Advances in Engineering Research, volume 207
Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021)

Image Processing Implementation on the Field Programmable Gate Array

Dianthika Puteri Andini[1,*] Gunawan Sugiarta[1] Feni Isdaryani[1] Fadhel Muhammad Hafidh[1]

[1] *Department of Electrical Engineering, Bandung State Polytechnic, Bandung, Indonesia*
[*] *Corresponding author. Email: dianthika@polban.ac.id*

ABSTRACT
Image processing is an essential principle that is increasingly being applied in innumerable applications to optimize abundant design complexities to modified version by implementing the hardware platform to practice the algorithms. This is especially for the practice of a real-time system that performing frames of the image processing directly. Currently, the FPGA technology becomes a viable target for implementing suitable image processing applications. Real-time image processing can be implemented into the FPGA platform by using VHDL. FPGA can be the requirement of real-time imaging applications through maintaining data and computational. This paper focuses on software simulation and hardware implementation of image processing algorithms for the use in an FPGA based on the real-time processing. It aims to verify the real-time performance and functionality of the proposed image processing algorithm using the FPGA board from Altera named Cyclone IV EP4CE10E22C8. It is compatible to implement the process in a real-time system with the Histogram Equalization proposed method. This paper shows design exploration to the real-time environment in the practice of implementing image processing algorithm in the specified FPGA board, especially in customized image processing simulation of Histogram Equalization, application of proposed algorithm into image processing FPGA board, and exploration of practice in FPGA board of compilation and downloading.

Keywords: *Image Processing, Histogram Equalization, FPGA, Altera Cyclone.*

1. INTRODUCTION

The growth of developing applications using image processing then becomes the image processing algorithms plays essential principle that is increasingly being applied in innumerable applications. The processing algorithms try to optimize abundant design complexities to modified version implementing in hardware platform to practice the algorithms. This is especially for the practice of a real-time system that performing frames of the image processing directly. Conversion and transformation of the algorithm for efficiency take part in the role of practice performance. This performance can be applied on the hardware as processor called Field Programmable Gate Array (FPGA). Currently, FPGA technology becomes a viable target for implementing suitable image processing applications. Real-time image processing can be implemented into the FPGA platform by using Verilog Hardware Description Language (VHDL); Very High-Speed Integrated Circuit (VHSIC); VHSIC Hardware Design Language (VHDL) [1][2].

Many image processing applications are essential that various operations be performed on each pixel in the image resulting in an even large number of operations per second. One option is to apply an FPGA because frequent growth in the size and functionality of FPGAs over recent years has resulted in an increasing interest in the application as implementation platforms for image processing, particularly real-time video processing [3][4][5]. FPGA could be structured to separate the image and distribute the resulting sections to multiple pipelines which could process data concurrently. Practically, parallelization is the main idea to the processing mode and hardware constraints of the system. This forces the designer to solve hardware issues such as concurrency, pipelining, and priming, which many image processing experts are unfamiliar with. The different image processing algorithms for RGB to grayscale, an

algorithm for image negatives, and image enhancement are available to be implemented in a system [6][7].

To overcome this problem, the general approach of the language is to hide low-level details by compiling the parallelism optimization automatically. Compiling and downloading is the result of an FPGA configuration file.

Software-to-hardware conversion schemes can be employed in MATLAB to VHDL which results can be implemented to hardware. MATLAB can manipulate matrices in general and is perfect for some image processing applications. In MATLAB, an image will be represented in the form of a matrix that allows a designer to develop optimal matrix operations for the application of algorithms. Whereas if the algorithm simulated with the board implementation is different, it will cause a slightly slow process so synchronization between software and hardware is required.

This paper focuses on software simulation and hardware implementation of image processing algorithms for the use in an FPGA based on real-time processing. Regarding the programming aspect, MATLAB can handle images as a matrix which allows a better understanding of the processing. The implementation in using MATLAB is far from being the fastest to generate HDL programs. The HDL Coder tool of MATLAB can make designers be focused on system-level design instead of struggling in specific programming. MATLAB/Simulink can interpret an environment of model-based design to shorten the time and meet the end performance approach.

To verify the real-time performance and functionality of the proposed image processing algorithm, FPGA board from Altera named Cyclone IV EP4CE10E22C8 that is compatible to implement the process in a real-time system is used. The Cyclone board presents some specific signals to display images with the signals.

2. LITERATURE REVIEW

2.1. Image Processing Algorithm

The project of image processing is to develop some image processing functions based on a proposed algorithm called Histogram Equalization. The histogram of a digital image is a classification of its discrete intensity levels in the range. Histogram equalization is a method to process images to adjust the contrast of an image by modifying the intensity distribution of the histogram. The objective of this technique is to set out a linear tendency to the cumulative probability function associated with the image. The processing of histogram equalization relies on the utilization of the cumulative probability function which is a cumulative sum of all the probabilities lying in its domain.

A real-time system response to a specific time meaning in image processing is regularly capturing images, analyzing images to obtain data, and controlling several activities. This algorithm can be applied in vision systems for path planning where the timing is a critical point of the parameter. Delay is a quiet criterion depending on the result. Then, the execution time is considered as a performance impact for the results. One way to support this approach is developed by image processing algorithm in real-time enhancement image processing algorithm. A histogram is one of many methods to support the image processing approach [8-12].

A histogram is a distribution of the number of pixels according to their intensities. In this case, it is to scrutinize the image to regulate this distribution. The various intensity images are the input that can be controlled and the perceived results of the normalized histogram and corresponding level of output images. The part of equalization shows the enhance of dark image contrast that the histogram is already distributed on all level. The histogram is the basis for various spatial domain processing techniques. Histogram manipulation can be used for image enhancement. In addition, to provide beneficial image statistics, the information attached to the histogram is also entirely beneficial in other image processing applications, such as image compression and segmentation. Histograms are straightforward to calculate in software as well as hardware implementations, making them an accepted approach for real-time image processing [13][14][15].

Histogram Equalization produces a transformation function that generates output images with uniform histograms. When automatic upgrades are desired, this way of approach can be considered because the results of this technique are predictable, and these methods are effortless to implement. However, there are applications where histogram equalization does not match. It is sometimes useful to be able to determine the histogram shape that matches with the image processed. The method used to create images that have a specific histogram is called histogram matching or histogram specification [16][17][18].

Histogram equalization is a preliminary process for image processing and enhancement with an implementation on hardware as the focus [19][20]. The adjustment of the criteria of histogram equalization and intensities can be better applied in the image improvement than the manual adjustment. First, the input will be calculated to histogram from the image pixels. Then, the next process is to compute the Cumulative Distribution Function (CDF) based on the created histogram. The histogram is computed as a set of bins by each tile. Finally, the last step is scaling and mapping the image pixels and creating equalized output.

2.2. FPGA Board

Over the last 20 years, FPGA has transformed from glue logic to computing platforms. It functionally comes up with a reconfigurable hardware platform for performing logic and algorithms. Being fine-grained hardware, FPGA is capable to draw on the parallelism inherent within a hardware design while simultaneously keep going the reconfigurability and programmability of software. FPGA is being practiced as a platform for accelerating computationally intensive tasks. This is particularly depicted in the field of image processing, where the FPGA-based acceleration of imaging algorithms has become mainstream. This is even more within an embedded environment where the power and computational resources of conventional processors are not up to the task of managing the data throughout and computational requirements of real-time imaging applications.

In contrast, it is compulsory in FPGA to design not only the algorithm but also the computational architecture, which leads to an explosion in the design space complexity. This way, integrated with the complexities of managing the concurrency of a highly parallel design and the bandwidth issues associated with the high volume of data associated with images and video, has led to a wide range of approaches and architectures used for realizing FPGA-based image processing systems [21][22][23].

Specifications [24]:

1. Size: 10cm × 6.5cm.
2. Main chip: ALTERA FPGA Cyclone IV EP4CE10E22C8.
3. Extended Interface: 2*22 pin interface, 2*28 pin interface, 2.54mm pitch pins.
4. Power supply: USB power supply interface, used for power supply.
5. Program debug and download interface : 2*5 pin JTAG interface, used for download program.
6. Onboard a power LED, 2 user LEDs.
7. Onboard 1 reset key, 2 user keys.
8. Onboard large capacity serial configuration chip EPCS16 or M25P16, save the program when power is off.
9. NIOS or large capacity cache, using SDRAM module to expand.

The Video and Image Processing Suite is a collection of intellectual property (IP) cores that can be applied to facilitate the development of custom video and image processing algorithms. The cores range from simple building-block functions, such as color-space conversion to sophisticated video-scaling functions that can implement programmable polyphase scaling. Along with an easily integrated connectivity portfolio, the Video and Image Processing Suite presents a comprehensive system design philosophy for the rapid new design creation and easy integration of custom value-added features.

3. RESEARCH METHODOLOGY

The algorithm method for image processing in this research is in two intermediate levels. The histogram is utilized as a representation in another version of pixel conversion. This method can be shown in the higher level that is high-level. It means that using an image as representation in the final information. The operation of intermediate and high levels is a corresponding to decrease parallelism by changing the pixel that brings about a reduction of total data and processing time.

Histogram equalization is used to improve contrast in images by computing histograms that correspond to the section of images and distribute the luminance values of the images. It is suitable for improving local contrast and strengthening the definitions of edges in each region of an image.

MATLAB/Simulink can transform the model-based design to a familiar hardware approach by using HDL code generation automatically. It also verifies and optimizes the model based on design requirements. Therefore, the model-based design effectively works as a rapid tool for system validation and testing. HDL coder allows image processing algorithm to FPGA implementation as illustrated in Figure 1.

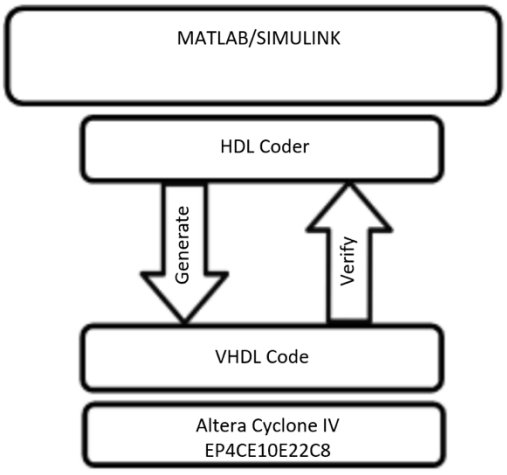

Figure 1 Model-based design

MATLAB Vision HDL Toolbox™ Histogram library implements an approach of histogram equalization for image processing algorithm. A simulation can be run in the model of Simulink part as software development. The model provides a hardware-compatible algorithm to enhance the contrast of images by applying this proposed algorithm. Video Source reads the multimedia file. Figure 2 shows the block system that is the combination of software and hardware design.

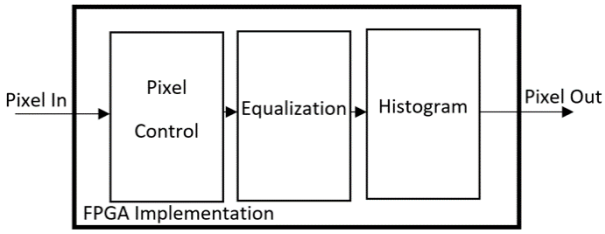

Figure 2 Algorithm design

a. Pixel control is a video partition that divides large images into four non-overlapping small images for parallel histogram computing.
b. The histogram is HDLHistogram that calculates the accumulated histogram of an image.
c. Equalization implements the histogram equalization technique in the original image and produces an image that has been increased in contrast.
d. Pixel In and Out are input and output of the system

Video Partition, the histogram is enumerated for the complete image and this operation grasps a considerable total of time because each of the regions of interest (ROI) of the image part is processed. The video partition component in this example separates large images into four small images that do not overlap. The histogram is computed on four small images simultaneously. Each small image is connected to the Frame To Pixels block to produce a pixel stream and corresponding control signal.

HDLHistogram is optimized for HDL code generation. Histograms of pixels are calculated by using the Vision HDL Toolbox Histogram. Since the input image is grayscale with the uint8 data type, the input pixels are grouped into 256 bins. The model scrutinizes the bin histogram figuring sequentially. The bin value is sent for cumulative histogram calculation. After 256 bin values are interpreted, the model generates a big reset to reset all bins to zero. Histograms are collected from each small image then next calculating the histogram accumulation of large images.

Equalization can be appealed to frames where accumulated histograms are being counted or frames afterward. If it is applied to the current frame, the input video needs to be stored. The results of the video that has been done in equalization then be compared to the original video.

Input can be taken from the source of the camera with a specific parameter for the simulation. Consequently, the output would be shown in the VGA monitor as an interface.

4. CONCLUSION

4.1 Result

The domain tasks are solved by software and hardware implementation as image processing in the first step and FPGA board is continuously in execution. Histogram equalization is an enhancement process that involves in improving the quality of the image.

The main part of histogram equalization is depicted in Figure 3, containing Video Source, Video Partition, HDLHistogram, Equalization, and Video Display.

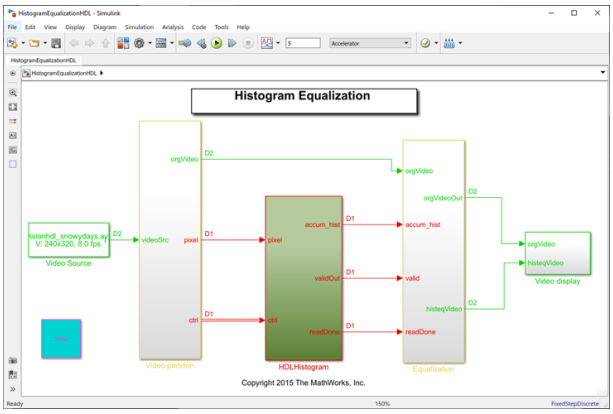

Figure 3 Block of Histogram Equalization [25]

The simulation depicted in Figure 4 shows the difference between the original image (left) and the result (right) performing enhancement of contrast image.

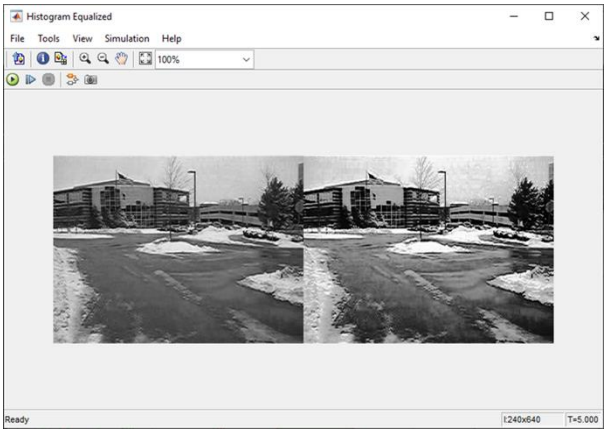

Figure 4 Result of Histogram Equalization

Simulation is the first operation software procedure with direct remaining hardware which is implemented on the FPGA board. The image processing realizes on the platform by corresponding function leading to the resources. The model requires to be maintained for less memory and fewer elements to be classified by reducing the computation time. The FPGA gives modest acceleration over software implementation on a high-end computer. Before implementation to the board, HDL code generation should be applied to create an easier program than writing it from zero. HDL code generation

can be applied with Simulink as in Figure 5 or using MATLAB command.

makehdl(HistogramEqualizationHDL/HDLHistogram)

makehdltb(HistogramEqualizationHDL/HDLHistogram)

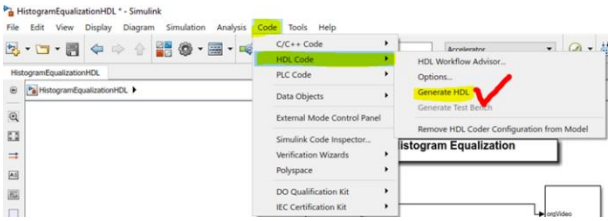

Figure 5 Simulink HDL code generator

All processes are generated to the folder that can be synchronized into the path in computer system completed HDL TestBench generation displays in Figure 6 proved with "*.vhd" file extensions are successfully categorized as VHDL file into a folder. Every single file and parameter set is generated by MATLAB and HDL Coder into VHDL programming to execute in the FPGA objective. Figure 7 depicts the file that has been created by converting the Simulink model. In that case, the VHDL file has been successfully created.

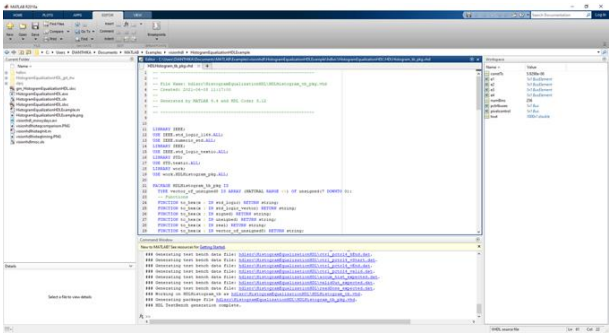

Figure 6 MATLAB HDL generator

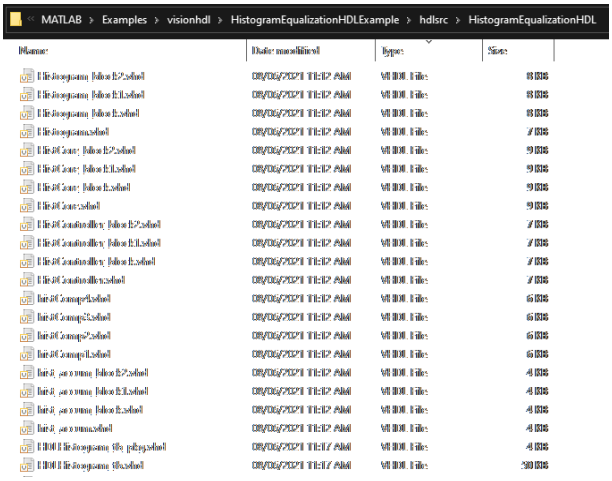

Figure 7 File VHDL on a computer system

System implementation on the FPGA board establishes with compilation in the design tool which is Intel Quartus Prime. Lite Edition. To upload the program, the system must run **New Project Wizard** that family, device, and board settings of the program setting. Family of Cyclone IV E dan EP4CE6E22C8 should be chosen as shown in Figure 8. Next, in **EDA Tool Setting,** 'Tool Name' and 'Format Simulation' should be changed to 'ModelSim-Altera' and 'VHDL' completed with the finish button figured in Figure 9. The top-level entity of the project can be the same as the VHDL code displaying in Figure 10 that is a flow summary of compilation and downloaded VHDL file to FPGA board.

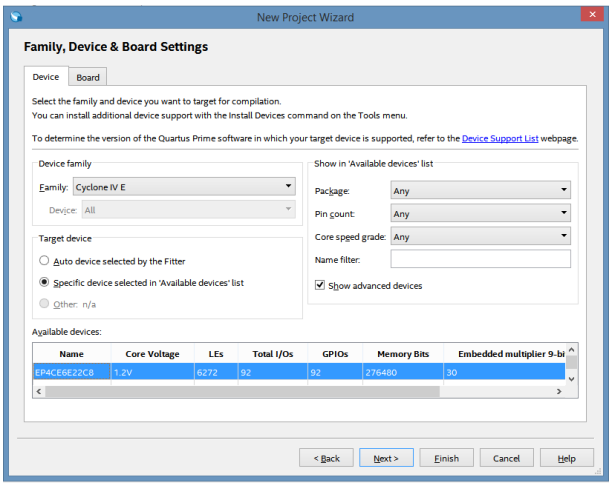

Figure 8 Board setting wizard

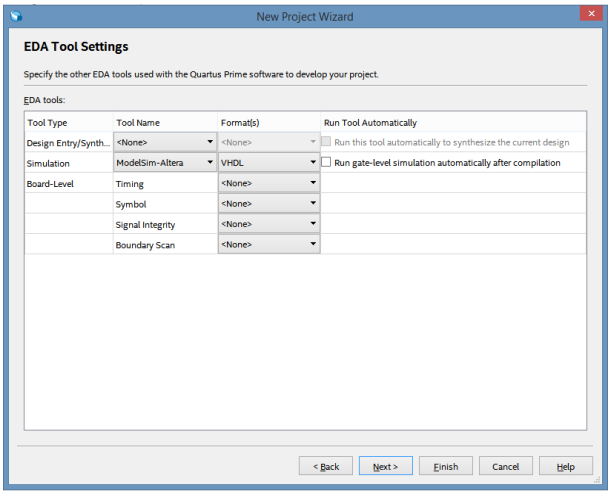

Figure 9 Tool settings

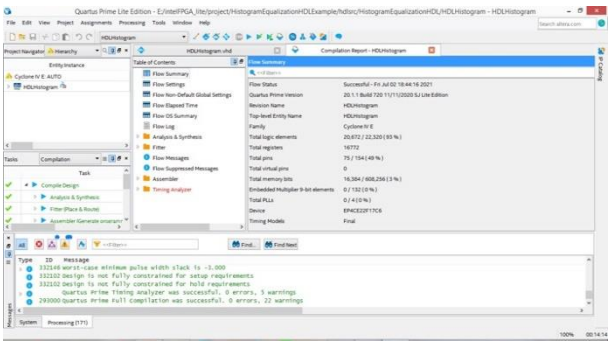

Figure 10 Practice to FPGA board

This second method is utilized by using the camera as input and output to VGA monitor implemented on FPGA

board using VHDL language. First, the software program for FPGA implementation should be synthesized to complete validation as shown in Figure 11. The next process is to upload the program to FPGA proven by "100% Successful" on the software as shown in Figure 12.

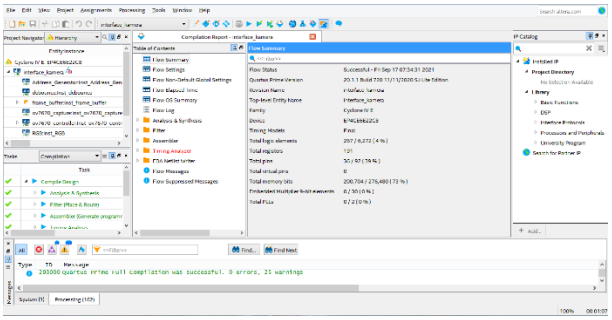

Figure 11 Synthezise program

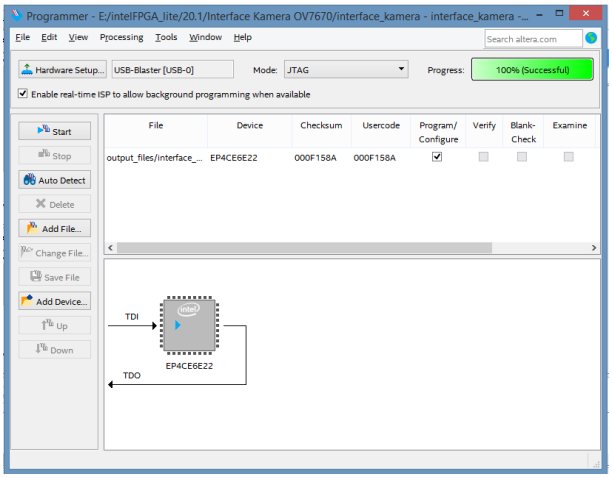

Figure 12 Uploading program to FPGA

The result of the camera interface to the VGA monitor can be seen in Figure 13 with the specification of LCD 7 inch, 1024x600. The camera has an active array size of 640x480 with the output formats 8 bit, RGB colour, and 30 fps for VGA. Then, the FPGA board implementation can be seen in Figure 14. The new entity for camera configuration by creating a .vhd file is in the same directory. This entity is also used to capture images from camera or real-time mode to video. Therefore, RAM configuration is also set up by specifying memory size as several bits and change the total of bits memory. The final step is the VGA configuration aims to give the signal to the VGA monitor. The RGB code is the useful information because of the specification of the camera interface. The compilation is done to assign the node name and location of the pin in the FPGA board. Connecting the FPGA and VGA monitor is done by using the specific cable and seeing the result in the monitor.

Figure 13 Running FPGA program

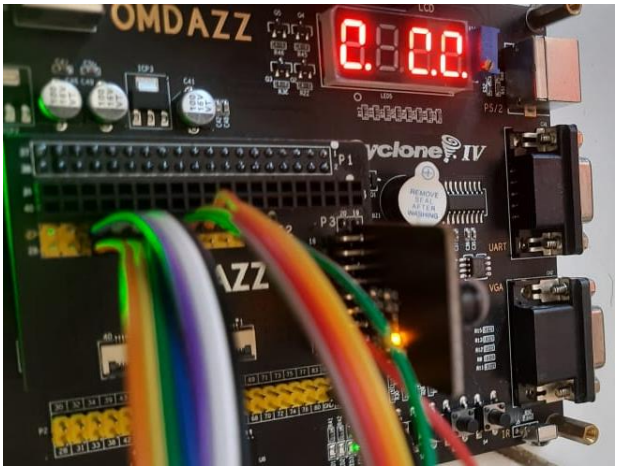

Figure 14 FPGA board implementation

4.2 Conclusion

This paper attempts to show design exploration to the real-time environment in the practice of implementing image processing algorithms in the specified FPGA board. Adapting high-level language can be implemented in both software and hardware design by the mapping process. For software operation, the requirements and specifications differ little by functionality based on software design and hardware application algorithm. Reformulating algorithms to successfully implement the design should be applied by the designer. Subsequently. MATLAB algorithm design is implemented to VHDL using HDL Coder and continuously applied to the FPGA board. FPGA board code generator is simulated in Simulink and compiled portion calculated in actual FPGA hardware While verifying functional correctness of the hardware. Simulation of design can be seen in the Simulink of MATLAB that equalized produced in the output. Another method is to implement a real-time camera as input and a VGA monitor as output.

The algorithm suits well in the implementation by using Altera Cyclone IV EP4CE10E22C8. The process

entirely depicts as the practice beginning from the block of approached image processing algorithm remaining into compilation report of VHDL implementation on the FPGA board. Testing of the algorithm by using the FPGA board can be completed to the next action to verify and validate through the finalized system. Moreover, generated VHDL code can be easily verified and mapped into FPGA by allowing the rapid prototyping of design.

AUTHORS' CONTRIBUTIONS

This paper conceived the presented practice to develop and perform the procedure in writing aspect to the manuscript for the proposed discussion. Specifically exchange of views,

- Customized image processing simulation of Histogram Equalization.
- Application of proposed algorithm into image processing FPGA board.
- Exploration of practice in FPGA board of compilation and downloading.
- Attempt method of real-time image processing method into FPGA board implementation.
- Figure the camera and monitor as input and output section of the system process.

ACKNOWLEDGMENTS

First and foremost, this paper expresses gratitude for DIPA Politeknik Negeri Bandung, Bandung State Polytechnic, Bandung, Indonesia for the financial support to complete the research that can be a part of laboratory development in practice learning.

REFERENCES

[1] Chou, C., Mohanakrishnan, S., Evans, J.: "FPGA Implementation of Digital Filters," Proc. ICSPAT, 1993.

[2] Benedetti, A., Perona, P.: "Real-time 2-D Feature Detection on a Reconfigurable Computer," Proceedings of the 1998 IEEE Conference on Computer Vision and Pattern Recognition, 1998.

[3] Johnston, C. T., K. T. Gribbon, and D. G. Bailey. "Implementing image processing algorithms on FPGAs." In Proceedings of the Eleventh Electronics New Zealand Conference, ENZCon'04, pp. 118-123. 2004.

[4] Bailey D.G. Design for embedded image processing on FPGAs. John Wiley & Sons; 2011 Jun 13.

[5] Tlelo-Cuautle, E., Carbajal-Gomez, V.H., Obeso-Rodelo, P.J., Rangel-Magdaleno, J.J. and Nunez-Perez, J.C., 2015. FPGA realization of a chaotic communication system applied to image processing. *Nonlinear Dynamics*, *82*(4), pp.1879-1892.

[6] Guragain DP, Ghimire P, Budhathoki K. Implementation of FPGA Based Image Processing Algorithm Using Xilinx System Generator. International Research Journal of Engineering and Technology (IRJET). 2018 Jan;5(01):2395-0056.

[7] Siddiqui F, Amiri S, Minhas UI, Deng T, Woods R, Rafferty K, Crookes D. Fpga-based processor acceleration for image processing applications. Journal of Imaging. 2019 Jan;5(1):16.

[8] Yadav G, Maheshwari S, Agarwal A. Contrast limited adaptive histogram equalization based enhancement for real time video system. In2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2014 Sep 24 (pp. 2392-2397). IEEE.

[9] Ishii I, Tatebe T, Gu Q, Takaki T. Color-histogram-based tracking at 2000 fps. Journal of Electronic Imaging. 2012 Mar;21(1):013010.

[10] Padilla DA, Villaverde JF, Magdaraog JJ, Oconer AJ, Ranjo JP. Vehicle and Weather Detection Using Real Time Image Processing Using Optical Flow and Color Histogram. In2019 5th International Conference on Control, Automation and Robotics (ICCAR) 2019 Apr 19 (pp. 880-883). IEEE.

[11] Mehta M, Goyal C, Srivastava MC, Jain RC. Real time object detection and tracking: Histogram matching and kalman filter approach. In2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE) 2010 Feb 26 (Vol. 5, pp. 796-801). IEEE.

[12] Jasmine J, Annadurai S. Real time video image enhancement approach using particle swarm optimisation technique with adaptive cumulative distribution function based histogram equalization. Measurement. 2019 Oct 1;145:833-40.

[13] Mathworks, Inc.: "MATLAB 5.3 Fact Sheet," Natick, MA, 1999.

[14] Mehra R, Verma R. Area efficient FPGA implementation of sobel edge detector for image processing applications. International Journal of Computer Applications. 2012 Jan 1;56(16).

[15] Raut NP, Gokhale AV. FPGA implementation for image processing algorithms using xilinx system generator. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP). 2013 May;2(4):26-36.

[16] Gonzalez, R. C., and R. E. Wood. "Digital image processing, 3rd edtn." (2007).

[17] Bailey DG. The advantages and limitations of high level synthesis for FPGA based image processing. InProceedings of the 9th International Conference on Distributed Smart Cameras 2015 Sep 8 (pp. 134-139).

[18] Stewart R, Duncan K, Michaelson G, Garcia P, Bhowmik D, Wallace A. RIPL: A Parallel Image processing language for FPGAs. ACM Transactions on Reconfigurable Technology and Systems (TRETS). 2018 Mar 14;11(1):1-24.

[19] Siddiqui F, Amiri S, Minhas UI, Deng T, Woods R, Rafferty K, Crookes D. Fpga-based processor acceleration for image processing applications. Journal of Imaging. 2019 Jan;5(1):16.

[20] Woods, R. E., Eddins, S. L., & Gonzalez, R. C. "Digital image processing using MATLAB." (2009).

[21] Memon, F., Jameel, F., Arif, M. and Memon, F.A. Model Based FPGA Design of Histogram Equalization. Sindh University Research Journal-SURJ (Science Series), 2016 Jun 14;48(2).

[22] Bailey, Donald G. "Image processing using FPGAs." (2019): 53.

[23] HajiRassouliha A, Taberner AJ, Nash MP, Nielsen PM. Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. Signal Processing: Image Communication. 2018 Oct 1;68:101-19.

[24] Altera. Intel Literature Cyclone IV. Cyclone IV Device Handbook, Volume 1. Altera Corporation. Marcg 2016.

[25] Mathwork Vision HDL Histogram. [Online]. Available: https://www.mathworks.com/help/visionhdl/ug/histogram-equalization.html