A PROJECT REPORT

ON

“STREAMING COLOUR INTERPOLATER IN A DIGITAL IMAGE”

BY

Himanshu Payal	140050011
Aakash Praliya	140050012
Amey Gupta	140050026
Surender Singh Lamba	140050075

UNDER THE GUIDANCE OF
PROF. Supratik Chakraborty

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Bombay

Acknowledgements

We are profoundly grateful to **Prof. Supratik Chakraborty** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

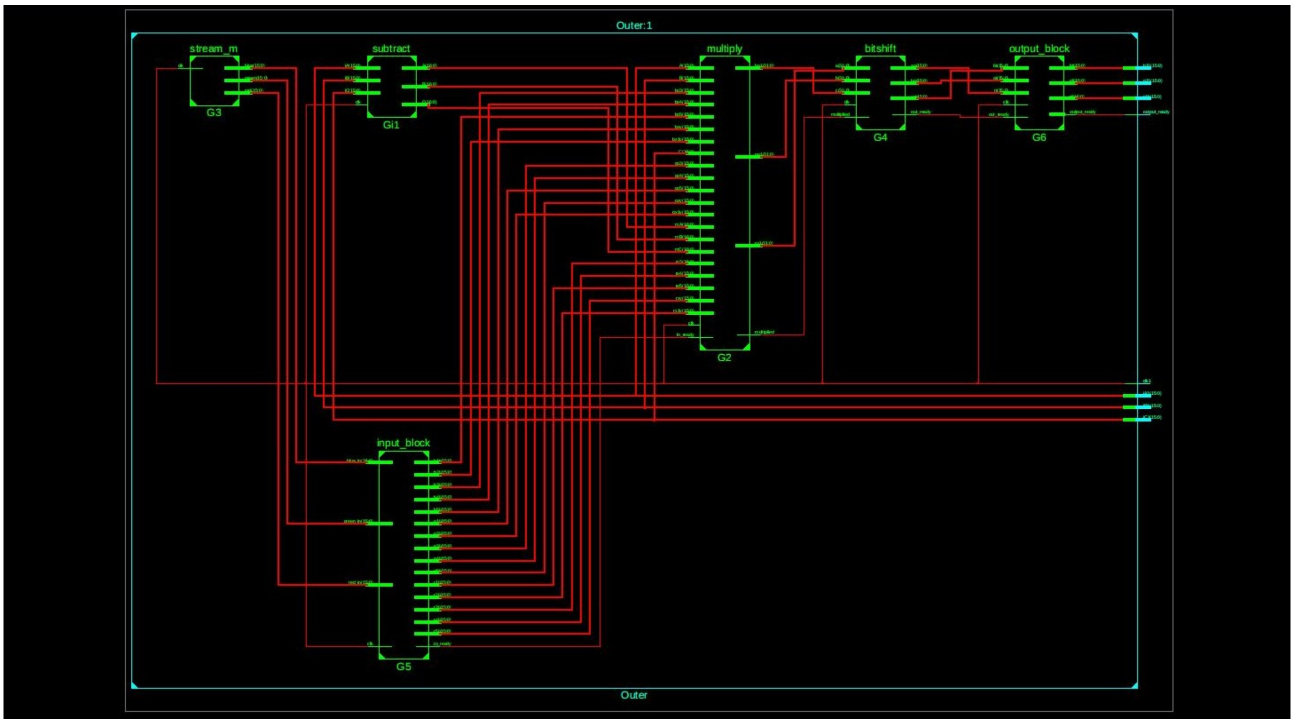
Himanshu Payal
Aakash Praliya
Amey Gupta
Surender Singh Lamba

Contents

Chapter 1

High Level Architecture of the System

1.1 Diagram



1.2 Description

The device consists of only one outer block whose sub-components are :

1. stream m
2. input block
3. multiply
4. bitshift

5. subtract
6. output block

The outermost block, i.e the device has the following inputs:

1. Start[1-bit] ,To start the reading of inputs of r,g,b and start processing
2. Reset[1-bit],To reset the device to initial state
3. Clock[1-bit],The clock used to synchronise the circuit
4. Red[16-bit],The red pixel value
5. Green[16-bit],The green pixel value
6. Blue[16-bit],The blue pixel value
7. A[16-bit],The value to be used in interpolation
8. B[16-bit],The value to be used in interpolation
9. C[16-bit],The value to be used in interpolation

It has the following outputs:

1. OutputReady[1-bit],Bit used to indicate output is ready to be read out
2. r11[16-bit],The interpolated red pixel value
3. g11[16-bit],The interpolated green pixel value
4. b11[16-bit],The interpolated blue pixel value

1.3 stream m component

This component has following inputs :

1. Clock [1-bit] is the only input

It has the following outputs:

1. red [16-bit],This serves as the input (red in) for the input block
2. green [16-bit],This serves as the input (green in) for the input block
3. blue [16-bit],This serves as the input (blue in) for the input block

Takes as input, the 16 bit vectors corresponding to the r,g,b values of the pixels of 100x100 image from a file in.txt and streams these values into the input block component for interpolation to begin

1.4 input block component

This component has following inputs :

1. Clock [1-bit]
2. red in[16-bit],The red pixel value
3. green in[16-bit],The green pixel value
4. blue in[16-bit],The blue pixel value

It has the following outputs:

1. in ready [1-bit]
2. r$< i >$f[16-bit],The red pixel value
3. g$< i >$f[16-bit],The green pixel value
4. b$< i >$f[16-bit],The blue pixel value

Takes as input, three 16 bit vectors corresponding to the r,g,b values from stream m component and streams these values in sets of 5, to the multiply block for the calculations to be performed.

1.5 multiply component

This component has following inputs :

1. Clock [1-bit]
2. A [16-bit]
3. B [16-bit]
4. C [16-bit]
5. mA [17-bit] (=2^{16}-A)
6. mB [17-bit] (=2^{16}-B)
7. mC [17-bit] (=2^{16}-C)
8. 5 [16-bit] (r,g,b) sets, The red,green,blue pixel values

It has the following outputs:

1. 1 [16-bit] (r,g,b) set,The red(new),green(new),blue(new) pixel values

Takes as input, 5 sets of the 16 bit vectors corresponding to the r,g,b values of a pixel and its 4 neighbours and streams the new calculated values into the output block

1.6 output block component

This component has following inputs :

1. Clock [1-bit]
2. ri[16-bit],The red pixel value
3. gi[16-bit],The green pixel value
4. bi[16-bit],The blue pixel value
5. out ready[1-bit], To know if it should get ready to receive the bit-stream from calculation block (multiply component) or not

It has the following outputs:

1. Clock [1-bit]
2. rf[16-bit],The final interpolated red pixel value
3. gf[16-bit],The final interpolated green pixel value
4. bf[16-bit],The final interpolated blue pixel value
5. out ready[1-bit], To signal to the users that the final interpolated bit stream is ready to be streamed out

It takes input from the multiply component which streams the interpolated values in the order r[0][99], r[0][98],...,r[0][0] then r[1][0], r[1][1],... and so on.
These values are then stored in a single array of 100 elements (one array for each r, g and b) and as soon as the 101st value is received, it is ready to stream out the values in the order they were originally fed to the Outermost block i.e. r[0][0], r[0][1], ..., r[0][99] then r[1][99], r[1][98],.... and so on.
After streaming out any value, the space freed up can be used to store the next value, thus making it possible to stream out r,g,b values using 3 arrays of size 100 each.

1.7 subtract component

This component has following inputs :

1. Clock [1-bit]
2. iA[16-bit],The input value of A
3. iB[16-bit],The input value of B
4. iC[16-bit],The input value of C
5. reset[1-bit]

It has the following outputs:

1. A[17-bit],The final (2^{16}-A)/4
2. B[17-bit],The final (2^{16}-B)/4
3. C[17-bit],The value (2^{16}-C)/4

It takes as input, the values of A,B,C and returns the values (2^{16}-A)/4, (2^{16}-B)/4, (2^{16}-C)/4 to assist the multiply block with it's computation

1.8 bitshift component

This component has following inputs :

1. Clock [1-bit]
2. a[32-bit]
3. b[32-bit]
4. c[32-bit]
5. reset[1-bit]

It has the following outputs:

1. ao[16-bit],The estimated value of a/2^{16}
2. bo[16-bit],The estimated value of b/2^{16}
3. co[16-bit],The estimated value of c/2^{16}

It takes as input, three values a,b,c (32 bits each) and returns the estimated value of a/2^{16}, b/2^{16}, c/2^{16} to assist the multiply block with it's computation

1.9 Overall Working

As soon as the Start input goes high (it should stay high during the interpolation process), a process starts storing the inputstream of RGB into array. It reads in the RGB values from (0,0) to (0,99) into array-1. It then reads in (1,99) and stores in array-2 at 99th position. As soon as it reads in (1,99), since (0,99) can now be calculated, Readin is enabled. And the outputs are assigned corresponding values so as to be able to interpolate (0,99). Now the inputstream data is read into second array. As soon as we get (1,98), we can interpolate (0,98). So proceeding this way we reach (1,0). We have calculated (0,99) to (0,0) uptill now. As soon as we receive (2,0), process passes (1,0), (0,0), (1,1), (2,0) and a zero pixel to account for padding (no pixel to left of (1,0). Then we delete (0,0) and store (2,0) in its place, and so on proceed as before. So we need to maintain only 2*100 RGB values.
After Readin is enabled we pass, 5 RGB values as input to computational block (multiply component) in every clock cycle. As OutReady goes high, we read data from RedOut, GreenOut, BlueOut every clock cycle and store into Array-3. The first value read in will be interpolated value corresponding to (0,99). And next will be (0,98) and so on. As soon as we receive 100th value, i.e (0,0), OutputReady goes high and we start outputting the interpolated value, every clock cycle, starting with the value received last, i.e (0,0). As soon as we ouput (0,0), we read in (1,0) in its place and so on. Hence we require only one 100 sized array to buffer the output values. Suppose we stored (0,0) at 0th position in array-1. Then we first move the output from 0th to 99th and then back from 99th to 0th and so on.

1.10 Computations

The multiply component takes the in ready signal,r,g,b values of (x,y) pixel, and its four neighbours and computes the interpolated values, making use of bitshift, and subtract components in the process.
It then streams out the output to the output block.
The output block is signalled that the output is ready to be streamed out using the out ready bit signal

1.10.1 Calculations Done

The multiply component takes A,B,C and the 5 pixel values and computes the interpolated pixel value using the formulas given

$$r_{x,y} = A.r_{x,y} + (1-A).(r_{x+1,y} + r_{x-1,y} + r_{x,y+1} + r_{x,y-1})$$

$$g_{x,y} = B.g_{x,y} + (1-B).(g_{x+1,y} + g_{x-1,y} + g_{x,y+1} + g_{x,y-1})$$

$$b_{x,y} = C.b_{x,y} + (1-C).(b_{x+1,y} + b_{x-1,y} + b_{x,y+1} + b_{x,y-1})$$

The Calculator block has the following inputs:

1. A,B,C
2. 5 Red Inputs[16-bit],The red inputs to interpolate from five pixels
3. 5 Blue Inputs[16-bit],The blue inputs to interpolate from five pixels
4. 5 Green Inputs[16-bit],The green inputs to interpolate from five pixels

The Calculator block has the following outputs:

1. Red[16-bit],The red part of pixel for final output
2. Blue[16-bit],The blue part of pixel for final output
3. Green[16-bit],The green part of pixel for final output

Chapter 2

Important Assumptions and Constraints

We obviously have assumed certain things as all devices have limitations:

1. It is assumed that the evaluation of formula takes place in less than 5 clock cycles, so that we can effectively manage the 1 pixel value/clock cycle stream-rate , this is why we have more calculator blocks for parallel computation
2. We are assuming that A,B,C are sufficiently large or the sum of surrounding pixels for every pixel is sufficiently small that the r,g,b final values are in 16-bit range.
3. The Start signal mus remain high throughout the interpolation process
4. We assume that the clock speed is not very high to involve relativistic effects
5. We also assume that there is at least 60KB of space in memory, even though our implementation would require less than a 10th fraction of it for safety.

Chapter 3

Testing and Verification

3.1 Verification

In order to test, we will use 100x100 sample array. Using C++ program (sci.cpp), the expected output will be calculated.This output will then be matched by output given by the project code.
compile the sci.cpp file after setting the values of A,B,C and use the executable formed to generate output file $< f2.txt >$ from input file $< f1.txt >$ which has the 16 bit r,g,b values of pixels in the following format: r[0][0]
g[0][0]
b[0][0]
r[0][1]
g[0][1]
b[0][1]
...and so on The generated output values are also in the same order as the input.
The exact values obtained by C++ program will be of double data type. However, we are only interested in integers, hence nearest integer will be used.
To visualise the changes taking place, use the following :
1)new.py, this takes as input an image,atleast as big as 101x101 and produces output of r,g,b values as 16 bit vectors
USAGE: run python new.py (image name.jpg/png/bmp) ¿in.txt
and place it in $< mainProjectPath >$/input)
It is of the form(in 16bit binary)
r
g
b
... 2)rgb.py that converts the final result(binary r,g,b values) got from xilinx to comma-separated integer (r,g,b) pixel values,

Usage:Take output¡*image.txtfrom*¡mainProjectPath¿/output*andplaceitinothersfolderand runpythonrgb.py*output¡*image.txt* > *out.txt*
3)new1.py uses the r,g,b pixel values from the output of above file and makes an image out of it. Now we can visualise the before image and after image.
Usage: run python new1.py out.txt
4)sci.cpp that mimicks the total function of the project and creates the output so that we can compare and verify that what values we are streaming out are the correct ones(A simple diff can be used to compare outputs of cpp and xilinx) Usage: compile,give input file and store values in a output file, this is the final result, you can convert this to image by using rgb.py and new1.py
NOTE: TO GET IMAGE USE BOTH rgb.py and new1.py SEQUENTIALLY.

Test cases will be such that there are spikes (abrupt values) in the pixel values so that interpolation can be suitably tested. Corner cases can also be tested by making the spikes appear at any desired position, while editing any 100x100 image.

Chapter 4

Work Allocation

The Overall work consists of four main parts:

1 . Input handling

2 . Output handling

3 . Streaming the values into a file

4 . Calculation components' design

5 . Testing and verification

4.1 Himanshu Payal

Handled the buffering of output after the computations and designed the output mechanism.
Wrote the c++ verification program to ensure that the data streamed out matches the expected values.

4.2 Aakash Praliya

He designed the 3 components to handle the calculation part and wrote the python script rgb.py to format the pixel order

4.3 Amey Gupta

He designed the outstream component which directs the streams into an output file in the format required, debugged the code and wrote the required testbenches

4.4 Surender Singh Lamba

He designed a mechanism to take in the streaming input, store it by efficient use of resources and send it forward to computation block for working on input.