# Real Time FPGA Implementation of an Efficient High Speed Harris Corner Detection Algorithm Based on High-Level Synthesis

**Refka Ghodhbani**
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia | Laboratory of Electronics and Microelectronics (EμE), Faculty of Sciences, Monastir University, Tunisia
refka.ghodhbani@nbu.edu.sa (corresponding author)

**Taoufik Saidani**
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia | Laboratory of Electronics and Microelectronics (EμE), Faculty of Sciences, Monastir University, Tunisia
taoufik.saidan@nbu.edu.sa

**Ahmed Alhomoud**
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia
aalhomoud@nbu.edu.sa

**Ahmad Alshammari**
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia
ahmad.almkhaidsh@nbu.edu.sa

**Rabie Ahmed**
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia | Department of Mathematics and Computer Science, Faculty of Science, Beni-Suef University, Egypt
rabie.ahmed@nbu.edu.sa

*Received: 16 September 2023 | Revised: 9 October 2023 | Accepted: 17 October 2023*

## ABSTRACT

**Computer vision systems use corner detection to identify features in an image. In applications such as motion detection, tracking, picture registration, and object recognition, corner detection is often one of the initial steps. In this paper, a real-time image processing system based on Harris corner detection was designed and implemented using Zynq architecture and model-based design tools. The system was based on a development board containing the Zynq-7000 chip, which consists of a combination of FPGA and microprocessor, and the image taken with a high-resolution camera was processed in real-time by applying color conversion and Harris corner detection. The filter hardware designs used in the system were made using the HDL Coder tool in Matlab/Simulink without writing HDL code. The hardware that receives images from the camera was designed on a model-based basis with the Xilinx Vivado 2020. The HDL code that was implemented on the Xilinx ZedBoard using Vivado software was then validated to ensure real-time operation with the incoming video stream. The results achieved exhibited superiority compared to prior implementations in terms of area efficiency (reduced number of gates on the target FPGA) and speed performance on an identical target card. Using the rapid prototyping approach, two alternative hardware**

**accelerator designs were created using various high-level synthesis tools. This design used less than 50% of the host FPGA's logic resources and was at least 30% faster than current implementations.**

***Keywords-rapid prototyping; automated hardware design; corner detection codesign; MBD; HDL coder; Xilinx Zynq-7000***

## I. INTRODUCTION

Recognizing points of interest (or corners), such as detecting contours, is a prerequisite for many computer vision algorithms. In a picture, the places of interest correspond to two discontinuities in the intensity function. These can be created by reflectance function discontinuities or depth discontinuities, such as contours. These include, for example, corners, T-junctions, and spots with significant texture differences. With changing needs and the development of technology, the need for real-time image processing systems, which are used in many fields today, rose. In real-time embedded system designs, which are widely used in areas such as driving support systems, autonomous vehicles, flight control, security, and defense systems, it is a priority to perform the desired work within a determined time frame rather than quickly. For this reason, predictability features along with time stability are indispensable conditions for real-time systems [1]. In such real-time image processing systems, intense processing power is needed as various filters and feature extraction operations are required on the image. Especially in systems requiring high resolutions, the necessity of processing each pixel in the image frame separately and in a very short time has rendered microprocessor-based classical systems inadequate, and thus different solutions have emerged. Among these, DSP, FPGA, or GPU-based systems, as well as System on Chip (SoC), have become more and more common. The DaVinci and OMAP series chips developed by Texas Instruments are examples of Digital Signal Processor (DSP) based systems, while the S32V34 series chips of NXP are found in systems that integrate a processor, a GPU, and special image processing units [2-4]. Such systems can provide real-time operation as hardware accelerators using the advantages of DSPs and coprocessors in signal processing. However, since DSPs are fixed architectures, they address a relatively limited area by being programmed with libraries and interfaces supported by the manufacturer [5]. On the other hand, the use of Field Programmable Gate Arrays (FPGA) in such embedded systems has increased due to their more convenient structure for parallel processing and low power consumption. In FPGA-based systems, the hardware can be completely programmed in the desired architecture and can be a DSP or a hardware block created specifically for the system. In addition, new generation architectures, such as Xilinx Zynq, have processors and programmable logic blocks together, can be programmed with C language just like DSPs, and even hardware design can be made in model-based development environments such as Matlab and Labview. For this reason, an FPGA-based system was preferred in the proposed image processing system. As a result of all these developments, systems that use microprocessors and FPGA hardware together have been shown the best performance in image processing systems [6].

Such solutions that use computer and FPGA hardware together are not practical in embedded system designs because they involve high power and space consumption. The Zynq architecture, introduced by Xilinx in 2012, has closed a very important gap in the embedded system market with its microprocessor and FPGA hardware on the same chip. Due to this SoC architecture, it is possible to realize both high speed and low power and space consumption by implementing all of the hardware and software on a single card. For this reason, this study used the Zedboard development board with Zynq-7000 architecture. However, designs made with Zynq may require a long development process and a development team consisting of different disciplines due to the complexity of the hardware design and its different structure from classical microprocessor systems [7].

In [6], the Canny Edge detection algorithm was applied in real-time on the ZC702 development board using the Zynq platform. The system was designed using Vivado HLS and its internal video library and created a system that can run at 60 fps at 1080p resolution. In [8], a system was proposed to recognize and classify traffic sign images, using 1920×1080 resolution images in real-time with an integrated camera system placed on the Zedboard development board. The entire project was built in six weeks using OpenCV libraries, thus emphasizing the platform's ability to design quickly. This study stated that it took approximately 5 seconds to classify a traffic sign in the developed system [8]. In [9], optical flow algorithms were optimized and comparisons were made on different platforms. Accordingly, the same algorithm was executed on a PC with a Core i7 processor, only on the ARM processor unit of the Zynq-7000 chip, and finally, with special optimization for the programmable logic unit of the Zynq-7000 architecture. The code of the system was synthesized in C language using the Vivado HLS program. The results showed that the performance of the algorithm executed on Zynq was close to the performance of a PC with a Core i7 processor, but the power consumption was only 1/7 of the PC [9].

In [10], an analysis of emotions and facial expressions was performed using different facial recognition methods. This study used OpenCV libraries, C++ language, and embedded Linux operating system on the Zedboard development board using the Zynq-7000 platform, and anger, happiness, and surprise expressions were classified with 97, 100, and 97% success rates, respectively, achieving a performance of approximately 4 to 5 fps. In [11], some image processing algorithms were implemented on the Zedboard development board using the Verilog hardware definition language, which was able to apply real-time filter operations at approximately 40 fps on an image of 256×256 pixels taken with a webcam via USB [11]. In [12], an automatic license plate identification system was designed on the XC7Z020-1CLG484 development board with the Zynq-7000 platform. All system hardware was designed with high-level synthesis tools using MATLAB/Simulink and Xilinx System Generator development environment, the most efficient among the implemented applications was determined, whereas all three methods were more than 90% successful. In [13], the Sobel edge detection filter was applied in real-time on the Zynq-7000 architecture Zedboard using the Vivado HLS tool. This study improved the Sobel filter algorithm to prevent delay in image transmission, making it work faster and achieving a maximum speed of 90.1 fps at 1080p, achieving 30% more performance and 75% less resource usage than previous designs. In [14], the Canny edge detection algorithm was implemented and optimized on Zedboard, using OpenCV libraries and high-level synthesis tools. This study also analyzed and compared the performance of the algorithms, stating that the improved algorithm was up to 3 times faster at the kernel level and up to 7 times faster at the application level [14].

These studies show that the Vivado HLS can be used both as a high-level synthesis tool and to write the C code. This study aimed to go one step further by looking for a solution to the specified design complexity and to create a real-time image processing system using high-level languages. To reduce design complexity, instead of using hardware definition languages, hardware design was performed with Matlab/Simulink, which is a model-based design software, and high-level synthesis tools called HDL Coder, Embedder Coder, and Vision HDL Toolbox. The camera reference design and system software were created with Xilinx's Vivado Design Suite and SDK. In this system, the high-resolution image taken instantly from the camera is subjected to gray tone, edge detection, noise removal, and sharpening filters, and the processed image is transferred to the monitor without delay.

## II. HARDWARE SUBSTRUCTURE

The proposed system used the Zedboard development board with Zynq-7000 architecture, the FMC-HDMI-CAM module produced by Avnet, and the Python 1300-C camera module, which is fully compatible with the system [15-16]. FMC stands for FPGA interconnect and defines a standard interface for FPGA cards that can operate at high speeds. Thus, a connection can be made between peripherals and FPGA cards easily, and all cards with this interface can be used [17]. The connection of the system with the computer is provided via UART and the system status can be monitored via the serial port. However, the system works independently and does not need a computer connection. With the help of buttons and switches on the Zedboard, the desired filter can be selected and other parameters can be adjusted. Figure 1 shows the general scheme of the system.

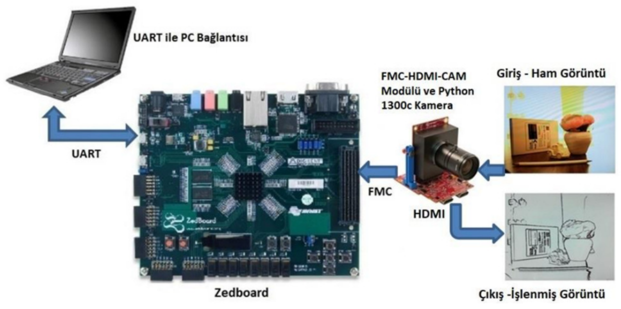

Fig. 1. General scheme of the proposed system.

### A. Zynq-7000 Architecture

The Zedboard development board consists of a Zynq7000 Z-7020 series SoC and peripherals [18]. Zynq-7000 is a Fully Programmable SoC architecture that includes a dual-core ARM Cortex-A9 processor and FPGA components on the same chip. This system has high-speed advanced expandable interface (AXI) buses to provide data transfer between the processor and the FPGA. Thus, it is possible to use these two units together without being independent of each other. In general, the Zynq-700 AP SoC consists of a processor unit (PS-Processing System), a Programmable Logic (PL) unit corresponding to the FPGA system, and interconnections [19]. Figure 2 shows a simple principal diagram of the Zynq architecture.

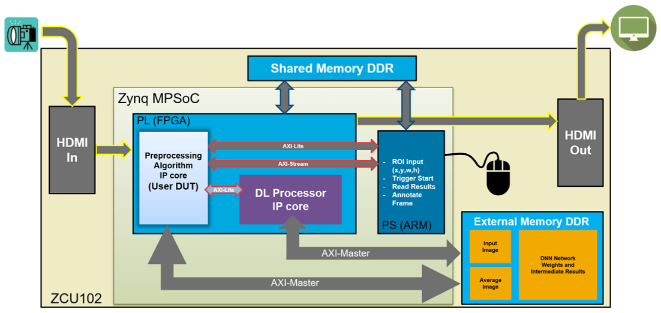

Fig. 2. Simple principle diagram of the Zynq7000 architecture.

### B. System Design Flow

After bringing together the hardware components of the system, a reference design that can take images from the camera should be realized. After the image is taken from the camera successfully, the filters to be used are designed and simulated in the Matlab/Simulink environment, and these filters are packaged as standard IP blocks with the HDL Coder tool. These IP blocks are integrated into the camera reference design, created by incorporating them into the Vivado environment. Finally, the system software was written in the Xilinx SDK environment the necessary drivers were included, and the system was given its final shape.

High-Level Synthesis (HLS) is gaining popularity as an approach that allows designers to express design behaviors at high abstraction levels while ensuring continuous verification throughout the design cycle. Vivado HLS [10] and MATLAB HDL Coder [11] are examples of HLS tools. These are often used by digital designers and architects to create and implement algorithms for a variety of aerospace, communications, image processing, deep learning, and neural network applications. HLS technologies can help reduce code complexity by a factor of seven to ten. They enable the reuse of behavioral IP across projects and allow verification teams to apply high-abstraction-level modeling techniques, such as transaction-level modeling [12].

Furthermore, most modern chip systems include integrated processors. The coexistence of microprocessors, DSPs, memory, and custom logic on a single chip necessitates the inclusion of additional software or firmware in the design process. As a result, an automated HLS process enables designers and architects to experiment with multiple algorithmic and implementation options to investigate various area, power, and performance tradeoffs from a shared functional specification. As a result of advancements in Register Transfer Level (RTL) synthesis techniques, industrial deployment of HLS tools has become increasingly feasible. Major semiconductor design houses, such as IBM [13], Motorola [14], Philips [15], and Siemens [16], have developed proprietary tools. Major Electronic Design Automation (EDA) suppliers have also begun to commercialize various HLS products. For example, Synopsys released the Behavioral Compiler tool [17] in 1995, which builds RTL implementations from behavioral Hardware Description Language (HDL) code and connects to downstream tools. Similar tools include Mentor Graphics' Catapult HLS [18] and Cadence's Stratus High-Level Synthesis [19]. Figure 3 depicts a typical flow for HLS in VLSI designs.

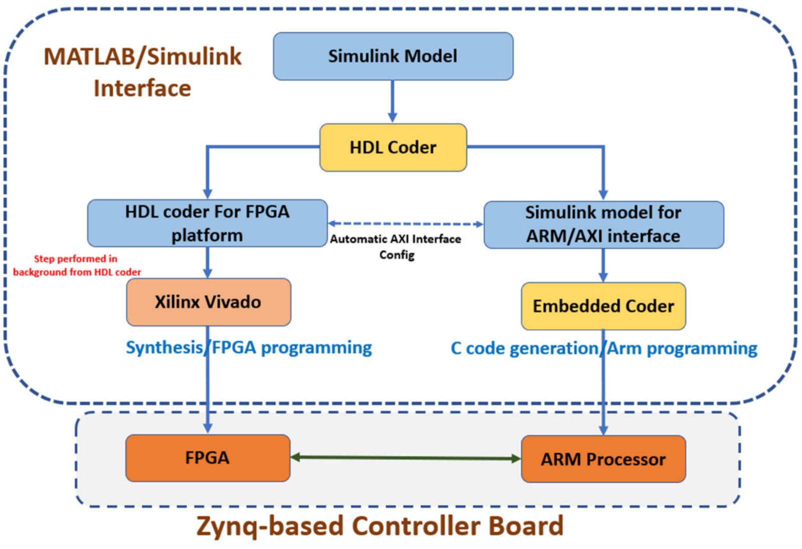
MATLAB/Simulink Interface
Simulink Model
HDL Coder
HDL coder For FPGA platform
Automatic AXI Interface Config
Simulink model for ARM/AXI interface
Step performed in background from HDL coder
Xilinx Vivado
Embedded Coder
Synthesis/FPGA programming
C code generation/Arm programming
FPGA
ARM Processor
Zynq-based Controller Board

Fig. 3. System design flow based on Simulink /HDL coder.

## *C. Camera Reference Design*

This study used the Avnet FMC-HDMI-CAM module to provide a connection to the camera via the high-speed FMC interface. The Python-1300c camera module, which is compatible with this module, was chosen as the image sensor. This module has the ON Semiconductors Python-1300 image sensor and can transfer images with a resolution of 1280×1024 pixels at 210 fps [16]. Thus, the infrastructure necessary for a real-time system was created. This design aimed to convert the image into the AXI4-Stream data type by performing some operations on the raw image coming from the camera sensor and making it ready for further processing. In this way, the image frames taken from the camera could be transferred to the screen without any errors. After these processes were carried out, the IP blocks required for image processing were designed according to the camera reference design. To get the image from the camera correctly, the Xilinx Video and Image Processing Pack and the 3rd party IP blocks provided for the camera hardware were downloaded and included in the system. After the necessary procedures were completed, the synthesis and implementation phase was passed, the test software of the system was created, and the image was successfully projected onto the screen. Figure 4 shows the status of the camera reference design and the IP blocks used in the system. The camera image is first converted to the AXI4-Stream data type, and then the missing color components are reconstructed with the CFA IP block. The image is converted to a YCbCr format with the RGB to YUV block. Using the Chroma Resampler IP block, the image is reduced to a 16-bit format, over 24 bits, and the 1280×1024 pixel image is centered on the 1920×1080 pixel monitor with the On-Screen Display IP block. Then, the image is converted again to be transferred to the monitor using a connection to the monitor from the HDMI output on the FMC-HDMI-CAM module. With the AXI VDMA IP block, data exchange was provided directly with the system memory, thus protecting the system from possible bottlenecks. Figure 5 shows the created working image of the created camera reference system.

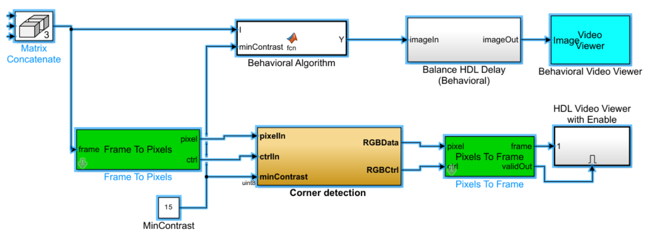

Fig. 4. Camera refrerence design.

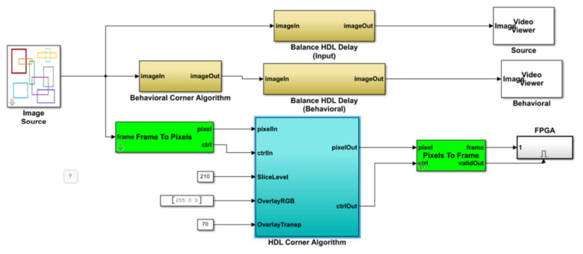

Fig. 5. Corner detection system with Simulink /HDL coder.

### *D. Image Processor System*

After the camera reference design, the image processing filters were designed using the Xilinx and Mathworks tools. The system design was completely model-based in the Simulink environment, and after the simulations were made on Simulink, HDL code was automatically generated with HDL Coder, and these codes were converted into usable IP packages. In addition, the Embedder Coder, Simulink Coder, Matlab Coder, and Vision HDL Toolbox and their additional packages for Zynq were installed in Matlab 2022 and Vivado 2020.2 to be used in the design of the system. The system running on Simulink is a kind of serial interface called a streaming pixel interface. In this method, pixels are transferred serially, and data transfer takes place over two kinds of signals, namely, data and control. This interface is fully compatible with the AXI4-Stream and ensures that the designed IP blocks work in full compliance with the camera reference system.

## III. DESIGN AND SIMULATION OF THE SYSTEM IN SIMULINK ENVIRONMENT

Four filters were individually designed with HDL Toolbox blocks and combined into two IP blocks. In this way, without increasing design complexity all filters could be activated at the same time.

### *A. Harris Object Corner Detector*

Corners are portions of a picture with a significant change in intensity and resistance to picture distortions. Based on the light intensity of the pixels, Harris points are generated to correlate to wedges. A detector based on the autocorrelation expression of the intensity fluctuations listed below is suggested:

$$M(x,y) = \sum_{(u,v)} w(u,v) \cdot \begin{pmatrix} I_x^2(x,y) & I_x I_y(x,y) \\ I_x I_y(x,y) & I_x^2(x,y) \end{pmatrix}$$

with $I_x$ and $I_y$ being the local derivatives in $x$ and $y$, and $w(u, v)$ is a weighting on the window $(u, v)$. The study of the eigenvalues of the matrix $M$ makes it possible to determine whether a point is a corner, a homogeneous region, or an outline. A final criterion is calculated from $M$, allowing deciding on the type of point found.

### *B. Grayscale Converter*

Vision HDL was used to design a grayscale conversion filter operation. Accordingly, all three color values were fixed. The gray tone values were obtained by multiplying with the coefficients [3]. The YUV format and its most commonly used encoding were used, following the ITU rec.601 standard.

## IV. DESIGN SYNTHESIS AND RESULTS

After completing the designs on Simulink, each filter system was separately converted to HDL code. This process was carried out using the Matlab/Simulink HDL Coder and the HDL workflow advisor. They necessary settings were made during the conversion process. After this stage, the IP block in the Vivado Suite Camera reference was attached to the design with the IP integrating system. The filter systems were implemented sequentially on the Zedboard. In later studies, the system will be less on-chip for space coverage and simplification of the software. Accordingly, the grayscale conversion and edge detection systems were 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.

### *A. Simulation Results*

The produced VHDL RTL code was simulated using a non-synthesizable test bench and the Vivado xSim program. The reference pixel values are equal to the pixel output from the suggested approach. They were also found to be identical to high-level simulation results obtained with MATLAB on the optimum bit-width model. This was proven by comparing the output photos from both paths with the same input image. This study also compared the quantization error produced by the choice of the lowered optimal signal widths with the double precision model running in MATLAB. This was achieved with the FPGA in the loop simulation feature of the MathWorks HDL Verifier [20].

### *B. Synthesis Results*

Table I shows the resource usage. As can be seen, at most, 35% of the available resources were used. It should also be noted that no optimization work was carried out here. In the system, the main clock speed was determined at 148.50 MHz, and the pixel clock speed, at which the camera and image processing blocks work, was determined at 108 MHz. Although EYH represents the screen refresh rate in the formula, the total horizontal pixels and total vertical pixel values consist of the sum of active pixels, gap pixels, and synchronization pixels.

TABLE I. PROPOSED HARRIS CORNER DETECTOR IMPLEMENTATION RESULTS

| Resource | Utilization | Available | Utilization percentage |
|---|---|---|---|
| Look Up Table (LUT) | 5917 | 53,200 | 11.12% |
| LUT RAM | 506 | 17,400 | 2.91% |
| Flip-Flops | 10,981 | 106,400 | 10.32% |
| BRAM | 37 | 140 | 26.79% |
| DSP | 21 | 220 | 9.55% |
| IOs | 102 | 200 | 51% |
| BUFG | 1 | 32 | 3.13% |

Accordingly, it was determined that the image processing system can operate at 1280×1024 at 60 Hz refresh time and, thus, can process 60 fps. This shows that the system is in real-time operating conditions.

### *C. Implementation of the System Software*

The software was designed following the hardware. The design was carried out in C using the SDK tool in Vivado Suite. In addition to the processes used in the software, such as the preparation and execution of IP blocks, the setting of input-output units and filter parameters were also adjusted. In the developed system, filters can be changed during operation and more than one filter can be active at the same time. The system is controlled with the buttons and switches of the Zedboard. The active status of the filters can also be monitored from the LEDs on the Zedboard. Using multiplexer logic, the system is designed so that the relevant filter is selected according to the status of the switches on the Zedboard. Among the Zeboard keys, the system activates the relevant filter according to the status of the keys, with DS0 being the lowest-value bit and DS7 being the highest-value bit.

## V. CONCLUSION

This study designed and implemented a real-time image processing system on the Zedboard development board. Due to the high processing power requirement of such systems, the Zynq architecture was chosen, and both hardware and software were designed. Hardware was designed using Vivado Suite and related tools developed by Xilinx for Zynq, while model-based development tools, such as HDL Coder and Vision HDL Toolbox developed by Mathworks, were used in the design of the image processing system. In this way, there was no need to write HDL code manually and the design time was shortened. A system that can operate at 60 Hz with a resolution of 1280×1024 pixels was developed on the Zedboard development board. Sobel edge detection, median filter, gray tone converter, and sharpening filter designs were applied, which can be controlled from the input and output units of the board. The designed system can meet needs and fulfill real-time working conditions. However, although resource usage is not very high, it can decrease further with optimizations in the HDL Coder tool. The proposed design was completely reusable, and it is possible to reuse IP blocks or the entire system designed in future studies and different systems. Future studies can investigate more advanced real-time object recognition and tracking applications using the proposed system as a preprocessing unit.

Hardware acceleration of corner edge detection for 1920×1080 images was implemented in a Xilinx Zynq-7000 SoC hardware platform. Simulation and synthesis results were obtained using Vivado 2020.2. Future research will focus on enhancing the performance metrics of speed, area, and power consumption using optimization techniques provided by prominent high-level synthesis tool providers, including MathWorks, Xilinx, Mentor, and Cadence, in conjunction with the proposed implementation approach. Furthermore, there are plans to expand the scope of this study to include ASIC designs in future research endeavors.

## REFERENCES

[1] V. H. Schulz, F. G. Bombardelli, and E. Todt, "A Harris Corner Detector Implementation in SoC-FPGA for Visual SLAM," in *Robotics*, Uberlândia, Brazil, 2016, pp. 57–71, https://doi.org/10.1007/978-3-319-47247-8_4.

[2] C. Cabani, "Implementation of an affine-invariant feature detector in field-programmable gate arrays," MSc Thesis, University of Toronto, Toronto, Canada, 2006.

[3] T. Saidani and R. Ghodhbani, "Hardware Acceleration of Video Edge Detection with Hight Level Synthesis on the Xilinx Zynq Platform," *Engineering, Technology & Applied Science Research*, vol. 12, no. 1, pp. 8007–8012, Feb. 2022, https://doi.org/10.48084/etasr.4615.

[4] S. Liu *et al.*, "Real-time implementation of harris corner detection system based on FPGA," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Okinawa, Japan, Jul. 2017, pp. 339–343, https://doi.org/10.1109/RCAR.2017.8311884.

[5] C. Xu and Y. Bai, "Implementation Of Harris Corner Matching Based On FPGA," presented at the 2017 6th International Conference on Energy and Environmental Protection (ICEEP 2017), Jun. 2017, pp. 807–811, https://doi.org/10.2991/iceep-17.2017.141.

[6] T. L. Chao and K. H. Wong, "An efficient FPGA implementation of the Harris corner feature detector," in *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, Tokyo, Japan, Feb. 2015, pp. 89–93, https://doi.org/10.1109/MVA.2015.7153140.

[7] C. Y. Lee, H. J. Wang, C. M. Chen, C. C. Chuang, Y. C. Chang, and N. S. Chou, "A Modified Harris Corner Detection for Breast IR Image," *Mathematical Problems in Engineering*, vol. 2014, Jul. 2014, Art. no. e902659, https://doi.org/10.1155/2014/902659.

[8] H. Mestiri, I. Barraj, and M. Machhout, "AES High-Level SystemC Modeling using Aspect Oriented Programming Approach," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6719–6723, Feb. 2021, https://doi.org/10.48084/etasr.3971.

[9] S. S. Rafiammal, D. N. Jamal, and S. K. Mohideen, "Reconfigurable Hardware Design for Automatic Epilepsy Seizure Detection using EEG Signals," *Engineering, Technology & Applied Science Research*, vol. 10, no. 3, pp. 5803–5807, Jun. 2020, https://doi.org/10.48084/etasr.3419.

[10] "Vivado Design Suite User Guide: High-Level Synthesis," Xilinx, UG902 (v2018.3), 2018.

[11] "Mathworks HDL Coder." https://www.mathworks.com/products/hdl-coder.html.

[12] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-Level Synthesis for FPGAs: From Prototyping to Deployment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 473–491, Apr. 2011, https://doi.org/10.1109/TCAD.2011.2110592.

[13] R. A. Bergamaschi *et al.*, "High-level synthesis in an industrial environment," *IBM Journal of Research and Development*, vol. 39, no. 1.2, pp. 131–148, Jan. 1995, https://doi.org/10.1147/rd.391.0131.

[14] K. Kucukcakar, C. T. Chen, J. Gong, W. Philipsen, and T. E. Tkacik, "Matisse: an architectural design tool for commodity ICs," *IEEE Design & Test of Computers*, vol. 15, no. 2, pp. 22–33, Apr. 1998, https://doi.org/10.1109/54.679205.

[15] P. E. R. Lippens *et al.*, "PHIDEO: a silicon compiler for high speed algorithms," in *Proceedings of the European Conference on Design Automation*, Amsterdam, Netherlands, Oct. 1991, pp. 436–441, https://doi.org/10.1109/EDAC.1991.206442.

[16] J. Biesenack *et al.*, "The Siemens high-level synthesis system CALLAS," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 3, pp. 244–253, Sep. 1993, https://doi.org/10.1109/92.238438.

[17] D. W. Knapp, Behavioral Synthesis: Digital System Design Using the Synopsys Behavioral Compiler. Englewood Cliffs, NJ, USA: Prentice Hall PTR, 1996.

[18] "Catapult High-Level Synthesis & Verification | Siemens Software." https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/.

[19] "Stratus High-Level Synthesis." https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html.

[20] "Mathworks HDL Verifier." https://www.mathworks.com/products/hdl-verifier.html.