Image Processing on FPGA

PRABHAT KUMAR RAI, SIDDHARTH GUPTA, GVV SHARMA
[ee18mtech01005, ee18mtech01003, gadepall] @iith.ac.in

I. Objective

Image processing is the means by which the input image is modified by one or more mathematical algorithms to generate an output image that is enhanced in some way. For example, edges may be enhanced or the noise reduced. Image processing is often used to prepare images prior to analysis. In the following work, we will be doing colour inversion of the image using FPGA and Arduino board.

II. Introduction

An image consists of three channels one for each R, G and B i.e. Red, Green and Blue. So for inversion, we need to invert all the three layers for an accurate result. An image is basically an array of these R, G, B values stacked one over other. Each number in the array is ranging between 0 to 255. So in this 3-Dimensional array, we will be having a different value between 0 and 255 representing the intensity of the pixel for the particular colour channel.

For colour inversion, there is a simple rule as depicted by the images shown in Figure 1. the original colour R, G, B values are 245, 0, 87 and inverted colour R, G, B values are 10, 255, 168.

So we need to subtract the original image colour from 255 to obtain the inverted image colour.

Hence:

$$\mathrm{R}_{inverted} = 255 - R_{original}$$
$$\mathrm{G}_{inverted} = 255 - G_{original}$$
$$\mathrm{B}_{inverted} = 255 - B_{original}$$

Figure 1: Colour Inversion Example

III. Hardware Specifications

1) 2 numbers of Arduino Uno Board, 14 Digital I/O Pins, Operating Voltage = 5V, Microcontroller = ATmega328P
2) IcoBoard, contains a Lattice FPGA with 8k LUT, 100 MHz max clock, 8 MBit of SRAM, programmable in Verilog by an open source FPGA toolchain. IcoBoard is pin-compatible with RaspberryPi 2B.
3) RaspberryPi 2B, 1GB RAM, 40pin extended GPIO, 4 pole Stereo output and Composite video port, Full size HDMI, 4xUSB 2 ports

IV. Python Script

```
# coding: utf-8
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy
import PIL
import serial
from serial import Serial
import time
from time import sleep

# Port Initilization
# For sending data to arduino
port1 = '/dev/ttyACM1'
speed1 = 9600
```

```
# For receiving data from arduino
port2 = '/dev/ttyACM0'
speed2 = 9600

ardData1 = serial.Serial(port1,speed1)
ardData2 = serial.Serial(port2,speed2)

# Convert PIL Image to NumPy array
from PIL import Image
image = Image.open("tiger.bmp")
img = image.resize((210,118))
arr = numpy.array(img)
img = PIL.Image.fromarray(arr)

# Send Array to Arduino
for i in range(arr.shape[2]):
   im = arr[:,:,i]
   for j in range(im.shape[0]):
      for k in range(im.shape[1]):
         string=str(int(im[j][k]))+'\n'
         # Sending array to arduino
         ardData1.write(string.encode())
         # Receiving array from arduino
         ms = ardData2.readline()
         try:
            msg=str(ms.decode("utf-8"))
         except:
            msg=0
         try:
            zz = int(msg)
         except:
            zz = 0
         print(im[j][k],zz,j,k )
         im[j][k] = zz
   arr[:,:,i] = im[:,:]
print("Colour Inversion is Completed")

# Convert Array to Image and save
plt.imshow(arr)
plt.savefig("inverted_tiger.jpg")
fig = plt.figure(figsize=(8, 8))
fig.add_subplot(1,2,1)
plt.imshow(img)
fig.add_subplot(1,2,2)
plt.imshow(arr)
plt.show()
fig.savefig("both")
```

V. ARDUINO TO ICOBOARD CODE

```
char serialData;
void setup() {
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(11, OUTPUT);
   pinMode(12, OUTPUT);
   pinMode(13, OUTPUT);
   Serial.begin(9600);
   //set the baud rate
   }

 void loop() {
   if(Serial.available()>0){
      serialData = Serial.parseInt();
      Int value = serialData;
 // Conversion of integer data to
   binary data
      bool b7 = value%2;
      value = (value>>1);
      bool b6 = value%2;
      value = (value>>1);
      bool b5 = value%2;
      value = (value>>1);
      bool b4 = value%2;
      value = (value>>1);
      bool b3 = value%2;
      value = (value>>1);
      bool b2 = value%2;
      value = (value>>1);
      bool b1 = value%2;
      value = (value>>1);
      bool b0 = value%2;
 // For sending binary bits to icoboard
      digitalWrite(13,b7);
      digitalWrite(12,b6);
      digitalWrite(11,b5);
      digitalWrite(10,b4);
      digitalWrite(9,b3);
      digitalWrite(8,b2);
      digitalWrite(7,b1);
      digitalWrite(6,b0);
   }
 }
```

VI. VERILOG .v CODE TO INVERT THE COLOUR OF IMAGE

```
module input_check(
       input clk, input [7:0] x_in,
       output [7:0] y_out );

reg [7:0] temp;

initial begin
reg temp1 = 1;
end
```

```
// For inversion, subtract every binary
    bits from 255(11111111)
  always@(posedge clk) begin
      temp[0] <= temp1 - x_in[0];
      temp[1] <= temp1 - x_in[1];
      temp[2] <= temp1 - x_in[2];
      temp[3] <= temp1 - x_in[3];
      temp[4] <= temp1 - x_in[4];
      temp[5] <= temp1 - x_in[5];
      temp[6] <= temp1 - x_in[6];
      temp[7] <= temp1 - x_in[7];
  end

endmodule
```

VII. VERILOG .PCF CODE

The mapping has done as per the .pcf(Physical Constraint File) file

```
    set_io clk R9
// y_out will give processed output from
    fpga to arduino
    set_io y_out[0] B11
    set_io y_out[1] B10
    set_io y_out[2] B9
    set_io y_out[3] D8
    set_io y_out[4] L9
    set_io y_out[5] G5
    set_io y_out[6] L7
    set_io y_out[7] N6
// x_in will give input from arduino to
    fpga for processing
    set_io x_in[0] B4
    set_io x_in[1] C3
    set_io x_in[2] A2
    set_io x_in[3] A5
    set_io x_in[4] B7
    set_io x_in[5] B6
    set_io x_in[6] B3
    set_io x_in[7] B5
```

VIII. ICOBOARD TO ARDUINO CODE

```
    void setup() {
       pinMode(2, INPUT);
       pinMode(3, INPUT);
       pinMode(4, INPUT);
       pinMode(5, INPUT);
       pinMode(10, INPUT);
       pinMode(11, INPUT);
       pinMode(12, INPUT);
       pinMode(13, INPUT);
       Serial.begin(9600);
       // set the baud rate
       }
    void loop() {
    // For receiving binary bits from
       icoboard
       int z0 = digitalRead(2);
       int z1 = digitalRead(3);
       int z2 = digitalRead(4);
       int z3 = digitalRead(5);
       int z4 = digitalRead(10);
       int z5 = digitalRead(11);
       int z6 = digitalRead(12);
       int z7 = digitalRead(13);
    // Conversion of binary data to
       integer data
       int no = (z7<<7) + (z6<<6) +
          (z5<<5) + (z4<<4) + (z3<<3) +
          (z2<<2)+(z1<<1)+z0;
       Serial.println( int(no) );
       }
```

Figure 2: Tiger Image

Figure 3: Inverted Tiger Image

IX. OTHER EXAMPLE IMAGE

Figure 4: Desktop Image

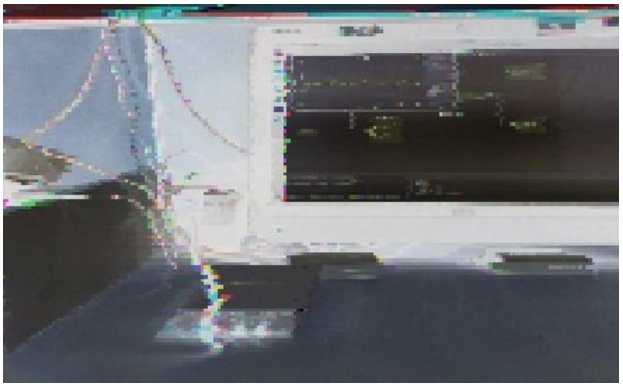

Figure 5: Inverted Desktop Image

X. IMPLEMENTATION DETAILS AND METHODS

Image processing is done on FPGA board in which colour inversion of Image has been implemented. To do so conversion of image into hexadecimal code is done with the help of python code and provide the output to FPGA via ardiuno and take the processed data back from FPGA via another arduino into the python script and save it as a new image.

XI. USE CASE

The image reading Verilog code will operate as a Verilog model of an image `sensor/camera`, which can be really helpful for functional verifications in real-time `FPGA` image processing projects. The image writing part is also extremely useful for testing as well when you want to see the output image in `BMP` or any format.

XII. REFERENCES

1) Git Repo: https://github.com/prabhatrai111/FPGA_PKR_SID_EE5811/
2) M. I. AlAli, K. M. Mhaidat and I. A. Aljarrah, "Implementing image processing algorithms in FPGA hardware," 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Amman, 2013, pp. 1-5.
3) https://www.fpga4student.com/2016/11/image-processing-on-fpga-verilog.html
4) https://www.vision-systems.com/articles/print/volume-22/issue-8/features/cpu-or-fpga-for-image-processing-which-is-best.html
5) Johnston, Christopher T., K. T. Gribbon and Donald G. Bailey. Implementing Image Processing Algorithms on FPGAs. (2005).