Low Cost Histogram Implementation for Image Processing using FPGA

D B Younis\textsuperscript{1} and B MK Younis\textsuperscript{2}

\textsuperscript{1}Department of Computer Engineering, Northern Technical University, Iraq
E-mail: duaa_basman@yahoo.com
\textsuperscript{2}Northern Technical University, Mosul, Iraq
E-mail: dr_bmk@ntu.edu.iq

Abstract. The solution for various applications of industrial programmed testing is image processing. In spite of this, if the contents of the image are not available in an appropriate format, even the most complex algorithms cannot extract the right information via using a histogram, the image content can be easily processed. In this paper a histogram construction hardware that it can be able to create a histogram for different types of gray scale images of different sizes, is presented which gives accurately equivalent data as derived from a histogram plot using ISE simulator and MATLAB. By taking benefit of the high-level features of the Spartan3EXC3S500E FPGA architectures and Artix-7XC7A100T, the execution time for the proposed design is 156 times faster than MATLAB time for (16×16) pixel image size in XC3S500E and it is 6 in the worst case. Also, the execution time for the proposed design is 236 times faster than MATLAB time for (16×16) pixel image size in XC7A100T and it is 8 in the worst case. The maximum frequency resulted and the maximum number of resources used in XC3S500E is 160.411 MHz, 45\% respectively. While the maximum frequency resulted and the maximum number of resources used in XC7A100T is 216.685MHz, 59\%.

Keywords: Image processing, histogram, FPGA, gray scale images.

1. Introduction

Modern moveable devices are embedding many multimedia applications and image processing applications on these moveable devices to load large calculations and large power/area. Given the battery constrained and small size of moveable devices, there is a need for a little difficulty design for image processing applications [1], [2]. Therefore, the histogram is used because it is a normally applied process and determines the performance of implementation [3].

A. Shahbahrami, et al. (2008) presented a hardware procedure for parallel histogram computation. They used dual-port memory to avoid memory collision in histogram functions. The
suggested hardware computes the histogram of an image in two phases. Firstly, arrays of the two histograms are computed for pixels of even and odd-numbered addresses, concurrently. Then the values that are stored at even- and odd-numbered addresses are utilized as indices to histogram arrays. So in half of a cycle, these pixel values are read from both input ports. In the other half of the cycle, updated values of histogram elements are stored in different histograms. Then the two histograms are added and lastly, the computed results are stored in the histogram array. The results taken by FPGA implementation have appeared that the suggested technique develops the performance compared to the highest scalar version by a factor of 1.92 and minimizes the number of cycles by a factor of 2 [4].

Yang Li, et al. (2009) present an FPGA based image histogram equalization implementation in detail which be able to meet real-time constraint very well. The main objective of this paper was to improve the speed of calculation of image histogram equalization by FPGA devices. To achieve this goal, the structure of the calculation was proposed and implemented on Xilinx Virtex4 family chip type xc4vsx35-10ff668. The simulation results indicate that for 640 * 360 image the total processing time took only about 230,915 clock cycles. Also, in the Xilinx ISE software, according to the timing report, the entire unit can achieve 200 MHz. In this case, a single 640 × 360 image only costs 1.15 milliseconds completely [5].

In September 2016 Sambaran Hazraa et al. introduced a new structure for the histogram generation, which can create a histogram for different grayscale images. The output histogram of the suggested hardware gave closely equivalent data derived from a histogram plot using simulation tool. Empirical results show that the frequency of operation of the suggested architecture is too high in the same way for FPGA using Virtex7XC7V2000T and for Programmable SoC using ZynqXC7Z030 based implementation [6].

In 2017 Harpreet Kaur and Neelofar Sohi present different applications of histograms which that help in the enhancement process. They also introduced three essential histogram processing techniques; histogram sliding, histogram stretching, and histogram equalization, and how these techniques support in the enhancement process and which factors Impact these techniques. It was found that one could choose the right technique by looking at the histogram of the image [7].

Also, Govind Bhai and Shweta Agrawal in 2017, presented four modern low complexities histogram generator algorithms. The effectiveness of the suggested algorithms has been demonstrated in the list through its implementation and simulation. Based on simulation results, the proposed algorithms have reduced implementation complexity [8].

There are various applications of histograms such as; study image, adjust the contrast of an image, image equalization, thresholding, develop the visual form of an image. Histograms reproduce a wide series of vulnerabilities for example saturation, spikes, gaps, and impacts of image compression. The form of histogram predicts information about the possibility of contrast enhancement [7, 9].

This paper is organized into five sections. Section 1, gave an introduction and the previous related researches. Section 2 includes theory, while section 3 describes a proposed hardware design then section 4 gives an analysis of results observed after implementation. Finally, section 5 ends this paper with conclusions.

2. Theory

A histogram of an image is a form that acts as a graphic representation of the distribution of tonality in a digital image [10, 11]. Furthermore, the graphical representation is a two-dimensional scheme with the integer of gray level values drawn down the x-axis and the occurrence of pixels down the y-axis. A typical gray-scale image is identified by 8 bits per pixel and its histogram includes 256 gray levels. The horizontal axis of the histogram plot symbolizing the different gray levels is partitioned into a number of slits identified as bins. It provides asimpe procedure to understand the information relating to the frequency of sample data [12] and display
how spread the data is and whether the data is placed at the correct location or not. It also shows whether the data is skewed on the way to one direction or not [13].

A digital image histogram with gray levels in the range \([0, L-1]\) is a discrete function

\[
H(r_k) = n_k 
\]

Where \(r_k\) is the kth gray level and \(n_k\) is the number of pixels in the picture having gray level \(r_k\). In order to normalize a histogram we can divide each of its values by the total number of pixels in the picture [1].

Histograms contain a large series of applications. They provide a perfect notion of the brightness contrast feature of a picture [6]. Figure 1 gives a number of images and it’s histogram to illustrate the distribution of color intensity that starts from black to white and knowledge of brightness locations and other features in the image.

![Histogram of images](image_url)

**Figure 1.** example of images and it’s histogram.

3. **Proposed Hardware Design**

The proposed architecture is applicable for all grayscale picture of different sizes (16×16, 32×32, 64×64, 128×128). This system, as shown in the block diagram of figure 2 consists of a number of blocks, start with reading an image file from the hard disk via the Matlab software, the pre-operation block, storage element block, histogram computation unit and histogram memory.

3.1 **Pre-operation block**

This block will convert the colored image to a gray level using a simple specific statement with other operations in Matlab program then the resultant image is saved into a .coe file.

3.2 **Storage element block**

In this block the resultant .coe file, from the pre-operation block, is stored in Block RAM that is generated by ISE IP core generator.

3.3 **Histogram computation unit**

The histogram computation block is responsible for generating a histogram of the image using the finite state machine technique. Seven states were used to generate the histogram values of the image as shown in figure 3. In the **first state** the address is initialized to read image data from the storage element (Block RAM memory) also gave the write enable zero value for reading operation.
from the Block RAM memory, then in the second state; the read state a counter (contains the value of the image locations) is used for address from which the image data is read. The value of the first pixel is read and this value is given to the address to the histogram memory location. The image address counter is incremented by one. After one clock it will move to third state, in this state it will read data from histogram Block RAM and increment the corresponding data by 1 then write it back in the same histogram Block RAM location after one clock. Then it will move to the fourth state where it will update histogram addresses and data and return to the second state (read state), this operation is repeated until the image address counter reach the last address in the image. At this time it will move to the fifth state (wait) for preparing to display data from histogram Block RAM and move to the sixth state; the display state to display these values using simulator, finally the last state is come to declare that the histogram for the current image is done asking for new image.

3.4 Histogram memory

This Block RAM is also created by ISE IP core generator. In this block, calculated histogram values will be stored on a dual-port block RAM memory. We have used a dual-port block RAM memory of size 256 bytes × 14 bits, which was calculated based on the size required to store the histogram values of 0 to 255. But ISE software will reserve a standard RAM block of size 16 Kbyte × 1.

Figure 2. Block Diagram for the Proposed Histogram Generator
4. Simulation and analysis result

The suggested histogram structural design is applied to a number of grayscale test images of different sizes. The test image is Lena, Xilinx ISE Design Suite 10.1 and 14.1 has been used to implement the proposed design and find the histogram computation of the Lena image as shown in figure 4. The histogram values for the first few gray levels (0 to 20) are shown in figure 5 (A) where as that of the last few gray levels (231 to 249) is shown in figure 5 (B). The resulting values of the histogram corresponding for various gray levels are accurately identical to that resulted by Matlab as shown in figure 6. The Matlab workspace results of the histogram values for the first few gray levels (1 to 20) is shown in figure 6 (A) while that for the last few gray levels (234 to 256) is shown in figure 6 (B).

Table 1 and Table 2 show the utilization resources of Spartan-3E XC3S500E and Artix-7 XC7A100T that are used to implement the proposed hardware designs. The differences between these hardware designs are when the image size is 16×16 and 32×32, the number of Block RAMs used is 2 in XC3S500E and 1 in XC7A100T while when the image size is 64×64, the number of Block RAMs used is 3 in XC3S500E and 2 in XC7A100T and in 128×128 image size, the number of Block RAMs used is 9 in XC3S500E and 5 in XC7A100T. The maximum operating frequency in XC3S500E is 160.411 MHz and the minimum operating frequency is 143.143 MHz, while the maximum operating frequency in XC7A100T is 216.685 MHz and minimum operating frequency is 200.020 MHz.
maximum speed up acquired in XC3S500E is 155.7262 when the image size is 16×16 and the 
minimum speed up acquired is 6.3609 when the image size is 128×128 but the maximum speed up 
acquired in XC7A100T is 235.729 when the image size is 16×16 and minimum speed up acquired is 
8.4313 when the image size is 128×128.

**Figure 4.** Lena grayscale test image of size (128×128) with its histogram.

(A) first 20 sample

(B) last 20 sample

**Figure 5.** ISE Functional Simulation Results using FPGA

(A) first 20 sample

(B) last 20 sample

**Figure 6.** MATLAB Results.
Table 1. Results Summary Table using SPARTAN 3E.

| Image size | 16x16 | 32x32 | 64x64 | 128x128 |
|------------|-------|-------|-------|---------|
| Resource Type | Used | Total | Ratio | Used | Total | Ratio | Used | Total | Ratio | Used | Total | Ratio |
| Number of Slices | 145 | 4656 | 3% | 169 | 4656 | 3% | 181 | 4656 | 3% | 197 | 4656 | 4% |
| Number of Slices FlipFlops | 173 | 9312 | 1% | 207 | 9312 | 2% | 225 | 9312 | 2% | 248 | 9312 | 2% |
| Number of 4 input LUTs | 220 | 9312 | 2% | 255 | 9312 | 2% | 266 | 9312 | 2% | 285 | 9312 | 3% |
| Number of Bounded IOBs | 73 | 232 | 31% | 89 | 232 | 38% | 97 | 232 | 41% | 105 | 232 | 45% |
| Number of Block RAMS | 2 | 2 | 10% | 2 | 2 | 10% | 3 | 2 | 15% | 9 | 2 | 45% |
| Number of GCLKs | 1 | 24 | 4% | 1 | 24 | 4% | 1 | 24 | 4% | 1 | 24 | 4% |
| Maximum Operating Frequency | 143.143 MHz | 153.516 MHz | 160.411 MHz | 150.898 MHz |
| Number of cycles | 1025 | 4097 | 16385 | 49154 |
| Clock period (ns) | 6.986 | 6.513 | 6.233 | 6.626 |
| Exec. Time(ns) | 7160 | 26690 | 102140 | 325740 |
| Speed up | 155.7262 | 39.3780 | 12.6590 | 6.3609 |

Table 2. Results Summary Table using ARTIX-7 board.

| Image size | 16x16 | 32x32 | 64x64 | 128x128 |
|------------|-------|-------|-------|---------|
| Resource Type | Used | Total | Ratio | Used | Total | Ratio | Used | Total | Ratio | Used | Total | Ratio |
| No. of Slice Registers | 176 | 126800 | 0% | 182 | 126800 | 0% | 196 | 126800 | 0% | 218 | 126800 | 0% |
| No. of Slice LUTs | 180 | 63400 | 0% | 186 | 63400 | 0% | 198 | 63400 | 0% | 213 | 63400 | 0% |
| No. with unused FlipFlops | 50 | 226 | 22% | 46 | 228 | 20% | 49 | 245 | 20% | 53 | 271 | 19% |
| No. with unused LUT | 46 | 226 | 20% | 42 | 228 | 18% | 47 | 245 | 19% | 58 | 271 | 21% |
| No. of fully used LUT_FF pairs | 130 | 226 | 57% | 140 | 228 | 61% | 149 | 245 | 60% | 160 | 271 | 59% |
| No. of Bounded IOBs | 81 | 210 | 38% | 89 | 210 | 42% | 97 | 210 | 46% | 105 | 210 | 50% |
| No. of Block RAMS/FIFO | 1 | 135 | 0% | 1 | 135 | 0% | 2 | 135 | 1% | 5 | 135 | 3% |
| No. of GCLKs | 1 | 32 | 3% | 1 | 32 | 3% | 1 | 32 | 3% | 1 | 32 | 3% |
| Max. Operating Frequency | 216.685 MHz | 203.707 MHz | 201.969 MHz | 200.020 MHz |
| No. cycles | 1025 | 4097 | 16385 | 49154 |
| Clock period (ns) | 4.6149 | 4.9090 | 4.9512 | 4.9995 |
| Exec. Time(ns) | 4730.36 | 20112.22 | 81126.3 | 245745.425 |
| Speed up | 235.729 | 52.262 | 15.9373 | 8.4313 |

5. Conclusion

The major function of this paper is to develop low cost histogram by using FPGA to be embedded inside any image processing system. The computation architecture above is proposed and implemented on Xilinx Spartan 3E XC3S500E and Xilinx Artix-7 XC7A100T. The reason to use two different kits is to find out the reserved space in each kit to implement a histogram that can be used to implement another algorithm that includes a histogram to enhance images in image processing. The simulation result Xilinx Artix-7 indicate that for 128x128 image the total processing time took only
about 49154 clock cycles. Also, in the Xilinx ISE software, according to the timing report, the entire unit can achieve 200.020. In this case, a single 128 ×128 image only costs 0.2457ms, while an image of size 128×128 costs 2.072 ms in MATLAB software. As well as when compared with Yang Li, et al. [5] in which the structure of the calculation was proposed and implemented on Xilinx Virtex4 family chip type xc4vsx35-10ff668. The simulation results indicate that for 640×360 image the total processing time took only about 230,915 clock cycles. Also, in the Xilinx ISE software, according to the timing report, the entire unit can achieve 200 MHz. In this case, a single 640 × 360 image only costs 1.15 milliseconds completely. We conclude that when the size of the image increases, the resources increase as shown in Table 1 and Table 2. Also notice that the ARTIX-7 XC7A100T achieved better results than Spartan 3E XC3S500E in terms of resource utilization, maximum operating frequency, the number of Block RAM used and better speed up.

6. References
1. R.C. Gonzalez and R.E. Woods2007Digital Image Processing,Prentice Hall, Third Edition.
2. A. McAndrew, “An Introduction to Digital Image Processing with MATLAB Notes for SCM2511 Image Processing 1”, School of Computer Science and Mathematics Victoria University of Technology, 2004.
3. L. Denby and C. Mallows, "Variations on the histogram", Journal of Computational and Graphical Statistics, 18(1):21–31, 2009.
4. A. Shahbahrami, J. Young Hur, B.H.H. Juurlink and S. Vassiliadis —FPGA Implementation of Parallel Histogram Computationl 2nd HiPEAC Workshop on Reconfigurable Computing, pp. 63-72, Gotenborg, Sweden, Jan 2008.
5. Y. Lia, S. Zhonga, B. Wanga, L.Yana, T. Liu," An FPGA based Histogram Equalization Implementation", Medical Imaging, Parallel Processing of Images, and Optimization Techniques. Vol. 7497, p. 74970R, International Society for Optics and Photonics, 2009.
6. S. Hazra, S. Ghosh, S. P. Maity, H.Rahaman, "A New FPGA and Programmable SoC Based VLSI Architecture for Histogram Generation of Grayscale Images for Image Processing Applications",6th International Conference On Advances In Computing & Communications, ICACC 2016, 6-8 September 2016, Cochin, India, pages 139 – 145, 2016.
7. H. Kaur and N.Sohi,"A Study for Applications of Histogram in Image Enhancement",The International Journal of Engineering and Science (IJES), Volume 6, Issue 6, Pages PP 59-63, 2017.
8. G. Bhaiand S. Agrawal,"A Novel Low Complexity Histogram Algorithm for High Performance Image Processing Applications", International Research Journal of Engineering and Technology (IRJET), Volume 04, Issue: 01, Jan -2017 , Pages 1398- 1403,2017.
9. W. Burger and M. J. Burge, "Digital Image Processing - An Algorithmic Approach using Java”, 2008 Springer-Verlag New York.

10. Sanny A, Yang YH, PrasannaVK, "Energy-efficient histogram on FPGA" In ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on 2014 Dec 8 (pp. 1-6). IEEE Cancun, Mexico.

11. Hajinoroosti, M., Grigoryan, A., & Agaian, S. “Image enhancement with weighted histogram equalization and heap transforms”, In World Automation Congress (WAC) Rio Grande, Puerto Rico, 2016 (pp. 1-6). IEEE.

12. K. S. Gautam, "Parallel Histogram Calculation for FPGA: Histogram Calculation," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 774-777

13. G. Bhaiand S. Agrawal, "A Novel Low Complexity Histogram Algorithm for High Performance Image Processing Applications”, International Research Journal of Engineering and Technology (IRJET), Volume 04, Issue: 01, Jan -2017 , Pages 1398-1403, 2017.