Simultaneously Realization of Image Enhancement Techniques On Real-Time Fpga

Muhammed YILDIRIM

Computer Engineering Department /Fırat University, Elazığ,Turkey

muhyldrm23@gmail.com

Ahmet ÇINAR

Computer Engineering Department / Fırat University, Elazığ,Turkey

acinar1972@gmail.com

Abstract:

Image Processing Techniques have recently become one of the most popular and widely used technologies. Image processing techniques are widely used in many areas [1]. Particularly in real-time applications, imaging and processing in a very short time is of great importance. In this paper, Spatial Domain Techniques which are commonly used in almost all Image Processing Techniques are discussed. Field Programmable Gate Array (FPGA) architecture is used in order to perform the operations as soon as possible while applying image processing techniques.Since the FPGA architecture has the ability to perform parallel processing, it will shorten the processing time and the efficiency will increase [2]. Also, the ability to use FPGA over and over again provides an extra advantage.In this paper, applications are performed using the Vivado program using the verilog hardware identification language. In addition to the simultaneous acquisition of numerical data, corresponding visual figures have been developed in the MATLAB program so that the subject can be better understood.

Index Terms: **Fpga, Histogram, Image Enhancement, Verilog**

I. INTRODUCTION

With today's developing technology, image processing methods have been using in many fields. In general, if the image is used for what purpose, the image needs to be improved in the first step. Image enhancement techniques allow us to interpret the image more easily. It also aims to eliminate the unwanted values in the image. Images that are imaged enhancement are of higher quality and are easier to interpretion. Several methods and algorithms have been developed to improve image quality [3]. In this work, FPGA architecture which has the ability to perform parallel operation is used. In as much as the FPGA architecture can handle multiple operations at the same time, image enhancement algorithms can be performed simultaneously.

We can categorize image enhancement techniques as follows.

- Image Enhancement Techniques in the Spatial Domain
- Image Enhancement TEchniques in the Frequency Domain

In these categories, there are more than one methods developed in-house. This work deals with in the Spatial Domain category [4]. Gray level transformation, Histogram equalization, Image negatives, Image subtraction and Contrast stretching operations under this category are performed simultaneously on Fpga. Since the aforementioned image enhancement techniques are implemented using the same FPGA architecture, the results are produced using more than one technique in a very short time.

II. METHOD AND APPLICATION RESULTS

There are several methods and algorithms used in the image enhancement process. The methods used in this paper have been extensively discussed.

In this paper, histogram extraction, histogram equalization, Contrast Stretching, Image Negetives and image Substraction operations are performed on FPGA in real time. Since FPGA architecture have the ability to perform parallel processing, the aforementioned image enhancement methods are performed simultaneously. The result of more than one method is obtained at the same time. Simultaneous Image Enhancement Techniques is shown in figüre 2.1.

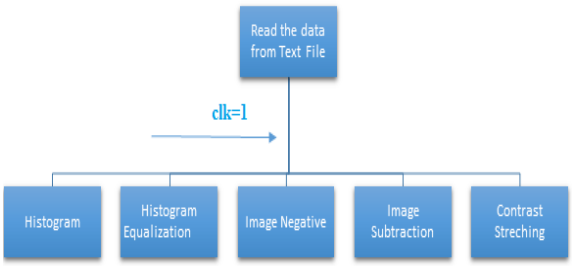

Fig. 2.1. Simultaneous Image Enhancement Techniques

In the application first read the data from the text file. The $ fopen command is used to read data from the file. Then we read the data we have been transferred to the array. Code block reading the data from the text file is shown in figüre 2.2.

```
fileH = $fopen ("D:\\text.txt", "r");
    for (m=0;m200;m=m+1) begin
      for(n=0;n<200;n=n+1) begin
         $fscanf (fileH, "%d\n", datafile[m][n]);
       end
   end
```

Fig 2.2. Read the data from text file

A. Histogram

The histogram is the graphical representation frequency of the values of the pixels on the image. This process is called the image histogram. The image histogram shows the detection of the pixels at each point of the scene and the number of these pixels. The left side of the graph shows the intensity of the image is dark, and the right side shows the intensity is clear [5].

Then, the histogram of the gray level view is obtained. The code block in Figure 2.3 is used for histogram extraction.

```
always @(posedge clk)  //histogram
    for (m=0;m<200;m=m1) begin
       for(n=0;n<200;n=n+1) begin
       h=datafile[m][n];
       histogram[h]=histogram[h] + 1;
       end
    end
```

Fig. 2.3 Histogram Code

When the histogram extraction code block is executed, the result is transferred to the histogram array. Simulation image in the Vivado program is the same as figure 2.4.

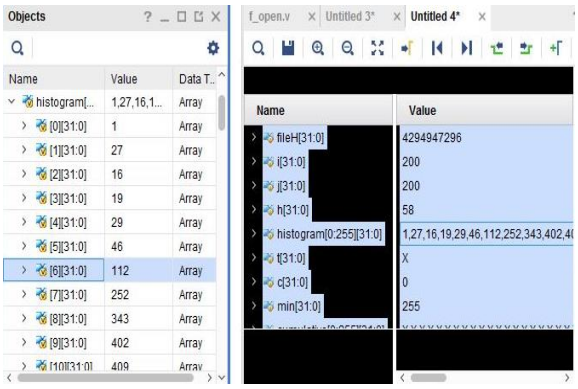

Fig. 2.4. Histogram Simulation

The histogram of the image on the matlab in Figure 2.5. is shown.

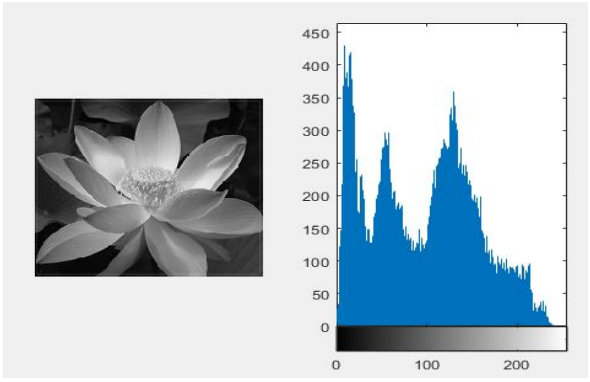

Fig. 2.5. Histogram of Image

After the histogram of the image is extracted, histogram equalization, Contrast Stretching, Image Negeatives and image Substraction are performed parallel to each other.

B. Histogram Equalization

Once the histogram of the image has been extracted, the histogram equalization method, one of the image enhancement methods, has been applied to the image. Histogram equalization is a frequently used pre-processing method that can improve image quality by extending the density dynamic range with the entire image histogram. The image density distribution is normalized and the image is improved with a uniform intensity distribution. The following steps are performed in the histogram synchronization process.

- The histogram of the view is obtained.

- Cumulative histogram is obtained from the histogram. The cumulative histogram is the magnitude of each value of the histogram,

including the values obtained before itself and the sum of its own.

- The cumulative histogram values are normalized (divided by the total pixel value), multiplied by the maximum color values that we want to be in the new image, and the resulting value is rounded to integers. Thus, new gray level values are obtained.

- The old (original) gray level values and the gray level values obtained in Step 3 are reduced to each other and a new histogram graph is drawn. [6].

The code block in figure 2.6 is written in the vivado program for histogram equalization.

```
always @(posedge clk)  //cumulative
    for (t=0;t<256;t=t+1) begin
        c= c+ histogram[t];
        cumulative[t]=c;
        if(cumulative[t]<min) begin
            min=cumulative[t];
        end
    end
    for(t=0;t<256;t=t+1) begin
    a= (cumulative[t] - min);
    b= 40000 - min; //MXN
    normalized_hist[t]=((a/b)*255);
    end
```

Fig. 2.6. Histogram Equalization Code Block

When the code block in figure 2.6 is executed, the result is transferred to the normalized_hist array. The simulation image in the Vivado program is the same as in figure 2.7.

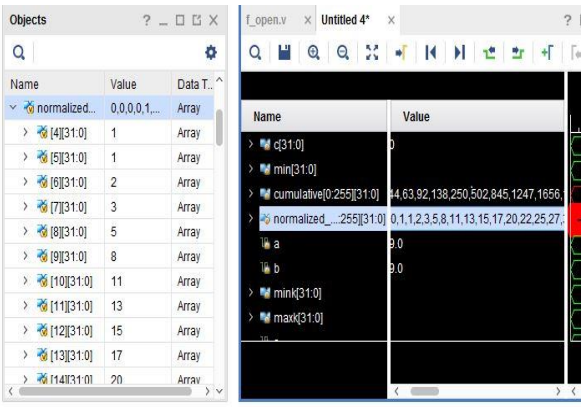

Fig. 2.7. Histogram Equalization Simulation

The histogram Equalization of the image on the matlab in Figure 2.8. is shown.

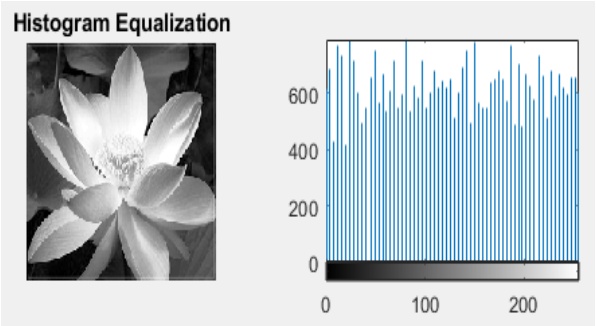

Fig. 2.8. Histogram Equalization on the Matlab

C. Contrast Streching

Contrast stretching is the process of distributing and displaying the gray tones across the screen for the monitor to use all shades of gray [7]. That is, in an 8-bit image, values between 0-255 must be displayed. But let's it be that our image is scattered between 20 and 180. Then 0-19 intervals and gray values between 181-255 are not used. This makes the image relatively dark and low contrast. After contrast stretching, values between 0-255 are displayed.

When contrast stretching process is applied, firstly minimum and maximum gray values are found in the image and a linear transformation is performed. All gray values are recalculated so that the minimum value 0 is the maximum value 255 and the other values fall within the range of 0-255.

The code block in Figure 2.9 is written for the Contrast Stretching process.

```
/// contrast streching
    always @(posedge clk)
     for (m=0;m<200;m=m+1) begin
          for(n=0;n<200;n=n+1) begin
```

```
            if(datafile[m][n]<mink) begin
               mink=datafile[m][n];
            end
            if(datafile[m][n]>maxk) begin
               maxk=datafile[m][n];
            end
          end
      end
      for (m=0;m<200;m=m+1) begin
              for(n=0;n<200;n=n+1) begin
               s=datafile[m][n] -mink;
               s1=maxk- mink;
               s=s/s1;
               Gdoutput[m][n] = s *255;
              end
       end
```

Fig. 2.9. Contrast Streching Code Block

When the code block in Figure 2.9 is executed, the result obtained after the Contrast Stretching process is transferred to the Gdoutput array. The simulation image in the Vivado program is the same as in figure 2.10.

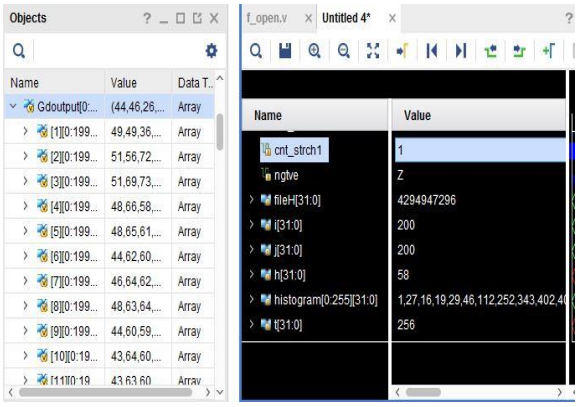

Fig. 2.10. Constrast Streching on Vivado Simulation

The Contrast Streching of the image on the matlab in Figure 2.11. is shown.

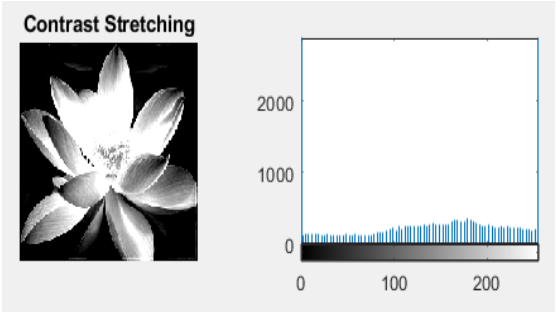

Fig. 2.11. Constrast Streching in Matlab

D. Image Negative

In images with a black background, it makes white and gray tones stand out. In 8-bit images, the pixel value is subtracted from 255 to perform the invert operation. Obtained values are normalized in range (0,255).

The code block in figure 2.12 is written for Image Negative operation.

```
always @(posedge clk)   //negative
     for (m=0;m<200;m=m+1) begin
          for(n=0;n<200;n=n+1) begin
          negative[m][n]= 255 - datafile[m][n];
      end
   end
```

Fig. 2.12. Image Negative Code Block

When the code block in Figure 2.12 is executed, the result obtained after the Image Negative operation is transferred to the negative array. The simulation image in the Vivado program is the same as figure 2.13.

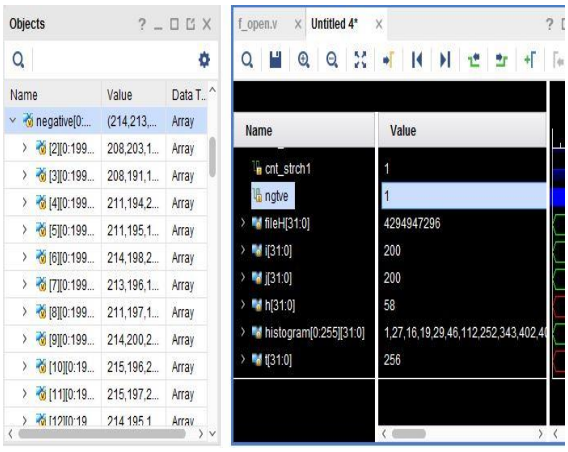

Fig. 2.13. Image Negative on Vivado Simulation

The Image Negative of the image on the matlab in Figure 2.14. is shown.

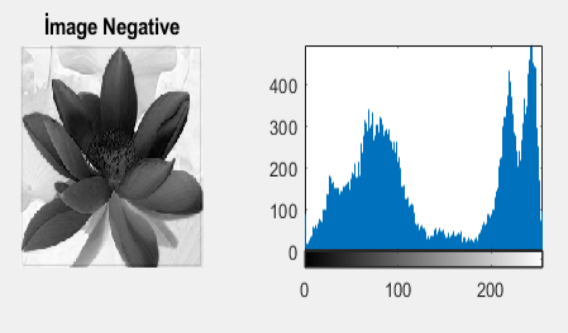

Fig. 2.14. Image Negative in the Matlab

E. Image Subtraction

With the Image Substraction process, images on bright floors can be made more visible. The main reason for using this method is that it is simple and easy to implement [8]. Two input images can be extracted from each other and a third image can be obtained or a constant value can be extracted from an image.

The code block in figure 2.15 is written in the vivado program for image subtraction operation.

```
always @(posedge clk)   //subtraction
for (m=0;m<200;m=m+1) begin
        for(n=0;n<200;n=n+1) begin
        sub=datafile[m][n] - sub_dgr;
        if(sub<0) begin
            subtraction[m][n]= 0 ;
        end
        else
            subtraction[m][n]=sub;
        end
    end
```

Fig. 2.15 Image Subtraction Code Block

When the code block in Figure 2.15 is executed, the result obtained after the image substraction is transferred to the substraction array. The simulation image in the Vivado program is the same as figure 2.16.

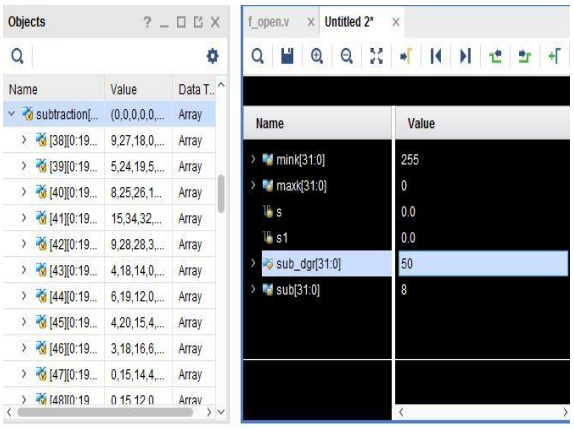

Fig. 2.16. Image Subtraction on Vivado Simulation

The Image Subtraction of the image on the matlab in Figure 2.17. is shown.

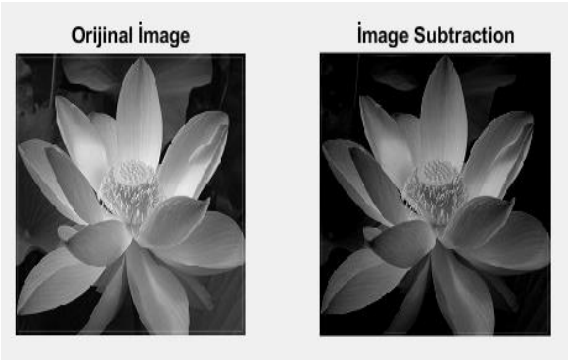

Fig. 2.17. Image Subtraction in the Matlab

III. RESULT

FPGA architecture is capable of parallel processing and very fast running hardware. It is also a great advantage that they can be used repeatedly. For this reason, FPGA architecture is preferred in this application because complex calculations such as image enhancement are used and high speed is required. Using FPGA architecture, multiple image enhancement methods were implemented in parallel. Image histogram subtraction, histogram equalization, image negative, image substraction and contrast stretching operations are performed in parallel. For this reason, the results were obtained simultaneously. In applications where time is of the essence, such as image enhancement, the FPGA architecture allows the image to be processed faster. Simultaneous image enhancement steps can be applied to multiple images with this work. This paper demonstrates that successful results can be achieved in a shorter time in applications where time is of the essence, such as image enhancement.

REFERENCES

[1] Janani, P., Premaladha, J., & Ravichandran, K. S. (2015). Image enhancement techniques: A study. Indian Journal of Science and Technology, 8(22).

[2] Özgen, C. (2010). Efficient FPGA Implementation Of Image Enhancement Using Video Streams (Doctoral dissertation, Middle East Technical University).

[3] Adak, E. (2011). FPGA based advanced video enhancement algorithms (Doctoral dissertation, DEÜ Fen Bilimleri Enstitüsü).

[4] Kaur, H., & Narang, S. Comprehensive Review on Image Enhancement Techniques.

[5] Hazra, S., Ghosh, S., Maity, S. P., & Rahaman, H. (2016). A New FPGA and Programmable SoC Based VLSI Architecture for Histogram Generation of Grayscale Images for Image Processing Applications. Procedia Computer Science, 93, 139-145.

[6] Singh, R. P., & Dixit, M. (2015). Histogram equalization: a strong technique for image enhancement. International Journal of Signal Processing, Image Processing and Pattern Recognition, 8(8), 345-352.

[7] Oktavianto, B., & Purboyo, T. W. (2018). A Study of Histogram Equalization Techniques for Image Enhancement. International Journal of Applied Engineering Research, 13(2), 1165-1170

[8] Gonzalez Rafael, C., Woods Richard, E., & Eddins Steven, L. (2004). Digital image processing using MATLAB. Editorial Pearson-Prentice Hall. USA.