International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

An High Speed FPGA Implementation of Image Contrast Enhancement using Histogram Equalisation

Agalya P.[1], M. C. Hanumantharaju[2]

Submitted: 15/01/2024 **Revised**: 23/02/2024 **Accepted**: 01/03/2024

Abstract: Image enhancement is one of the basic and important image processing techniques used to improve the quality of image in wide variety of applications including real-time video surveillance, medical imaging, industrial automation, intelligent self navigation systems, and oceanography. This research paper proposes a novel high-speed parallel and pipelined reconfigurable field programmable gate arrays (FPGA) architecture for histogram equalisation to enhance the contrast of an image. The proposed parallel histogram equalisation architecture comprises of comparators, unique counter modules and a register array. The proposed architecture has been developed using register transfer level (RTL) - compliant Verilog HDL code, simulated using Xilinx's integrated simulator (ISim), synthesized and implemented on a Kintex7 family of FPGA device. Simulation and synthesis results demonstrate that the proposed architecture can be implemented with a processing time of 2.364ns with a maximum frequency of operation 422.976MHz. Extremely dark images and overexposed images from standard datasets have been used to test the performance of the developed architecture and compared with matlab results through the quality metrics like Entropy, Contrast per pixel. Results obtained by proposed architecture and the results obtained by matlab found similar. Experimental results show that the proposed architecture outperforms other existing architectures

Keywords: *Contrast enhancement, FPGA, HE, High speed, Pipelined Architecture*

1. Introduction

Histogram equalization is the simplest and broadly applied image processing technique[1] used to enhance the contrast of an image by redistributing its pixel intensities to the entire range. The main objective of histogram equalization is to make the pixel intensity distribution more uniform across the entire intensity range, thereby improving the overall visual quality and enhancing features in the image.

Histogram computation and pixel transformation function are the two major operations in histogram equalisation technique. Serializing the histogram computation increases the run time, but whereas parallelizing the data for histogram computation reduces the computation time to a larger extent. Parallel computation of histogram equalization can be achieved using FPGA technology to accelerate the process and improve its efficiency. Histogram equalization is an image processing technique used to enhance the contrast of an image by redistributing pixel intensities. Parallel processing on an FPGA can significantly speed up the computation of the histogram equalization algorithm for images.

Histogram computation and histogram mapping are the two major sub modules of histogram equalisation. Many researchers have proposed various architectures for histogram computation. Two different approaches have been used for the computation of histogram. One is memory based approach and the other is counter based approach. In memory based histogram computation BRAMs are used for two purposes

(i) To store image pixel values
(ii) To store histogram for each bin

By streaming the image pixel values into the computation unit instead of storing in BRAM may use less number of BRAMs.

Parallelizing the histogram computation using memory based architecture has a challenging problem. If the group of pixels that are read parallel from memory has several occurrences of same pixel value, then there are several writes to the same memory location at the same time which leads to memory collision. A sample to illustrate memory collision is shown in figure1. Another challenging problem in BRAM based histogram computation is that increase in image size demands large number of BRAMS.

1Department of Electronics & Communication Engineering
Sapthagiri College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi, India
agalyas@sapthagiri.edu.in
2 Department of Electronics & Communication Engineering
BMS Institute of Technology & Management, Bangalore, Visvesvaraya Technological University, Belagavi, India
mchanumantharaju@gmail.com

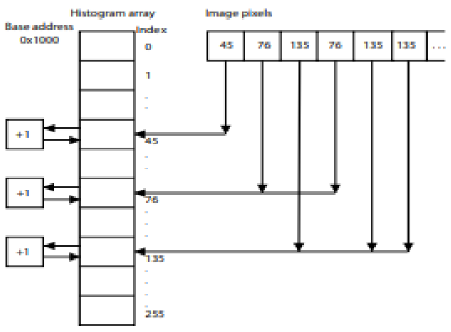


Fig 1: Memory collision in parallel histogram computation [2]

The proposed research work is focused on developing a novel parallel and pipelined counter based histogram computation architecture to overcome the above said challenges. In counter based approach the number of counters to count the bins remains 256 only irrespective of the size of the input image and only the counter bit size increases with increase in image size. The proposed design demands only the increase in counter register size instead of the number of counters for larger image size.

The major contributions of the proposed research work are

(i) Development of an efficient parallel and pipelined architecture for histogram computation of an image
(ii) Use of Register Transfer Level modeling style of verilog HDL coding for the development of all sub modules of histogram equalisation architecture.
(iii) A novel counter design to count when the two consecutive pixels of an image holds the same pixel intensity value which are read parallel from odd and even address.
(iv) Efficient high speed design architecture to implement output pixel transformation function.

2. Related Work

The huge challenge in hardware implementation of histogram equalization algorithm is to reduce the processing time by utilizing fewer hardware resources. Histogram computation and pixel transformation are the two major steps in histogram equalisation. Many researchers have come out with various architectures for histogram computation. The two most convenient architectures proposed in the literature for computing histograms use an array of counters [2] and array of accumulators with memory [3]. In Memory based architectures, every pixel histogram computation needs one read and write memory access before and after computation. This kind of memory access time dominates the actual computation which eventually increases the processing time.

A sequential histogram computation architecture using an array of 256×16-bit memory has been proposed by Li et al.[3]. Each memory location is used to update and store the histogram count of the respective greyscale bins. This architecture needs 3 clock cycles to read and update the histogram of an input greyscale value. Use of this sequential architecture may demand additional hardware for the computation of cumulative histogram and equalised output greyscale value which may reduce the overall speed of histogram equalisation architecture.

A decoder based sequential architecture that uses an array of counters has been proposed by Abdullah et al.[4] and Nitin et al. [5]. In this architecture the author use a special kind of decoder called hierarchical decoder which has a cascaded OR gate structure for computation of cumulative histogram. But this cascaded OR gate structure introduces additional delay at the higher order outputs of decoder which limits the speed of operation of the system.

Parallel and pipelined array architecture for real-time histogram computation has been proposed by Cadenas et. al. [6]. In proposed method a set of processing cells to compute k histogram bins for 1 or 2 or 4 input pixels in pipelined fashion. Finally the histogram bin counts are read serially from all processing cells. Registers are used to accumulate the histogram bins. Proposed system demands more number of processing cells if the processing cells accommodate less number of histogram bins.

Asadollah Shahbahrami et al. [2] proposed a parallel histogram computation architecture that overcome the memory collision problem by using two port memory. The architecture computes the histogram in two phases by using three memories. In first phase it uses two memories to compute the histogram of pixels stored in odd and even memory addresses individually and in second phase it adds the histograms stored in both memories and store it in third memory. Second phase incurs an additional 256 clock cycles in computing the final histogram along with N2/2 clock cycles used in first phase for an image of size N×N. Proposed architecture is found to be efficient for smaller histograms but fails for larger histograms as it demands additional memory.

Krishna Swaroop Gautam [7] proposed a parallel architecture for histogram computation of n pixels simultaneously. The proposed architecture use n input registers, demultiplexers and adders. Proposed parallel architecture reduces the computation time by n as compared to the sequential computation architecture at the same time it increases the hardware resources usage by n times.

Parallel array histogram architecture proposed by Gan et.al [8] uses a register array instead of a memory array to store the histogram values. In each step, M inputs can be processed in parallel to renovate the histogram bins without any additional dormancy. Experimental results show that the design can achieve a super-linear speed-up of 43.75 for a 16- way PAHA when compared to a software implementation in a general purpose processor but however the proposed design demands more hardware resources.

Ernest Jamro, Maciej Wielgosz, and Kazimierz Wiatr in 2007[9] has proposed a highly parallel architecture module for histogram computation. Proposed architecture includes a Dual port BRAM module that could process multiple input data in a single clock cycle. The fundamental disadvantage arises from this design is that the number of BRAMs which is required in total is the product of BRAM for parallel histogram calculations (or LUT programming) and the reading of the histogram. This increase in BRAM requirement limits the amount of parallelism adopted in the design.

Riyadh Zaghlool Mahmood et. al. [10] has proposed a hardware architecture for parallel histogram computation that avoids memory collisions by using a dual-ported memory. The proposed architecture uses two BRAM's, one BRAM to store the image pixels and another BRAM to store histogram array. In proposed method the input pixels are used as addresses of BRAM to update the histogram array. The system architecture uses 3 clock cycles to update the histogram array of two pixels parallel. Proposed system needs 24756 (6000h) clock cycles to compute the histogram of 128×128 image. Proposed Architecture demands additional BRAM's for larger image sizes with higher histogram size.

The speed at which histogram statistics are computed is increased by each of these parallel histogram computing architectures at the expense of higher hardware resource utilization. At the same time most of the existing work is addressing only the architecture and processing time for computation of histogram, but the additional processing time and resources needed for computing histogram equalised image has not been addressed.

3. Proposed Work:

Proposed research work supports massive parallelism in computing image histogram using a set of comparators and counters with an additional control circuit by reading two pixels at the same time. Flowchart for the proposed HE architecture is shown in figure 2.

1. Read two pixels at a time one from odd numbered address and another from even numbered address from memory
2. Compare the read pixels with all 256 bin values simultaneously using 256 × 2 comparators.
3. The comparator output for odd addressed pixel (pixel_in1) is c1n and the comparator output for even addressed pixel (pixel_in2) is c2n.
4. Check the value of c1n and c2n for controlling the counter operation.
5. If both c1n and c2n are equal to 0, then the counter value will be retained as it is. If either c1n or c2n is equal to 1 then the counter will update its counter value by 1. If both c1n and c2n are equal to 1, the counter updates its count value by 2.
6. Repeat the steps 1 to 5 until all the pixels in an image are read.
7. The most significant 8 bits of updated counter value is loaded into a 256 × 8 bit array registers which holds the histogram equalized pixel values for all 256 bins.
8. Output the two histogram equalized pixel values at each clock.

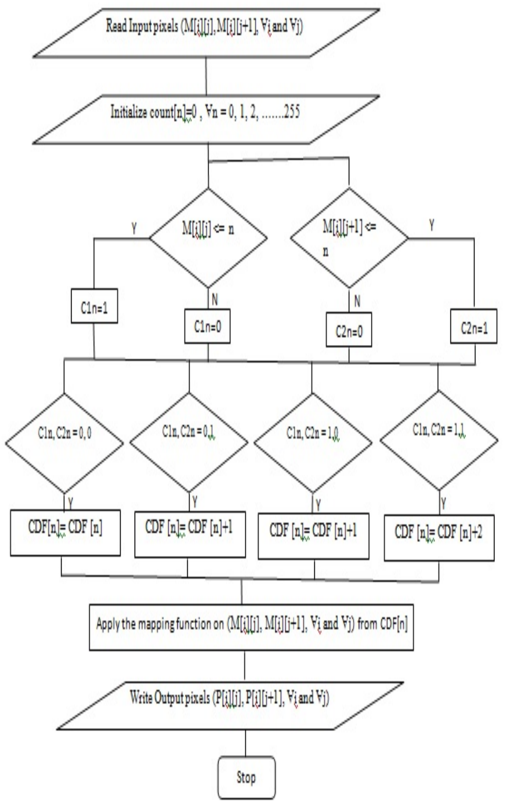

Read Input pixels (M[i][j], M[i][j+1], ∀i and ∀j)
Initialize count[n]=0 , ∀n = 0, 1, 2, ...255
M[i][j] <= n
M[i][j+1] <= n
Y
Y
N
N
C1n=1
C1n=0
C2n=0
C2n=1
C1n, C2n = 0, 0
C1n, C2n = 0,1
C1n, C2n = 1,0
C1n, C2n = 1,1
Y
Y
Y
Y
CDF[n]= CDF [n]
CDF [n]= CDF [n]+1
CDF [n]= CDF [n]+1
CDF [n]= CDF [n]+2
Apply the mapping function on (M[i][j], M[i][j+1], ∀i and ∀j) from CDF[n]
Write Output pixels (P[i][j], P[i][j+1], ∀i and ∀j)
Stop

Fig 2: Flowchart of Proposed PHE Architecture

3.1 Numerical Illustration:

Traditional HE Computation

Consider a 3 bit image (M) of size 4×4

$$M = \begin{bmatrix} 0\ 1\ 1\ 3 \\ 7\ 2\ 5\ 6 \\ 6\ 3\ 2\ 1 \\ 1\ 4\ 4\ 2 \end{bmatrix}$$

(i) Find the range of intensity values of the image
Range of pixel intensity values for a 3 bit image (k_{max}) is 7

(ii) Compute the frequency of occurrence (n(k):count) of each pixel (k) in the image.

k:	0	1	2	3	4	5	6	7
n(k):	1	4	3	2	2	1	2	1

(iii) Compute the cumulative count (n(r)) of each pixel in the image using equation 1.

$$\mathbf{n(r)}=\sum_{k=0}^{r} \boldsymbol{n(k)} \qquad (1)$$

n(r):	1	5	8	10	12	13	15	16

(iv) Compute the Histogram equalised output pixel values of image using equation 2.

$$T(r)=k_{max}\times\frac{\sum_{k=0}^{r} n(k)}{N} \qquad (2)$$

Where T(r) is the histogram equalised output pixel value for the pixel intensity r. Substituting kmax=7 and N=16 in equation 2

r:	0	1	2	3	4	5	6	7
T(r)	:0	2	4	4	5	6	6	7

Histogram equalised output image (M_{he}) obtained is shown as

$$M_{he} = \begin{bmatrix} 02\ 2\ 4 \\ 7\ 4\ 6\ 6 \\ 6\ 4\ 4\ 2 \\ 2\ 55\ 4 \end{bmatrix}$$

Proposed HE Computation

Consider a 3 bit image (M) of size 4×4

$$M = \begin{bmatrix} 0\ 1\ 1\ 3 \\ 7\ 2\ 5\ 6 \\ 6\ 3\ 2\ 1 \\ 1\ 4\ 4\ 2 \end{bmatrix}$$

(i) Find the range of intensity values of the image
Range of pixel intensity values for a 3 bit image (k_{max}) is 7

(ii) Compute the frequency of occurrence (n(k):count) of each pixel (k) in the image.

k:	0	1	2	3	4	5	6	7
n(k):	1	4	3	2	2	1	2	1

(iii) Compute the cumulative count (n(r)) of each pixel in the image using equation 1.

$$\mathbf{n(r)} = \sum_{k=0}^{r} \boldsymbol{n(k)}$$

n(r)	1	5	8	10	12	13	15	16

(iv) Compute the Histogram equalised output pixel values of image.

Equation (2) reduces to equation (3) by substituting kmax=8 and N=16

$$T(r)=8\times\frac{\sum_{k=0}^{r} n(k)}{16}=\frac{\sum_{k=0}^{r} n(k)}{2} \qquad (3)$$

According to equation (3), the histogram equalised output pixel values are obtained by right shifting the cumulative count n(r) by 1 bit which reduces the complex operations of multiplication and division in traditional approach.

r:	0	1	2	3	4	5	6	7
T(r)	0	2	4	5	6	6	7	7

Histogram equalised output image obtained using proposed method is shown as

$$M_{he} = \begin{bmatrix} 02\ 2\ 5 \\ 7\ 4\ 6\ 7 \\ 7\ 5\ 4\ 2 \\ 2\ 6\ 6\ 4 \end{bmatrix}$$

Numerical illustration of proposed method shows that the output pixel valued obtained from proposed method is same as the traditional method.

3.2 Architecture for Proposed PHE System

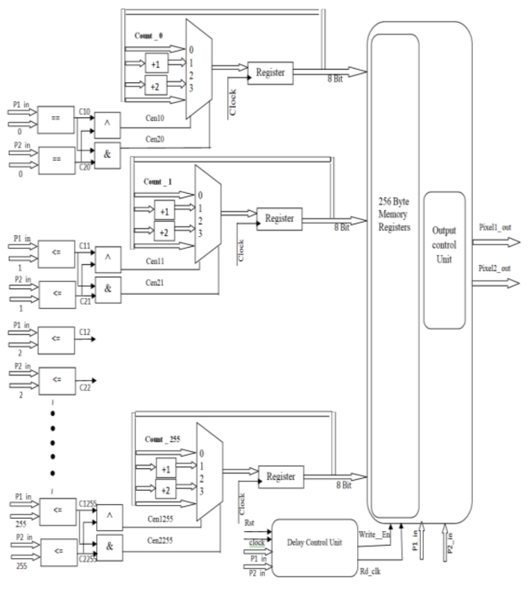


Fig 3: Proposed PHE Architecture

The proposed FPGA architecture for the parallel histogram equalization computation is shown in Figure 3. The proposed PHE architecture simultaneously reads two input pixels from two consecutive memory addresses for every positive edge of the clock signal. The proposed system's architecture consists of histogram computing unit and a pixel transformation unit. The histogram computation unit includes an array of comparators, an array of control modules and an array of registers. A register array and a delay control unit are part of the pixel transformation unit. The histogram count values are held in a set of counters, the histogram equalized pixel values are stored in a set of registers. The proposed parallel architecture speeds up the computation of histogram equalization without requiring a substantial increase in device cost.

3.2.1 Comparator Module

Comparator module in the Proposed design uses a set of two comparators of less than or equal comparison type as shown in figure 4.

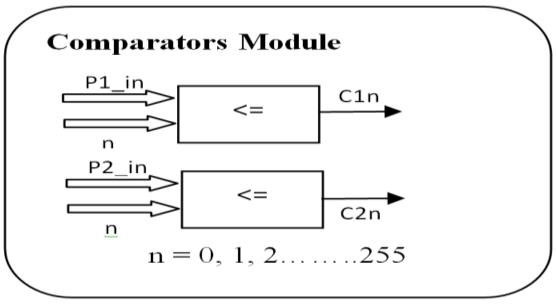


Fig 4 Architecture of Comparator Module

This comparator module compare a specific grayscale value in the range of 0 to 255 with two input pixels from two consecutive memory addresses and generates the output signals C1n and C2n. The comparator outputs C1n and C2n are set to 1 when the input pixels p1_in and p2_in are less than or equal to the pixel intensity n. A set of AND, OR and XNOR gates are used to compute less than or equal to operation of comparator. This comparator module outputs C1n and C2n are used as the control signals to trigger the counter enable signals as shown in figure 5.

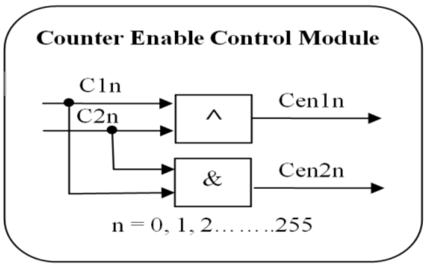


Fig 5 Counter Enable Control Circuit

Counter Enable control circuit provides the control signals to the counter registers to update the histogram value. The module consists of a set of logical XOR and logical AND gates. XOR gate is used to trigger the nth counter when any one pixel out of 2 input pixels is less than or equal to the intensity value n. AND gate is used to trigger nth counter when both the input pixels are less than or equal to the intensity value n.

3.2.2 Histogram Count Module

High speed design architecture proposed for histogram count module which includes 2 adders, a 4:1 multiplexer and an 8 bit register has been shown in figure 6. Adders increment the count values and store the updated count into counter register in every positive edge of the clock. Proposed design employs 256 counter modules to update the histogram count of all 256 grayscale bins. The cumulative histogram count is incremented by 1 when any one of input pixel is less than or equal to the pixel intensity n and it is incremented by 2 when both the consecutive input pixels are less than or equal to the pixel intensity n. The counter design incorporates two adders (Adder1 and Adder2), a multiplexer, and a register. Adder1 and Adder2 add the constants 1 and 2 to it, accordingly when their input (prior count) changes. Multiplexer picks the results from Adder1 or Adder2 based on the control signals cen1n and cen2n. When cen1n=1 and cen2n=0, the multiplexer provides the Adder1 result (count_n = count_n+1), and when cen1n=0 and cenn=1, it provides the Adder2 result (count_n=count_n+2).On each positive clock edge, the multiplexer output (cumulative pixel count) is loaded into the register. Proposed PHE system architecture uses 256 counters to hold the cumulative count of all pixel intensity values in an image.

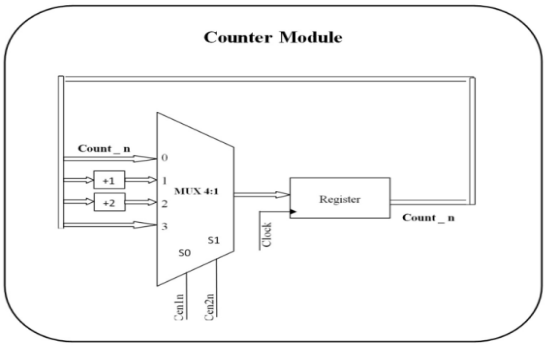


Fig 6 Counter Design

3.2.3 Delay control unit

Delay control unit acts as an interface between histogram count module and output mapping module. In order to write the histogram equalised pixel values to the output register array, there is a need of read and write control signal with a delayed clock. The control signals used to read (Rd_clk) and write (Write_En) the output registers are generated by the delay control unit. Each input pixel is mapped with its corresponding histogram equalised output pixel value from 256 byte register array with appropriate delay. The block diagram and detailed architecture of delay control unit is shown in figure 7. Delay control signals Write_En and Rd_clk are generated after n/2 (n: total number of pixels in an input image) and n/2+1 clock cycles respectively for an image of size n.

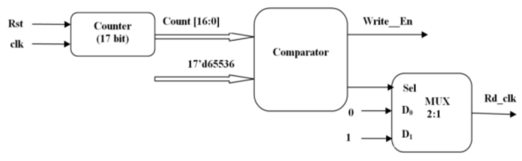


Fig 7 Architecture of Delay Control Circuit

3.2.4 Output Mapping Unit:

A set of 256 output registers are used to store the computed histogram equalized output pixel values. Least significant 8 bits are stored as histogram equalised output pixels when write control signal (write_en) is enabled. Histogram equalized output pixels are read from the output register array when Rd_clk control signal is set by the delay control unit. The input pixels which are delayed for n+1 clock cycles are used as index values for accessing appropriate registers to read the output pixel values.

4. Experimental Results

Architecture design to implement histogram equalisation algorithm onto an FPGA has been coded in matlab to ensure the correctness of the developed hardware architecture. Subsequently the proposed architecture has been developed using RTL compliant verilog HDL (Hardware Description Language) code so that it can be implemented on an FPGA.

The pixel intensities (hexadecimal value) of a standard image in .jpg and .bmp format are read and stored it in a text (.txt format) file. Different variety of dark images, over exposed images, aerial images and medical images from different datasets [11] are used to test the functionality of the propose architecture. A size of 256×256 and 512×512 Grayscale scale and RGB color images have been used in testing the design architecture. The text file generated by the Matlab code has been used by Xilinx ISIM (Ver. SE 14.7) for simulation.

The simulated output of the proposed design has been written in an output text (.txt format) file. To display the histogram-equalized output image, a Matlab program has been written to read the pixel intensities from the output text file and convert them back into an output image in the.jpg (.jpg format) file.

4.1 Simulation Results

The Simulation results of histogram equalization architecture shown in figure 8 portrays that two input pixels are read in each positive edge of the clock. After a latency of 32768 clock cycles histogram equalized output pixels are generated for two input pixels in each positive edge of clock. The histogram equalized output pixel values obtained for the two input pixels 201 and 198 respectively at the positive edge of 32768th clock has been shown in figure 8.

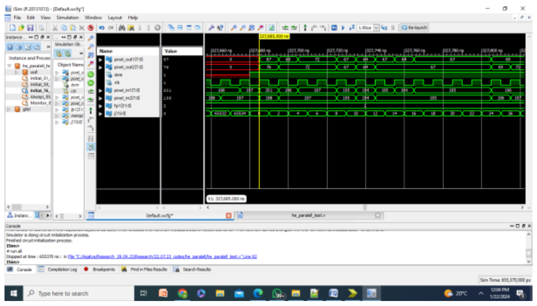


Fig 8 Simulation Results of Proposed System

Developed RTL code for PHE architecture has been tested for RGB color images and the simulation results are shown in figure 9. Two pixels from each RGB color channel are read and processed in one clock cycle in the proposed design.

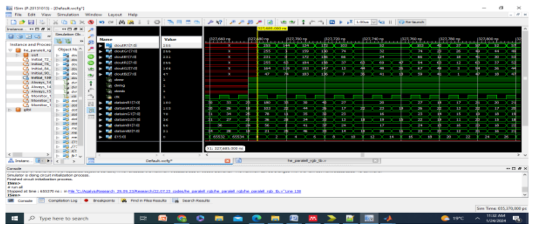


Fig 9 Simulation Results of Proposed PHE Architecture for Color Images

4.2 Synthesis Results

The proposed PHE architecture has been synthesized and implemented on Kintex 7 family FPGA board. After synthesis and implementation, it is relatively simple to examine and analyze the various metrics, like area, power, and timing requirement. The RTL schematic of comparator and counter control module for the pixel input values from 194 to 200 has been shown in figure 10 and the RTL schematic of counter and output pixel mapping unit has been shown in figure 11.

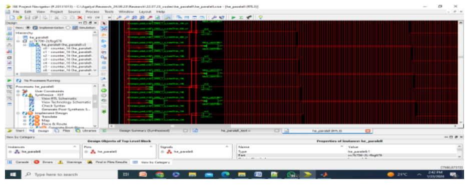


Fig 10 RTL Schematic of proposed PHE Architecture (Input Side)

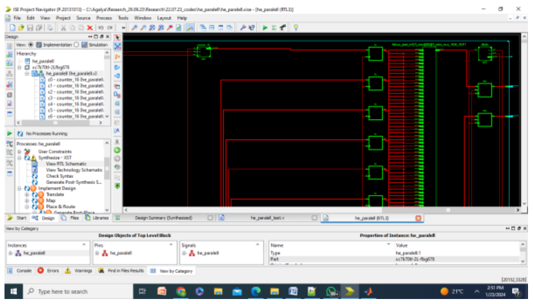
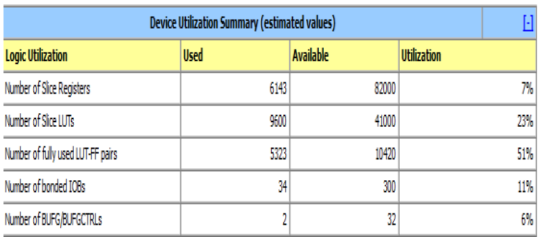


Fig 11 RTL Schematic of proposed PHE Architecture (Output Side)

The summary of utilization of slices and LUTs of Kintex7 FPGA for the proposed PHE architecture has been shown in figure 12. Utilization summary shows that the design uses 7% of slice registers, 23% of slice LUTs for implementing the combinational logic of the design and 50% of available LUT-FF pairs as memory elements for the proposed PHE design. Synthesized design uses 2064 D-type flip-flop(s), 507 comparator(s), 2 multiplexer(s) and 257 adders/subtractors



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	6143	82000	7%
Number of Slice LUTs	9600	41000	23%
Number of fully used LUT-FF pairs	5323	10420	51%
Number of bonded IOBs	34	300	11%
Number of BUFG/BUFGCTRLs	2	32	6%

Fig 12 Device Utilisation of Proposed PHE Architecture

4.3 Experimental Results Analysis

The research work focus on developing efficient high speed architecture for histogram equalisation technique which is one of the prominent image contrast enhancement technique. According to the timing report, the design's processing time is 2.596ns with a maximum frequency of operation of 385.26MHz when only parallelism is used, and 2.364ns with a maximum frequency of operation of 422.976MHz when both parallelism and pipelining are used. The proposed design has a latency of 32768 clock cycle for a 256×256 image size.

Table 1 Comparison of Processing time of Proposed PHE Architecture

Image Size	Processing time in [12]	Processing time in [4][5]	Processing time in [13]	Processing time of Proposed Architecture
256×256	0.97 ms	0.263 ms	0.15ms	0.077ms

The proposed architecture has been compared with various existing architectures [4], [5], [12], [13] in terms of processing time and the results are listed in table1for a standard image of size 256×256.

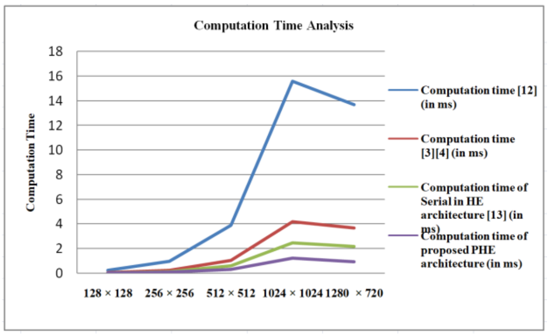


Fig 13 Comparison of Computation Time

Experimental results show that the proposed PHE architecture enhances the contrast of an image with a higher speed compared to the architectures proposed in [4], [5], [12], [13]. According to the comparison shown in table 1 proposed PHE architecture has speed up factor of 2, 3.14 and 12.5 compared to the serial in architecture proposed by [13] , [4,5] and [12]. Computation time of various architectures for different image size is shown in figure 13.

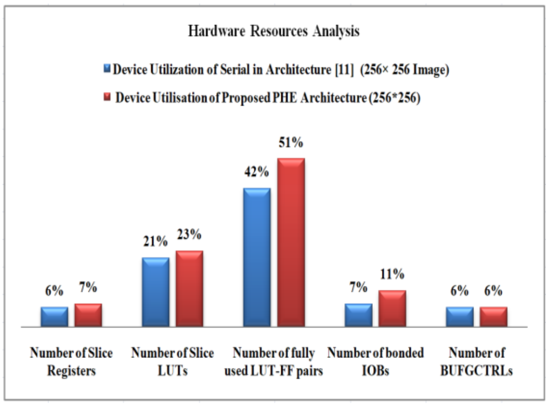
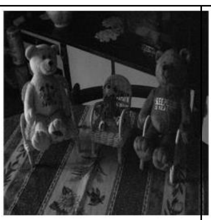
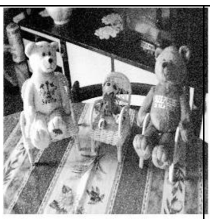
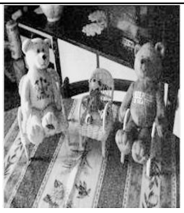


Fig 14 Comparison of Hardware Resource Utilisation

Figure 14 illustrates that the proposed PHE architecture utilizes only 8% more hardware resources compared to serial computation architecture proposed in [13] by increasing the speed by a factor of 2.

		
(A1) (001.jpg)	(B1)	(C1)

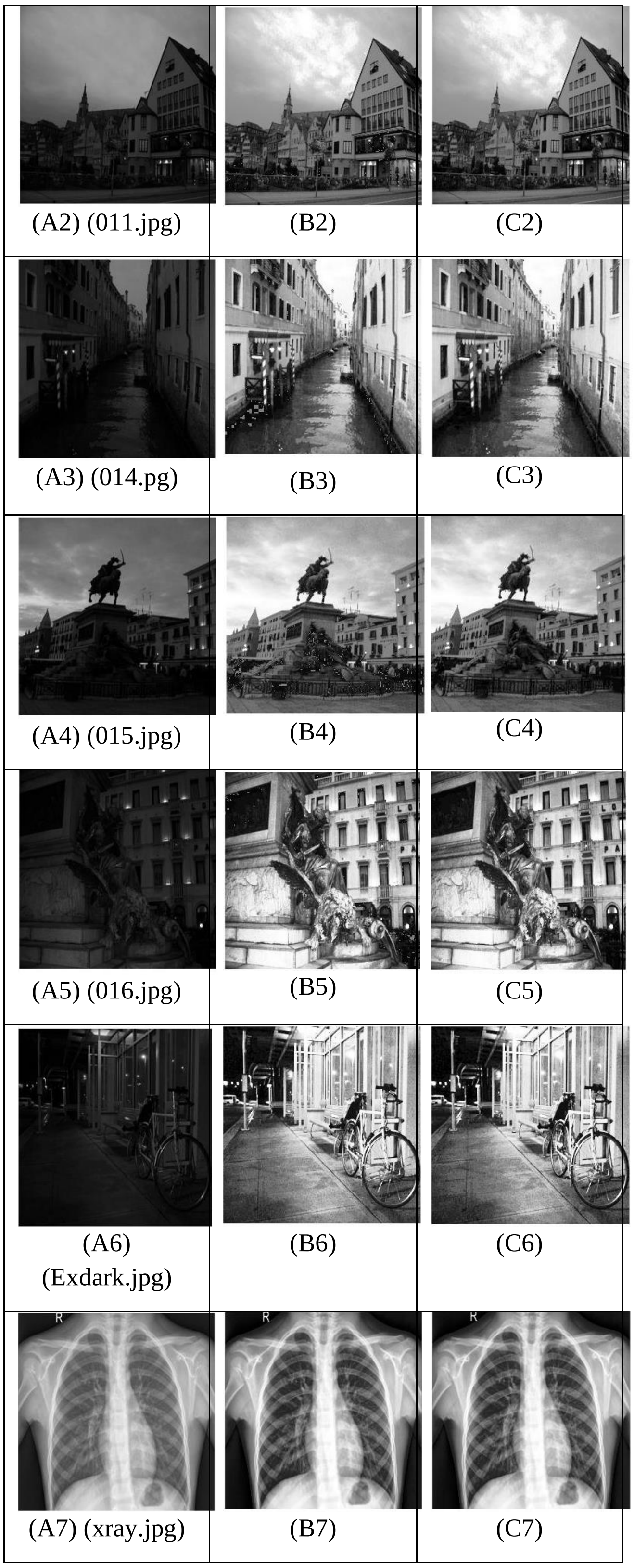


Fig 15 Input and Output images of Proposed Design

A set of dark image samples from the dataset [10] have been used to test the performance of the proposed architecture. First the color images are converted into grayscale images and given as an input to the proposed architecture. Columns 1, 2, and 3 of Figure 15 display the input grayscale and the corresponding output images generated by the Matlab and FPGA implementations. Figure 16 displays the results of processing the R, G, and B channels of the color images using the proposed architecture. The results show that the output images produced by the proposed architecture have better visual quality compared to the input images in terms of brightness and contrast.

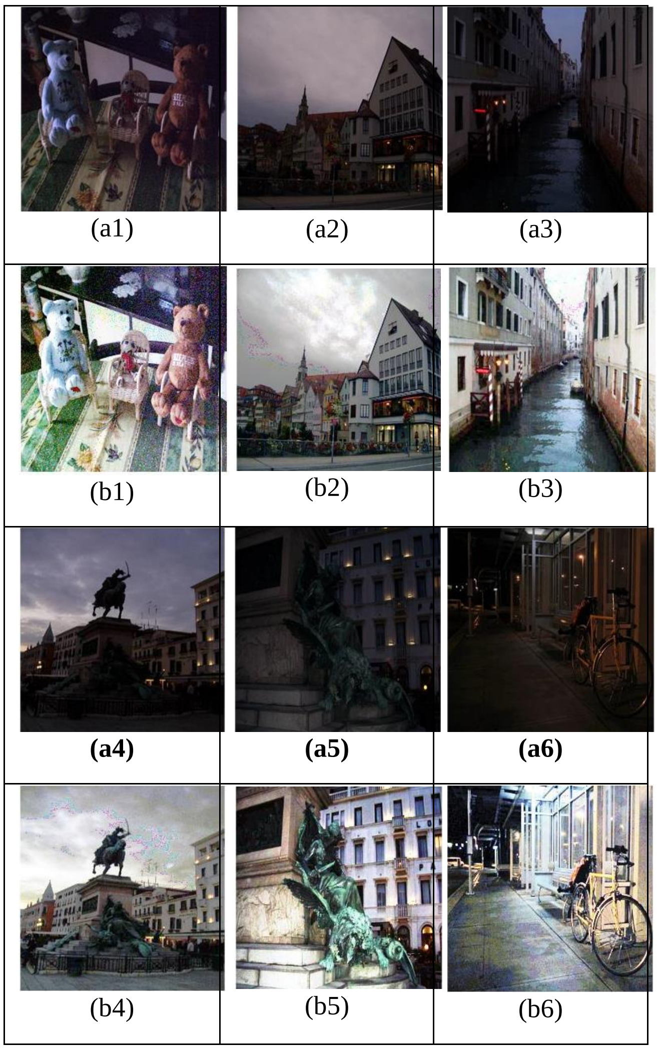


Fig 16 Input and Output RGB images of the proposed method (a1) – (a6) : Input Images (b1) – (b6) : Output Images

The common image enhancement qualitative metrics like entropy (E) and contrast per pixel (cpp) presented in equation (4) and (5) has been used to measure the performance of the proposed PHE architecture implemented using verilog. The entropy[14–17] of an image quantifies its information content. This term refers to the level of uncertainty or randomness in images. More information in an image improves its quality. Figure 17 shows the measured entropy for input and output images.

$$E = -\sum_{i=0}^{n-1} P_i \, log_2 \, P_i \qquad (4)$$

Where E refers to Entropy of image, n = 256 for 8 bit grayscale levels Pi refers to probability of pixels with grayscale level i.

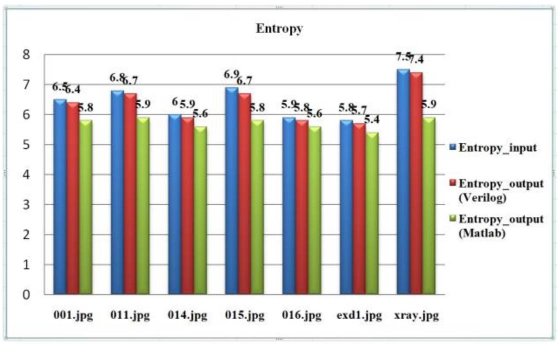


Fig 17 Entropy of Input and Output Images

In general Histogram equalisation reduces the entropy of input images by amplifying the noise. According to the experimental results, the output images obtained from matlab implementation of traditional Histogram equalisation have reduction in entropy by more than 7% to 20%, whereas the output images obtained from verilog implementation of the proposed design have reduction in entropy by only 2% compared to the input image. From this it can be concluded that the proposed design improves the image contrast with a very less information loss.

Contrast of an image is the difference between the image's light and dark intensities. A wider range of grey intensities leads to increased contrast. Contrast per pixel (CPP) is a measure that indicates how a pixel value differ from its neighboring pixel values.

$$\text{CPP} = \frac{\sum conv2(I,K)}{N} \tag{5}$$

Where I=3×3 image window,

k=kernel = $\frac{1}{8} \times [-1,-1,-1;\ -1,8,-1;\ -1,-1,-1\ ]$ and N=Image Size.

Figure 18 shows the contrast per pixel value for a set of low contrast dark input images and their corresponding output images. According to experimental results, the proposed design improves input image contrast by more than 50 percent, which makes it more appropriate for enhancing the brightness and contrast of low contrast dark images.

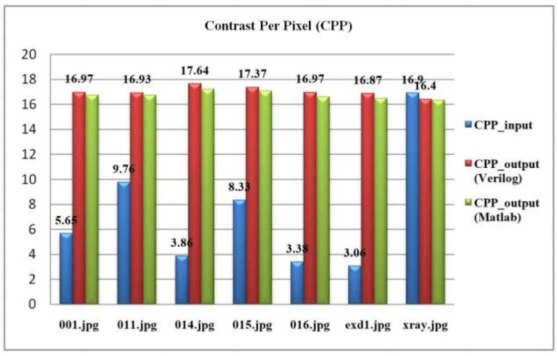


Fig 18 Contrast per Pixel of Input and Output Images

5. Conclusion and Future Work

The fundamental goal of this research work is to create reconfigurable parallel in hardware architecture for histogram equalisation that consumes reduced computing time and modest hardware resources. The suggested architecture was coded in both matlab and RTL compliant verilog HDL, and its functioning was tested using Xilinx ISIM, an integrated simulator tool. The proposed design was synthesized and implemented using the Xilinx Kintex 7 low voltage FPGA family device Xc7k70tl. According to the experimental results, the computation speed of the proposed architecture for implementing histogram equalisation on FPGA is 2 times faster than the serial architecture proposed in [11], 3.14 times faster than the existing architectures proposed in [3,4], and more than 12.5 times faster compared to the architecture proposed in [12]. The proposed design was synthesized and implemented using the Xilinx Kintex 7 low voltage FPGA family device, Xc7k70tl. According to the experimental results, the computation speed of the proposed architecture for implementing histogram equalisation on FPGA is 2 times faster than the serial architecture proposed in [11], 3.14 times faster than the existing architectures proposed in [3,4], and more than 12.5 times faster than the architecture proposed in [10]. This improvement in computation speed opens up new opportunities for improving the suitability the suggested design for real-time applications.

References

[1] A. Vyas, S. Yu, and J. Paik, *Fundamentals of digital image processing*. 2018.

[2] A. Shahbahrami, J. Y. Hur, B. Juurlink, and S. Wong, "FPGA Implementation of Parallel Histogram Computation."

[3] X. Li, G. Nt, Y. Cut, T. Pu, and Y. Zhong, "Real-time Image Histogram Equalization Using FPGA," [Online]. Available: http://proceedings.spiedigitallibrary.org/.

[4] A. M. Alsuwailem and S. A. Alshebeili, "A new approach for real-time histogram equalization using FPGA," *Proc. 2005 Int. Symp. Intell. Signal*

Process. Commun. Syst. ISPACS 2005, vol. 2005, pp. 397–400, 2005, doi: 10.1109/ispacs.2005.1595430.

[5] N. Sachdeva and T. Sachdeva, "An FPGA Based Real-time Histogram Equalization Circuit for Image Enhancement," vol. 7109, no. 2, pp. 63–67, 2010.

[6] J. O. Cadenas, R. Simon Sherratt, P. Huerta, and W.-C. Kao, "Parallel Pipelined Array Architectures for Real-time Histogram Computation in Consumer Devices," 2011.

[7] K. S. Gautam, "Parallel Histogram Calculation for FPGA: Histogram Calculation," in *Proceedings - 6th International Advanced Computing Conference, IACC 2016*, Aug. 2016, pp. 774–777, doi: 10.1109/IACC.2016.148.

[8] Q. Gan, J. M. P. Langlois, and Y. Savaria, "Parallel array histogram architecture for embedded implementations," *Electron. Lett.*, vol. 49, no. 2, pp. 99–101, Jan. 2013, doi: 10.1049/el.2012.2701.

[9] E. Jamro, M. Wielgosz, and K. Wiatr, "FPGA Implementation of Strongly Parallel Histogram Equalization."

[10] R. Z. Mahmood and H. A. T. Abdullah, "FPGA-Based high speed two ways parallel histogram computation for grey image," *Prz. Elektrotechniczny*, vol. 99, no. 1, pp. 120–123, 2023, doi: 10.15199/48.2023.01.23.

[11] M. A. Qureshi, A. Beghdadi, and M. Deriche, "Towards the design of a consistent image contrast enhancement evaluation measure," *Signal Process. Image Commun.*, vol. 58, no. August, pp. 212–227, 2017, doi: 10.1016/j.image.2017.08.004.

[12] D. B. Younis and B. M. K. Younis, "Low Cost Histogram Implementation for Image Processing using FPGA," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 745, no. 1, pp. 0–9, 2020, doi: 10.1088/1757-899X/745/1/012044.

[13] P. Agalya and M. C. Hanumantharaju, "High Speed and Efficient Reconfigurable Histogram Equalisation Architecture for Image Contrast Enhancement," in *Information, Communication and Computing Technology*, 2023, pp. 142–156.

[14] L. G. More, M. A. Brizuela, H. L. Ayala, D. P. Pinto-Roa, and J. L. V. Noguera, "Parameter tuning of CLAHE based on multi-objective optimization to achieve different contrast levels in medical images," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2015-Decem, no. May 2016, pp. 4644–4648, 2015, doi: 10.1109/ICIP.2015.7351687.

[15] K. Singh and R. Kapoor, "Image enhancement using Exposure based Sub Image Histogram Equalization," *Pattern Recognit. Lett.*, vol. 36, no. 1, pp. 10–14, 2014, doi: 10.1016/j.patrec.2013.08.024.

[16] J. C. M. Román, J. L. V. Noguera, H. Legal-Ayala, D. P. Pinto-Roa, S. Gomez-Guerrero, and M. G. Torres, "Entropy and contrast enhancement of infrared thermal images using the multiscale top-hat transform," *Entropy*, vol. 21, no. 3, pp. 1–19, 2019, doi: 10.3390/e21030244.

[17] D. Y. Tsai, Y. Lee, and E. Matsuyama, "Information entropy measure for evaluation of image quality," *J. Digit. Imaging*, vol. 21, no. 3, pp. 338–347, 2008, doi: 10.1007/s10278-007-9044-5.