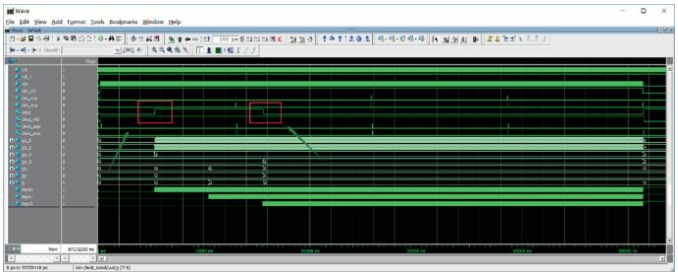

Fig. 5. HDL simulation results (Sobel edge detector).

B. Synthesis Results

Each filter system was converted to HDL code independently when it was determined that the simulations of the designs on simulink were successful. This method improves the functionality of the Matlab/Simulink plugins known as HDL Coder and HDL business articles. A statement that indicates that HDL work instructions have been fulfilled is called a fulfillment of HDL work instructions statement. This statement enables the necessary settings to be made during the process of converting systems designed in Matlab and Simulink to HDL code. After this step, the system was changed to a normal IP gateway, and the IP integrating system in Vivado Suite was used to make the connection between the gateway and the camera reference design. On the Zedboard, the filtering processes were thus carried out in a sequential fashion. The following areas have been modified by combining IP blocks on Simulink, which were created to take up less room on the system chip and to simplify the software. This was done in order to improve performance.

As a consequence of this, the greyscale conversion system and the edge detection system have been merged into a single IP block. The second IP block was made up of the median filter and the sharpening filter. Necessary interconnections were made again and the system was synthesized again. The IP blocks that were created throughout the course of the research can be seen noted on the figure representing the block design of the final implemented system on Vivado 2017.4 Suite. Table I present the resources results for the proposed design.

TABLE I. PROPOSED SOBEL EDGE DETECTOR RESOURCES RESULTS

Resources	Utilization	Available	% Utilization
LUT	5300	53200	10%
LUT RAM	180	17400	1%
Flip-Flops	7500	106400	7%
BRAM	8	140	6%
DSP	11	220	5%
IOS	100	200	50%
BUFG	1	32	3.1%

The Vivado synthesis tool reported a total power of 0.330 W, which was comprised of 0.210 W of dynamic power and 0.120 W of static power. In addition to that, it incorporates a whole host of other optimization strategies, such as high-level synthesis toolkits, resource sharing, and pipeline design. These can be used to improve the results that were mentioned above; however, the scope of this study does not allow for such optimization to be performed.

VI. CONCLUSION

In the process of development is a high-speed, optimal (weak surface), implementation of the Sobel edge-detection algorithm that will be suitable for real-time deployment on the FPGA. The conception was made possible with the use of a ground-breaking conception process known as HLS. This approach restricts intermediate noeuds and the major points of conception to absolute minimums and maximums for each noeud. The RTL for the method was developed on a Xilinx ZedBoard by using an input time video stream with a resolution of 240 by 320 pixels and 8 bit color inputs.

In order to do a functional evaluation of the RTL idea, the Xilinx Vivado xSim simulator was used. We found that our HLS technique results in quantification mistakes that are less than 1% of the total. The findings of the synthesis indicate that our implementation is superior to comparable existing ideas in terms of performance. As a consequence of this, the approach is especially well-suited for FPGA-based applications that call for real-time image processing. Although the method was

tested using the MATLAB HDL codeur procedure with connections to Xilinx Zedboard, it can also be used with other devices and (technology-neutral) FPGA cables.

In spite of the absence of evidence, we are of the opinion that the design process used in this methodology would result in improved results when applied to ASIC synthesis. Future research will involve, in addition to the manner of implementation that was recommended, the optimization of speed, area, and power consumption utilizing optimization approaches given by high-level synthesis tool vendors such as MathWorks, Xilinx, Mentor, and Cadence. In the not too distant future, one of our goals is to broaden the scope of this study to encompass ASIC design.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-2225-02".

REFERENCES

[1] V.H. Schulz, F.G. Bombardelli, E. Todt, A Sobel edge detector implementation in SoC-FPGA for visual SLAM, in: F. Santos Osorio, ´ R. Sales Gonçalves (Eds.), Robotics. SBR 2016, LARS 2016. Communications in Computer and Information Science 619, Springer, Cham., 2016.

[2] C. Cabani, Implementation of an Affine-Invariant Feature Detector in FieldProgrammable Gate Arrays, University of Toronto, 2006, pp. 5–13.

[3] M. Komorkiewicz, T. Kryjak, K. Chuchacz-Kowalczyk, P. Skruch, M. Gorgon, ´ FPGA based system for real time structure from motion computation, in: 2015 Conference on Design and Architectures for Signal and Image Processing (DASIP), Krakow, 2015, pp. 1–7, https://doi.org/10.1109/DASIP.2015.7367241.

[4] S. Liu, Real time implementation of Sobel edge detection system based on FPGA, in: 2017 IEEE International Conference on Real time Computing and Robotics (RCAR), Okinawa, 2017, pp. 339–343, https://doi.org/10.1109/ RCAR.2017.8311884.

[5] C. Xu, B. Yunshan, Implementation of Sobel edge matching based on FPGA, in: 2017 6th International Conference on Energy and Environmental Protection (ICEEP 2017)., Atlantis Press, 2017. [6] T.L. Chao, H.W. Kin, An efficient FPGA implementation of the Sobel edge feature detector, in: 2015 14th IAPR International Conference on Machine Vision Applications (MVA)., IEEE, 2015.

[6] C.Y. Lee, H.J. Wang, C.M. Chen, C.C. Chuang, Y.C. Chang, N.S. Chou, A modified Sobel edge detection for breast IR image, Math. Probl. Eng. 2014 (2014).

[7] J. Vourvoulakis, J. Kalomiros, J. Lygouras, Fully pipelined FPGA-based architecture for real-time SIFT extraction, Microprocess. Microsyst. 40 (2016) 53–73.

[8] H. Ahmed, O. Sidek, An energy-aware self-adaptive System-on-Chip architecture for real-time Sobel edge detection with multi-resolution support, Microprocess. Microsyst. 49 (2017) 164–178.

[9] Xilinx, (2020) Vivado design suite: high-level synthesis. https://www.xilinx.com/ support/documentation/sw_manuals/xilinx2020_3/ug902-vivado-high-levelsynthesis.pdf (accessed 12 Sep, 2020).

[10] MathWorks HDL coder. (2021) https://www.mathworks.com/products/hdl-coder. html, (accessed 14 Aug, 2021).

[11] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, Z. Zhang, High-level synthesis for FPGAs: from prototyping to deployment, IEEE T. Comput. Aid. D 30 (2011) 473–491. https://doi.org/10.1109/TCAD.2011.211059.

[12] R.A. Bergamaschi, R.A. O'Connor, L. Stok, M.Z. Moricz, S. Prakash, A. Kuehlmann, D.S. Rao, High-level synthesis in an industrial environment, IBM J. Res. Develop. 39 (1995) 131–148. https://doi.org/10.1147/rd.391.0131.

[13] Badawi A, Bilal M. High-Level Synthesis of Online K-Means Clustering Hardware for a Real-Time Image Processing Pipeline. Journal of Imaging. 2019; 5(3):38. https://doi.org/10.3390/jimaging5030038

[14] Ahmed Alhomoud, "Real Time FPGA Implementation of a High Speed for Video Encryption and Decryption System with High Level Synthesis Tools" International Journal of Advanced Computer Science and Applications(IJACSA),15(1),2024. http://dx.doi.org/10.14569/IJACSA. 2024.0150172

[15] J. Biesenack, M. Koster, A. Langmaier, S. Ledeux, S. Marz, M. Payer, M. Pilsl, S. Rumler, H. Soukup, N. Wehn, P. Duzy, The Siemens high-level synthesis system CALLAS, IEEE Trans. Very Large Scale Integr. Syst. 1 (1993) 244–253. https://doi. org/10.1109/92.238438.

[16] Ghada Elsayed; Somaya Ismail Kayed. "A Comparative Study between MATLAB HDL Coder and VHDL for FPGAs Design and implementation". Journal of International Society for Science and Engineering, 4, 4, 2022, 92-98. doi: 10.21608/jisse.2022.136645.1056.

[17] Catapult H.L.S., (2019) https://www.mentor.com/hls-lp/catapult-high-level-synth esis/.

[18] Stratus H.L.S., (2019) https://www.cadence.com/content/cadence-www/global /en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level -synthesis.html.

[19] MathWorks HDL Verifier, (2019) https://in.mathworks.com/products/hdl-verifier. html.