A Novel Low Complexity Histogram Algorithm for High Performance Image Processing Applications

Govind Bhai[1] and Shweta Agrawal[2]

[1]Research scholar, [2]Assistant Professor,
Dept. of Electronics and Comm., SRCEM Banmore, Morena, India

Abstract - *The modern portable devices requires low complexity image processing designs to achieve high performance economically. The histogram is used in modern camera to capture the image of desired brightness. To achieve high performance low cost histogram, low complexity VLSI architectures are developed. The existing architecture are not efficient, therefore this paper presents four new low complexity histogram generator algorithm. The effectiveness of the proposed algorithms is shown over the existing by implementing and simulating them. The simulation results show that the proposed algorithms show significant reduction in implementation complexity.*

Key Words **-** Image Processing, Histogram Generator, Integrated Circuits, VLSI designs.

1. INTRODUCTION

The modern portable devices are embedding several multimedia applications. The image processing applications on these portable devices demands huge computations and large power/area [1]. Due to the limited battery and small size of portable devices, low complexity design is required for image processing applications [2], [3]. Among the different task used in the image processing, the histogram is one of frequently used operation and determines the performance of the application [4], [5], [6]. Although the histogram is being used in several applications, most of the applications employ a software program to achieve it and very little efforts have been given to design a hardware architecture for the histogram generator. This paper presents the limitations of existing software level approach by implementing a VLSI architecture for of the histogram.

In this paper, various existing architectures and algorithms for histogram generation are reviewed and analyzed [7], [8], [9], [10]. Further, four new algorithms namely prop1, prop2, prop3 and prop4 are developed by approximating pixels of nearly same value to a single value. In the prop1, the pixels differ by one are approximated to one value while in prop2 the pixels differ by max of 3 value are approximated to single value. This is achieved by truncation least significant bits (LSBs). In the proposed architectures, the number of LSB bits truncated are 1-bit, 2-bit, 3-bit, and 4-bit in prop1, prop2, prop3, and prop4 design respectively to reduce the implementation complexity.

In order to evaluate the efficacy of the proposed designs, the proposed and existing designs are implemented and simulated on MATLAB and in Verilog. The histogram generated by the different histogram generator architectures are compared where the simulation results show the histogram using proposed algorithm provide same information and exhibits identical structure over the conventional design. Further, design implemented in Verilog are syntheses and post synthesis results shows that proposed architectures consume small area over the conventional histogram generator. The area in terms of LUTs register and LUT-BUF pairs is evaluated and compared. Finally, the delay metrics is also computed and compared for the different histogram architectures where the proposed architectures show small delay.

The remainder of this paper explores different histogram generator with comparative analysis by implementation and simulation.

2. HISTOGRAM GENERATOR

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image [11], [12]. A histogram of a set of data is shown in Fig. 1 where each bar representing the frequency of input data range.

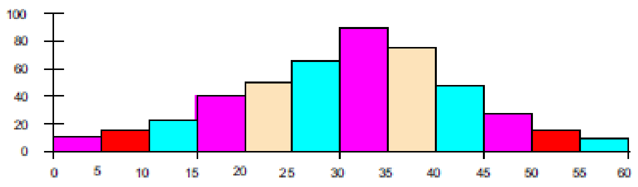


Fig-1: An illustration of histogram for data range.

The histogram provides an easy way to perceive the information regarding the frequency of sample data [13]. The histogram shows how spread/narrow the data is and whether the data is located at the proper position or not as shown in Fig. 2. It also shows whether the data is skewed toward one direction or not. Finally, some common shapes of histograms are shown in Fig. 3.

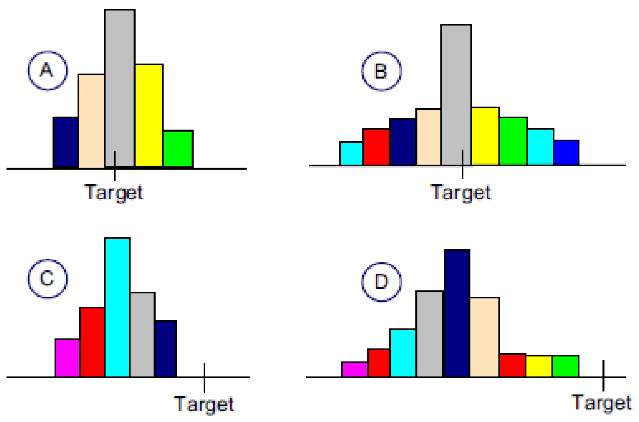


Fig-2: Histogram data spread and location.

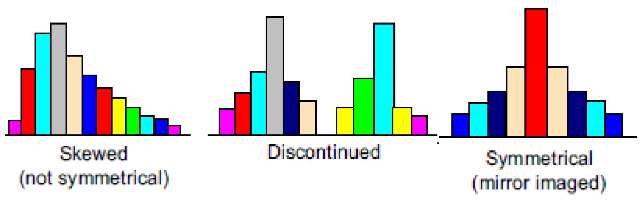


Fig-3: Common shapes of the Histograms.

3. LOW COMPLEX HISTOGRAM ALGORITHS

This section present various algorithms and architectures for low complexity histogram generation.

3.1 Programmable VLSI architecture for histogram

A programmable VLSI architecture for the histogram generator is presented in [14]. This architecture is able to compute the histograms of four free position-able window which may overlap. This architecture computes the energy, mean, maximum, minimum of the pixels which are having the value above the given threshold. These value finally determines the shape of the histogram. The histogram as presented in this paper accumulates the number of pixels of the same intensity.

3.2 Reconfigurable histogram architecture

A reconfigurable histogram architecture for the FPGA based smart cameras is proposed by Maggiani et al. in [15]. The proposed histogram generator provides low complexity of implementation and is suitable for the FPGA based designs. The proposed histogram also does not exhibit memory access conflict and employ very efficient way of parallelism. The proposed architecture can provide the histogram of input data i.e. employ the paradigm of streaming. This architecture provides improved performance over to the existing architectures. The histogram core is the prime histogram generator block and exhibits all steps of the histogram algorithm. This architecture is reconfigurable and can provide the other functionality. The prime advantage of this reconfigurable architecture is the ability to work with real-time input and providing full compatibility with the existing reconfigurable architecture.

3.3 Low Complexity histogram architecture

A programmable histogram generator that can be efficiently utilized in the image processing applications is presented in Hazra et al. [16], [17]. The proposed histogram generator architecture consists of two part where one part is responsible for generating the histogram data while other is responsible for displaying the results. The architecture exhibits 'operation selector block' which chooses kind of operation i.e. either generating the histogram or displaying. This architecture also contains ROM, decoder and array of counters. The ROM size is 216x8 used to hold the input image pixels while the number of counters are 256 as there are 256 different intensity levels. The different steps of the proposed architectures are illustrated in Fig. 4.

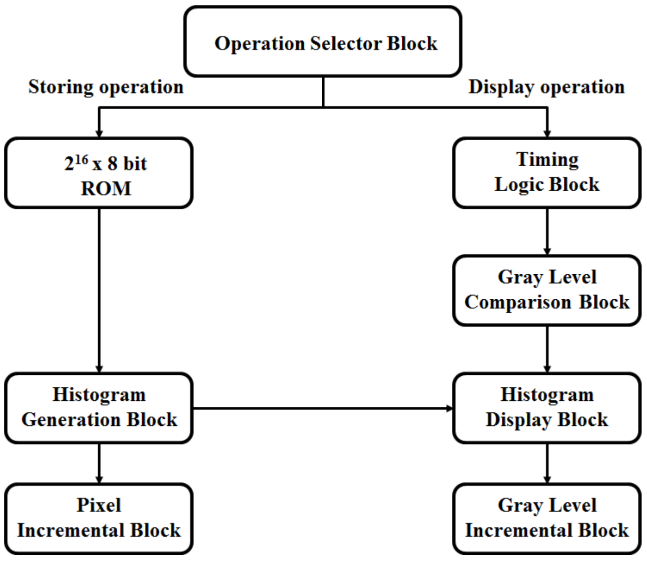


Fig-4: Various steps of histogram generator.

Since the existing algorithms and architectures for histogram still exhibit large implementation complexity, new low complexity histogram architectures are proposed and are presented in the following section.

4. PROPOSED LOW COMPLEX ALGORITHMS

The number of the registers depends on the different intensity level considered in the histogram. Since in the grayscale image there are 256 different value, it requires 256 memory registers to store the frequency of each intensity. Further, a counter is used to increase the value of each intensity once a pixel is encountered of that intensity. These counters and registers requires very large space on the IC or large number reconfigurable cells in the FPGA. In a histogram, it is observed that the histogram will provide the even if the pixels of values differed by 1 are combined i.e. the pixels which are having value with difference of one. By approximating two pixels in an image with values differ by one, it reduces the number of different pixel intensity from 256 to 128. Further, the truncation of two, three, and four least significant bits in the pixels results in 64, 32 and 32 different intensity level only. In the proposed algorithms the number of counters and registers are reduced significantly. The proposed algorithms are shown in shown below.

Algorithm1: Prop1

```
Input: image;
Output: histogram;
Initialize reg_array[1:128];
Initialize i, j, x;
[row, col] = size(image);
for i= 1: row
  for j = 1: col
    x= floor(image(i, j)/2);
    increase  reg_array(x);
  end
end
```

Algorithm2: Prop2

```
Input: image;
Output: histogram;
Initialize reg_array[1:64];
Initialize i, j, x;
[row, col] = size(image);
for i= 1: row
  for j = 1: col
    x= floor(image(i, j)/4);
    increase reg_array(x);
  end
end
```

Algorithm3: Prop3

```
Input: image;
Output: histogram;
Initialize reg_array[1:32];
Initialize i, j, x;
[row, col] = size(image);
for i= 1: row
  for j = 1: col
    x= floor(image(i, j)/8);
    increase reg_array(x);
  end
end
```

Algorithm4: Prop4

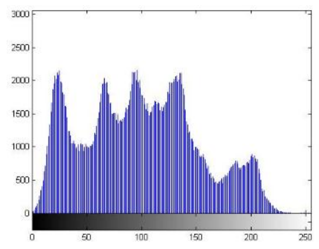
```
Input: image;
Output: histogram;
Initialize reg_array[1:16];
Initialize i, j, x;
[row, col] = size(image);
for i= 1: row
  for j = 1: col
    x= floor(image(i, j)/16);
    increase reg_array(x);
  end
end
```

The proposed algorithms prop1, prop2, prop3, and prop4 reduce the number of counters by 50%, 75%, 86.71%, and 93.75% respectively. The reduction in the number of counters and registers reduces the power and delay significantly. The histogram of the resulting image will provide the same information as that of original while significantly reducing the number of counters and registers. Therefore, the proposed algorithms can be effectively applied to reduce the implementation complexity.

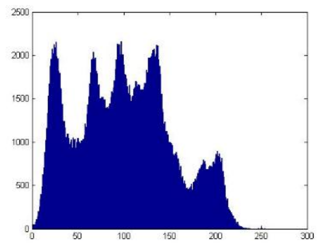
5. SIMULATION RESULT & ANALYSIS

To evaluate the efficacy of the proposed designs, all the algorithms are implemented on MATLAB and simulated with benchmark image (Lena). The histograms of the image are extracted via proposed and existing architectures. Further to achieve hardware analysis, the algorithms are implemented in Verilog and area, delay metrics are extracted.

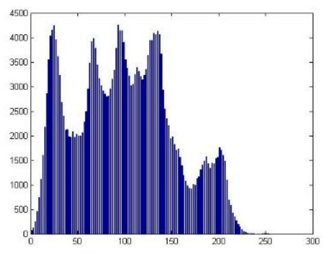
The histogram of the Lena image using MATLAB inbuilt function (imhist), conventional algorithm and proposed algorithms are shown in Fig. 5 below. It can be observed from these histograms that that the proposed algorithm exhibits nearly same algorithm while conveying same information.



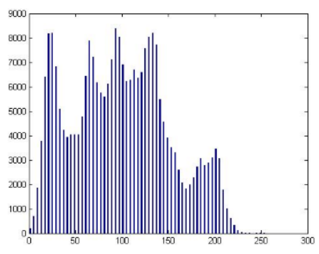
(a) imhist



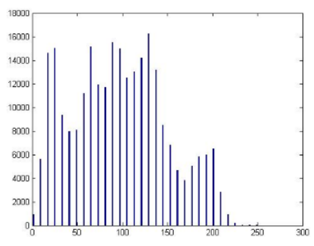
(b) conventional



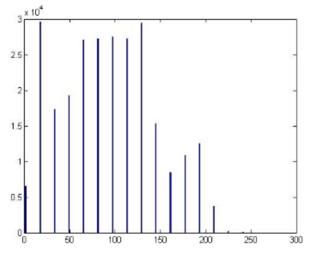
(c) prop1



(d) prop2



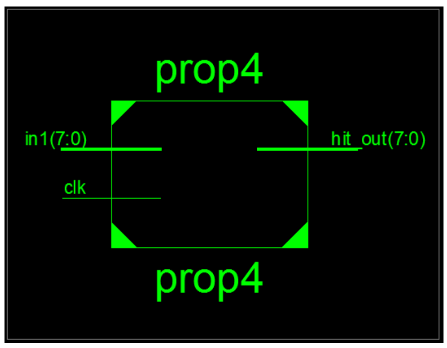
(e) prop3



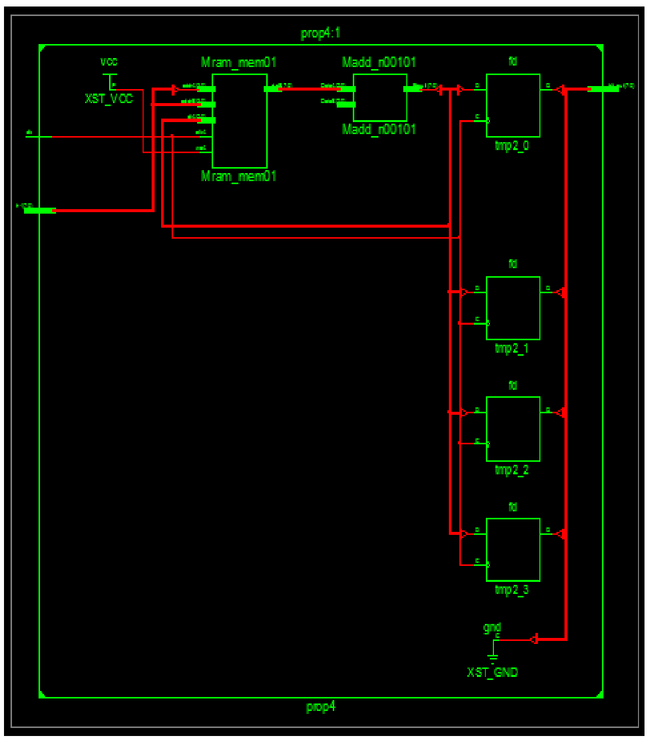
(f) prop4

Fig-5: Histograms of Lena image using various algorithms.

On the other hand, to extract the hardware analysis, all the designs are implemented in Verilog and synthesized on FPGA using Xilinx ISE CAD tool [18]. The functionality of the designs is verified. The RTL schematic from the tool for the proposed algorithm prop2 is shown in Fig. 6.



(a)



(b)

Fig-6: Proposed histograms generator a) Block diagram, b) RTL schematic.

The post-synthesis results of the different histogram generator are tabulated in Table 5.2. The area of the different histogram architectures is computed in terms of number of slice registers, slice-LUT and LUT-BUFF pairs.

Table-1: Design metrics of different histogram generator architectures.

Metrics	**Conv.**	**Prop1**	**Prop2**	**Prop3**	**Prop4**
Delay (nS)	3.585	2.732	2.472	2.13	2.124
Frequency (MHz)	278.9	365.9	404.7	469.4	470.7
#Slice Regs	8	7	6	5	4
# Slice LUTs	41	39	14	9	8
#LUT-BUF pair	8	7	6	5	0
IOBs	17	17	17	17	17

It can be observed from the Table 1 that the area required by the proposed histogram algorithms are much smaller over the conventional histogram architecture. The proposed designs prop1, prop2, prop3 and prop4 reduce the delay by 23.8%, 31.04%, 40.58%, and 40.86% respectively. The comparison of the delay for various histogram architectures are shown in Fig. 7.

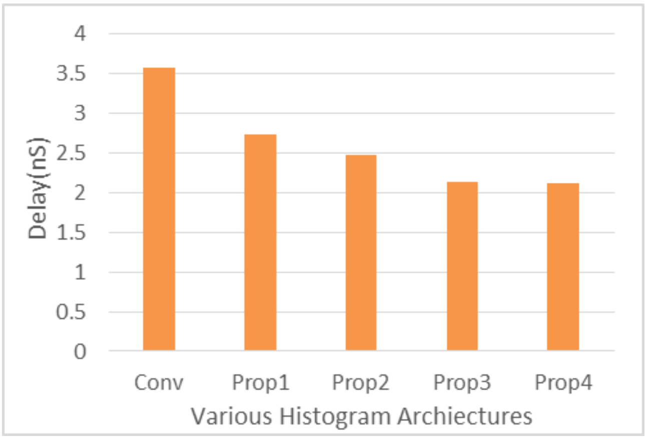


Fig-7: Delay comparison for various histogram architectures.

Further the operating frequency of the design depends on the critical path delay, higher the value of the frequency is required for the higher performance designs. The operation frequency of the implemented designs is also computed and compared as shown in Fig. 8 where it can be seen that proposed designs provides higher performance over the existing. Finally, the performance increase with more approximation in the pixel's values.

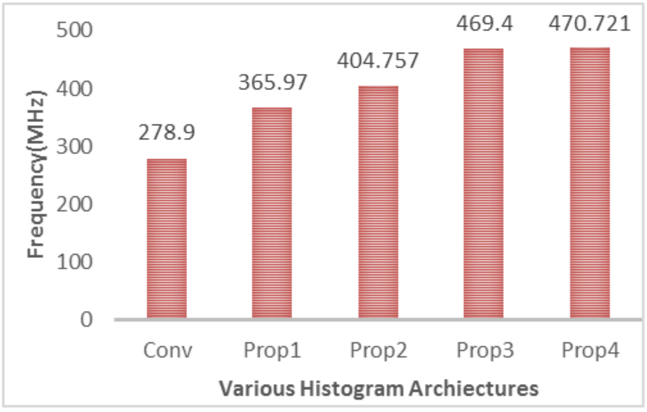


Fig-8: Performance of various histogram architectures.

The number of LUT-BUF pairs required by the different histogram architectures are compared in Fig. 9. It can be observed from the figure that the proposed design prop4 requires very small number of LUT-BUF pairs over the other architectures.

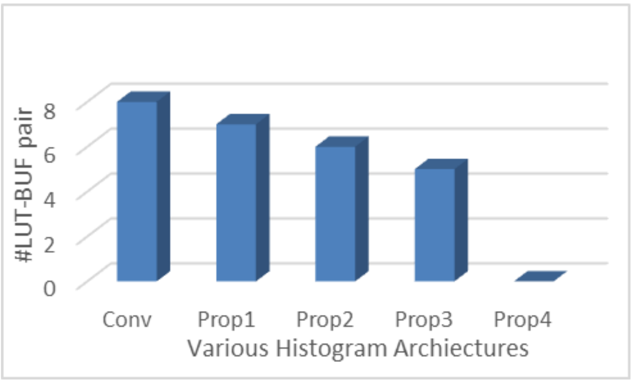


Fig-9: # LUT-BUF pairs for different histograms.

From the simulation results it is observed that proposed design requires very small implementation area over the conventional design while providing histograms that exhibit same information as that by conventional histogram generator. Therefore, the proposed architectures are very useful for the applications where implementation complexity is prime challenge.

6. CONCLUSION

The limited battery lifetime and the small size requirement of the modern portable devices requires low complexity designs. Most of these portable devices exhibits multimedia applications, area efficient designs for various image enhancement techniques are of serious concern. In this paper novel low complexity algorithms for the histogram generator are presented. The implementation complexity of the proposed algorithms very small due to reduce number of registers and counters. To evaluate the effectiveness of the different designs, all the filters architectures are implemented in MATLAB and Tanner. The designs on the MATLAB are simulated with benchmark input images and corresponding scaled image and quality metrics are extracted. The simulation results show that the proposed filters requires 12.5%, 25%, 37.5%, and 50% reduced area over the conventional histogram architecture. Moreover, the proposed designs prop1, prop2, prop3 and prop4 reduce the delay by 23.8%, 31.04%, 40.58%, and 40.86% respectively.

REFERENCES

[1]. T. Arslan, 1.M. Moreno, and A.M. Grigoryan, "New Methods of Image Enhancement," in the Proceedings of the SPIE conference on Visual Information Processing XIV, 2007.

[2]. R.C. Gonzalez and R.E. Woods, Digital Image Processing, Prentice Hall, 2nd Edition, 2001.

[3]. 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.

[4]. Lorraine Denby and Colin Mallows, "Variations on the histogram", Journal of Computational and Graphical Statistics, January 2007.

[5]. P. Skelly, S. Dixit. and M. Schwanz, "A histogram-based model for video traffic behaviour in an ATM network node with an application to congestion control." in Proceeding IEEE INFOCOM '92. Florence. Italy. 1992.pp. 95-104.

[6]. Huaifeng Zhang, Xiangjian He, Qiang Wu, "Refined Gaussian Weighted Histogram Intersection and Its Application in Number Plate Categorization," CGIV 2006, pp.249-255.

[7]. Muller, Steffen. "A new programmable VLSI architecture for histogram and statistics computation in different windows." Image Processing, 1995. Proceedings., International Conference on. Vol. 1. IEEE, 1995.

[8]. Cadenas, J., R. Simon Sherratt, and P. Huerta. "Parallel pipelined histogram architectures." Electronics letters 47.20 (2011): 1118-1120.

[9]. Gan, Q., J. M. P. Langlois, and Y. Savaria. "Parallel array histogram architecture for embedded implementations." Electronics Letters 49.2 (2013): 99-101.

[10]. Shahbahrami, A., Hur, J.Y., Juulink, B., and Wong, S.: 'FPGA implementation of parallel histogram computation'. 2nd HiPEAC Workshop on Reconfigurable Computing, Goteborg, Sweden, 2008, pp. 63–72.

[11]. Sanny A, Yang YH, Prasanna VK. Energy-efficient histogram on FPGA. InReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on 2014 Dec 8 (pp. 1-6). IEEE.

[12]. Hajinoroozi, M., Grigoryan, A., & Agaian, S. "Image enhancement with weighted histogram equalization and heap transforms", In World Automation Congress (WAC), 2016 (pp. 1-6). IEEE.

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

[14]. Yin, S., Ouyang, P., Chen, T., Liu, L., & Wei, S. "A Configurable Parallel Hardware Architecture for Efficient Integral Histogram Image Computing". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vo. 24, No. 4, pp. 1305-1318.

[15]. Maggiani, L., Salvadori, C., Petracca, M., Pagano, P., & Saletti, R. (2014, June). Reconfigurable architecture for computing histograms in real-time tailored to FPGA-based smart camera. In Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on (pp. 1042-1046). IEEE.

[16]. Ghosh, S., Hazra, S., Maity, S. P., & Rahaman, H. "A new algorithm for grayscale image histogram computation", In India Conference (INDICON), 2015 Annual IEEE (pp. 1-6). IEEE.

[17]. Hazra, S., Ghosh, S., Maity, S. P., & Rahaman, H. "A New FPGA and Programmable SoC Based VLSI Architecture for Histogram Generation of Grayscale Images for Image Processing Applications", Procedia Computer Science, Vol. 93, pp. 139-145.

ModelSim, "ModelSim HDL simulator," http://www.mentor.com/products/fpga/model, 2013.