FPGA Implementation of Image Enhancement Using Verilog HDL

Mandeep Singh Narula[1], Nishant Singla[2]

[1]Professor, Dept. of E.C.E., JIIT, Noida, India
[2]Student, Dept. of E.C.E., JIIT, Noida, India.

---***---

Abstract **-** *The demand of Image Processing methods traditionally implemented on a digital processing software such as MATLAB is increasing widely to get high performance.*

In this project we implemented four basic operations of Image Enhancement i.e. threshold, contrast, brightness, invert to manipulate the RGB values of every pixel of the image to improve the human interpretation of image[1].

To perform the above mentioned operations we have implemented Image Enhancement on FPGA (Field Programmable Gate Array) using Verilog HDL. Implementation in HDL (Hardware Description Language) is quite different from implementation in MATLAB mainly because of the parallel nature of the HDLs. The system is implemented on FPGA[5], which is modern programmable logic device, i.e. we can program almost any digital function in it.

Keywords - Verilog; FPGA,DE0 Nano; Image Enhancement

I. INTRODUCTION

There are many Hardware Description Languages (HDLs) available to help the engineers describe the circuit both logically and functionally so that they can simulate and properly calculate the performances with the help of personalized test environment and clock cycle.

Since the HDL syntax is always related to a hardware structure, the timing information of the potential hardware implementation is also available allowing specific speed optimizations. Above all, with the use of HDLs it means that we can enjoy hardware portability and on-the-fly re-programmability. But here the bigger challenge is to implement the validated algorithms into a non-programming language as hardware description languages are. Also, the input and output RGB files need to be constructed accordingly to match the binary content permitted into the hardware simulators[4].

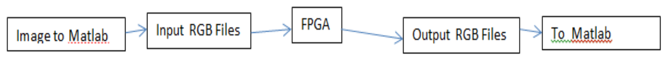

Fig - 1 : Block Diagram of the System

Among all, the most interesting image processing approaches is the image enhancement. The importance for this domain is mainly for two application directions:

1. Improve the human interpretation and enhance the pictorial visual information;

2. Modify information of image illustration so as to optimize it for data storage, transmission or different illustration for autonomous machine perception.

The main goal of any improvement methodology is simply too acquire a a lot of appropriate result compared with the first as is from the purpose of read of a selected application.

Any image improvement procedures are often categorised into 2 approaches: spatial domain methods and frequency domain methods. The spatial domain refers to the pixels structure of the image plane itself and this sort of improvement is predicated on direct manipulation of these pixels of a picture. Frequency domain process techniques area unit mistreatment mathematical transforms to induce totally different enhancements. The Fourier remodel of a picture is accepted for these functions[9].

Some of the best, yet useful, image process operations within the spatial domain involves the adjustment of brightness, distinction or colour a picture. A reason for manipulating these attributes is to reduce the difficulties in image acquisition and with image process we will increase the general brightness of the item of interest and amplify the small residual variations in distinction across it. This image process operations will reveal enough detail to permit correct interpretation. Some mainly used point operations are: [2]

- modifying image brightness or contrast,
- applying arbitrary intensity transformations ("curves"),
- quantizing (or "posterizing") images,
- global thresholding,
- gamma correction,
- color transformations.

II. IMPLEMENTATION OF IMAGE ENHANCEMENT METHODS USING VERILOG HDL

Point process operation is performed to reinforce a picture and details not clearly visible within the original image could come into view upon application of the point operation. The aim of the paper is to explain some basic image enhancement strategies employing a hardware description language, Verilog.

The Verilog language has the power to browse or write files from a storage setting. This feature create it potential to significantly style the test benches to browse the test information from device, generate the stimulant signals to the Verilog check module and write back the results to the device. sadly, Verilog solely browse (and write) ASCII character files being unable to browse pictures in commonplace formats like image or jpeg directly from disk [6].

The project is implemented on FPGA using Verilog, which is Hardware Description Language. The code written in Verilog describes the behaviour of the desired hardware. The code is taken by Altera's synthesis tool (we use Altera's toolchain because we have an Altera Cyclone IV FPGA) which 'try' to find an implementation of the description of the code.

The word try is empathised as the tools, advance as they may be, might not be able to find a correct implementation for given description or might produce poor or overly complex implementation, thus it is our job as designer to write synthesizable code.

The main idea is to have a general view of the kind of circuit implementation will be derived by the tools and to logically partition the module in the code, it is easy to correctly implement small module interconnected then a very big and complex block.

RGB-files contain only information about RGB vector for each pixel of the input image and does not contain information about image dimensions or similar. The data from files was applied as stimulus to the point operations blocks described in Verilog language.

The result was obtained in another external file and we create an application described in MATLAB to show the modified output image and to compare with the original input image.

Next, we describe the theory and implementation, using Verilog language, of most commonly used point operations used for image enhancement[3]:

A. Contrast manipulation

B. Brightness manipulation

C. Inverting images

D. Threshold operation

In view of above ideas, we have tried to best partition our implementation, using many modules.

a. Hardware

The system is supposed to be implemented on a Terasic DE0-Nano[8] FPGA development board. As the board does not have many peripherals we need to make our own expansion board to connect the FPGA to controller, VGA monitor . All the hardware built along with the development board is explained in this section.

b. Terasic DE0-Nano FPGA Development Board:

This Project uses a Terasic DE0-Nano FPGA Development Board, it introduces a compact-sized FPGA development platform suited for to a wide range of portable design projects.

The DE0-Nano features a powerful Altera Cyclone IV FPGA (with 22,320 logic elements), 32 MB of SDRAM, 2 Kb EEPROM, and a 64 Mb serial configuration memory device. For connecting to real-world sensors the DE0-Nano includes a National Semiconductor 8-channel 12-bit A/D converter, and it also features an Analog Devices 13-bit, 3-axis accelerometer device.

The DE0-Nano board includes a built-in USB Blaster for FPGA programming, and the board can be powered either from this USB port or by an external power source. The board includes expansion headers that can be used to attach various Terasic daughter cards or other devices, such as motors and actuators. Inputs and outputs include 2 pushbuttons, 8 user LEDs and a set of 4 dip-switches.

The key feature of the board are listed below –

- Featured device
  - o Altera Cyclone IV EP4CE22F17C6N FPGA
  - o 153 maximum FPGA I/O pins
- Configuration status and set-up elements
  - o On-board USB-Blaster circuit for programming
  - o Spansion EPCS64
- Expansion header
  - o Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
  - o 32MB SDRAM
  - o 2Kb I2C EEPROM
- General user input/output
  - o 8 green LEDs
  - o 2 debounced pushbuttons
  - o 4-position DIP switch
- G-Sensor
  - o ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
- Clock system
  - o On-board 50MHz clock oscillator
- Power Supply

- o USB Type mini-AB port (5V)
- o DC 5V pin for each GPIO header (2 DC 5V pins)
- o 2-pin external power header (3.6-5.7V)

c. Verilog HDL Hardware

The complete system is implemented in Verilog. As a successful implementation in Verilog calls for good logical partitioning of the circuit, various modules are created that are interconnected to make the whole system.

Module Name: Contrast Operation: - This module changes the contrast of the picture by setting the darkest pixel value to black, the brightest value to white, and others to different shades of gray which makes good use of the display and enhances the visibility of features in the image[10].

```
if (value > threshold) begin
      tempR = Rin + valueToAdd;
      if (tempR > 256)
            Rout = 255;
      else
            Rout = Rin + valueToAdd;
      tempG = Gin + valueToAdd;
      if (tempG > 256)
            Gout = 255;
      else
            Gout = Gin + valueToAdd;
      tempB = Bin + valueToAdd;
      if (tempB > 256)
            Bout = 255;
      else
            Bout = Bin + valueToAdd;
end
```

```
if (value < threshold) begin
      tempR = Rin - valueToSubstract;
      if(tempR[8] == 1)
            Rout = 0;
      else
            Rout = Rin - valueToSubtract;
      tempG = Gin - valueToSubtract;
      if(tempG[8] == 1)
            Gout = 0;
      else
            Gout = Gin - valueToSubtract;
      tempB = Bin - valueToSubstract;
      if(tempB[8] == 1)
            Bout = 0;
      else
            Bout = Bin - valueToSubtract;
end
```

Module Name: Brightness Operation: - This module changes the brightness of the picture by adding or subtracting a fixed value to the pixel value. The purpose of the below mentioned code is add and subtract a constant value to the image pixel values

```
if(sign == 1) begin
      tempR = Rin + value;
      if (tempR > 256)
            Rout = 255;
      else
            Rout = Rin + value;
      tempG = Gin + value;
      if (tempG > 256)
            Gout = 255;
      else
            Gout = Gin + value;
      tempB = Bin + value;
      if (tempB > 256)
            Bout = 255;
      else
            Bout = Bin + value;
end
```

```
if(sign == 0) begin
      tempR = Rin - value;
      if (tempR[8] == 1)
            Rout = 0;
      else
            Rout = Rin - value;
      tempG = Gin - value;
      if (tempG[8] == 1)
            Gout = 0;
      else
            Gout = Gin - value;
      tempB = Bin - value;
      if (tempB_b[8] == 1)
            Bout = 0;
      else
            Bout = Bin - value;
      end
```

Module Name: Invert Image: - This module inverts an image by inverting the bits of the grayscale pixel value of an image.

And, to change a coloured image into a grayscale one, the RGB pixel values must be equalized and it is done by taking e the average of the three color components[7].

$$R_{eq} = G_{eq} = B_{eq} = \frac{R+G+B}{3}$$

```
  value2 = (Rin + Gin + Bin)/2;
  value4 = (Rin + Gin + Bin)/4;
  value  = (value2 + value4)/2;
      Rout = 255 - value;
      Gout = 255 - value;
      Bout = 255 - value;
```

Module Name: Threshold operation: - This module is used to perform the threshold operation i.e. set the pixel above a threshold value to 255 and below it to 0.

The threshold operation can be performed using below mentioned Verilog testing code[9].

```
if (value > threshold) begin
      Rout = 255;
      Gout = 255;
      Bout = 255;
end
if (value < threshold) begin
      Rout = 0;
      Gout = 0;
      Bout = 0;
end
```

III. Advantages of FPGA

- Better performance than DSP's
- Less Time to Market
- Low long-term Cost
- More Reliability
- Long-term Maintenance

IV. RESULTS

The simulation results obtained after applying the operations described using Verilog HDL to an input image are shown here.

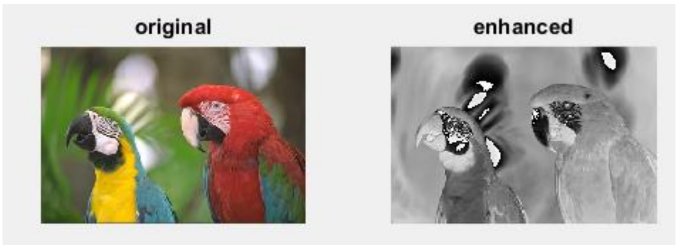

Fig - 2 : Verilog result for Invert Operation

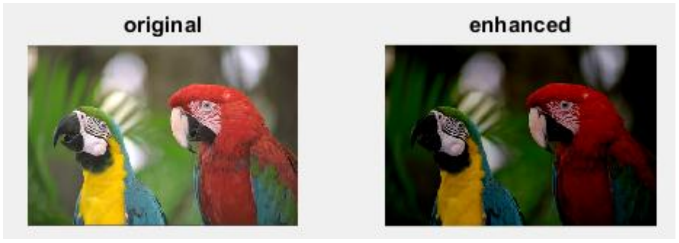

Fig - 3 : Verilog result for Contrast Operation using threshold = 90, valueToAdd = 10 and valueToSubtract = 15

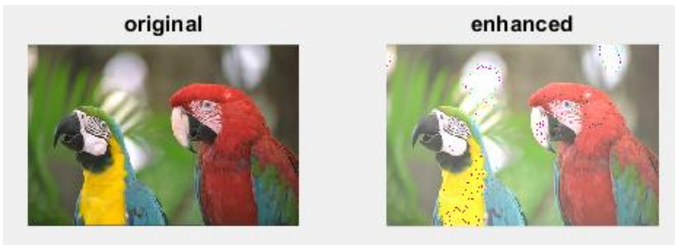

Fig - 4 : Verilog result for Brightness Operation using sign = 0 and value = 60

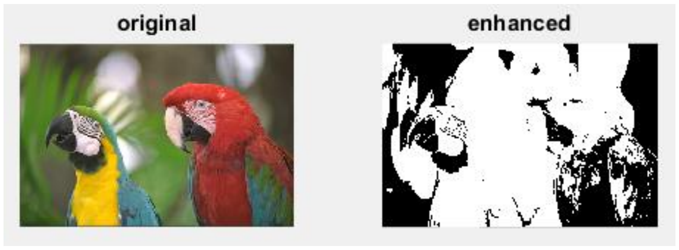

Fig - 5 : Black & White result using threshold = 120

V. REFERENCES

[1] John C. Russ - "Image Processing Handbook (sixth edition)", CRC Press, pp. 270-331, 2011

[2] Raman Maini, H. Aggarwal - "A Comprehensive Review of Image enhancement Techniques", Journal of Computing, vol. 2, issue 3, ISSN 2151-9617, pp. 269-300, 2010.

[3] Wilhelm Burger, Mark J. Burge - "Principles of Digital Image Processing – Fundamental Techniques", Undergraduate Topics in Computer Science, DOI 10.1007/978-1-84800-191-6_4, Springer- Verlag London Limited, 2009.

[4] A. Zuloaga, J.L. Martin, U. Bidarte, J.A. Ezquerra - "VHDL test bench for digital image processing systems using a new image format", ECSI, 2007 (http://mx.reocities.com/CapeCanaveral/8482/).

[5] Daggu Venkateshwar Rao, Shruti Patil, Naveen Anne Babu and V. Muthukumar - "Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture using C-based Hardware Descriptive Languages", International Journal of Theoretical and Applied Computer Sciences, Volume 1, Number 1, pp. 9–34, 2006 (http://www.gbspublisher.com/ijtacs/1002.pdf).

[6] "Verilog HDL" by Samir Palnitkar, 2003, ISBN 0-13-044911-3 [Publisher: Prentice Hall PTR]

[7] R. C. Gonzalez, R. E. Woods – "Digital Image Processing", Prentice Hall, ISBN 0-13-094659-8, pp. 1-142, 2002.

[8] Bovik, A. (Ed.). (2000). Handbook of image and video processing. Texas: Academic Press.

[9] Nick Efford - "Digital Image Processing – A Practical Introduction Using Java", pp. 103-132, 2000.

[10] Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (1996). Hypermedia image processing reference. Chichester: John Wiley & Sons, Inc.