DESIGN AND IMPLEMENTATION OF AN ADAPTIVE EDGE ENHANCED COLOR INTERPOLATION PROCESSOR FOR IMAGE APPLICATIONS

[1]MEGHANA S, [2]DR. SUBODH KUMAR PANDA

[1]PG Student, VLSI & Embedded Systems, Dept. of ECE, B.N.M Institute of Technology, Bangalore, India, [2]Associate Professor, Dept. of ECE, B.N.M Institute of Technology, Bangalore, India

Abstract: *In this era of internet and multimedia communication, interpolating images to make them suitable for many advanced applications become a trend. Color interpolation is a very significant issue in digital image processing and has a wide variety of applications such as digital cameras, computer graphics, editing, online image viewing, biomedical field, remote sensing, etc. The digital cameras are developed by Charge Coupled Device (CCD) or Complementary Metal Oxide Semiconductor (CMOS) image sensor that can capture images by Color Filter Array (CFA) technique. A CFA used in the digital camera is a mosaic of spectrally selective filters, which allow only one color component to be sensed at each pixel. The missing two components of each pixel have to be estimated by methods known as Demosaicing. A color interpolation processor for image applications is designed and implemented. Color interpolation is a process used on images to convert a Bayer array into an RGB color model by approximating the pixel values at unknownpoints using known data.*

The proposed low complexity color interpolation processor is implemented using Matlab and synthesized using ModelSim simulator and also the area and delay calculations are done using Xilinx ISE 9.1i.

In this paper, an adaptive edge enhanced algorithm for edge detection, green interpolation, and red-blue interpolation is implemented to enhance the image quality in terms of PSNR. Finally, the Peak signal- to-noise-ratio (PSNR) is increased from 31.1436 to 37.7731 which is increased by 6.6295. This is because, as the value of PSNR increases the quality of the image is also increased. The overall delay reduction is 4.000ns. The overall area utilization is 44% which includes slices, LUTs and the gate count

Keywords: Camera, Color filter array (CFA), color interpolation, demosaicking, edge detector.

1. Introduction

Digital cameras have become very popularand they have almost replaced cameras even in the film industries too. When a digital image is recorded, the camera needs to perform a significant amount of processing to provide the user a viewable image. This processing includes white balance adjustment, gamma correction, compression filtering and many more. A very important part of this image processing chain is color filter array interpolation or demosaicking.

A color image requires at least three color samples at each pixel location. A camera would need three separate sensors to make these measurements. To reduce the size and cost, many cameras use a single sensor array with a color filter array. The color filter array allows only one part of the spectrum to pass to the sensor so that only one color is measured at each pixel. This means the camera must estimate the missing two color values at each pixel. This process is known as demosaicking. The most frequently used color filter array pattern is the Bayer CFA pattern as shown in figure 1.

A Bayer filter mosaic is a color filter array (CFA) for arranging RGB color filters on a square grid of photo sensors. The filter pattern is 50% green, 25% red, and 25% blue, hence is also called BGGR, RGBG, GRGB, or RGGB. Many efficient high-quality algorithms have been proposed for reconstructing the full RGB color from CFA images.

There are basically two main types of demosaicking algorithms that can be used for the interpolation of images, namely, Adaptive interpolation algorithms and Non Adaptive interpolation algorithms.

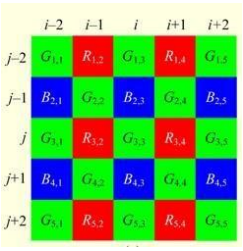


Figure 1 Bayer color filter pattern

Non-adaptive interpolation algorithms apply fixed patterns to each pixel without considering its other parameter as features, and edges of images. Some examples of Non-Adaptive interpolation algorithms are nearest-neighbor interpolation, bilinear interpolation, Bi-Cubic interpolation etc.

Adaptive interpolation algorithms employ spectral and spatial features available in the neighboring pixels in order to interpolate the unknown pixel as close to the original as possible. An example of thistype is Edge preserving interpolation.

The proposed algorithm in our project is a type of adaptive interpolation algorithm and by comparing the PSNR values that we get in this project with some of the Non- adaptive interpolation algorithms (see results and discussion section), we observe that adaptive interpolation algorithms give better quality images as their output compared to non-adaptive interpolationalgorithms.

The below Figure shows the flow chart of how the Bayer CFA image is reconstructed to full RGB format.

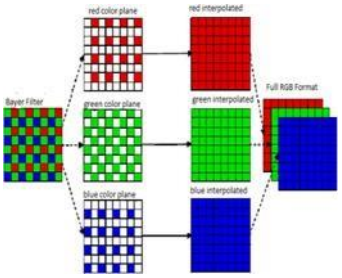


Figure 2 The flow chart of how the Bayer CFAimage is reconstructed to full RGB format

2. The proposed algorithm

Low-complexity edge detection, green color interpolation, and red-blue color interpolation techniques make up the suggested color interpolation algorithm. According to the relative locations and references of surrounding samples, each color is interpolated using a different methodology, as illustrated in Fig. 2, where the BRg I j) and RBg I j) indicate that the green color pixel g(i, j) was interpolated and prepared when it interpolates R(i, j) and B. (i, j).

2.1 low-complexity edgedetection

Edge detection technique is used to identify points for a given image where there exists a distinct change in the brightness or possess discontinuities. The boundary between two

image regions is called an edge. We need edge detection to capture important events and changes in properties of the given picture. We use edge detection for image segmentation and data extraction. Image segmentation helps us todetermine boundaries in images.

In this project, I have used the Sobel edge detection algorithm. It is a discrete differentiation operator, calculating the approximate gradient of image intensity function. The basic principle of the algorithm is based on the following equations. The difference between the vertical (DV) and horizontal (DH) directions is used to find the edge information.

Vertical Direction:

$$DVi, = |RBi{-}1,{-}1 - RBi{+}1,j{-}1 \,| + |Gi{-}1,j - Gi{+}1, \,| + |RBi{-}1,{+}1 - RBi{+}1,j{+}1 \,| \quad (1)$$

Where G is the pixel in green color and RB is the pixel in red or blue color in the CFA images.

Horizontal direction:

$$DHi, = |RBi{+}1,{+}1 - RBi{+}1,j{-}1 \,| + |Gi,j{+}1 - Gi,{-}1 \,| + |RBi{-}1,j{+}1 - RBi{-}1,j{-}1 \,| \quad (2)$$

Total difference:

$$TDi, = DHi,\mathrm{j} + \mathrm{DVi,j} \quad (3)$$

BR (i-1, j+1)	G (i-1, j)	BR (i-1, j-1)
G (i, j+1)	RBg (i, j)	G (i, j-1)
BR (i+1,j+1)	G (i+1, j)	BR (i+1, j-1)

Figure 3 A basic 3x3 matrix of the input imageused for calculation of an edge

The above equations can be easily comprehended when we see Fig.1. Which represents a simple 3x3 matrix of input CFA pixels of any given image. However, it should be noted that when we pick the first 3x3 matrix, we should pad zeros around the edges of the entire input CFA image matrix before we do any sort of processing so that we receive the output at the pixels located at the corners and along the edges of the input image.

After we run the Edge detection technique,we get an output as:

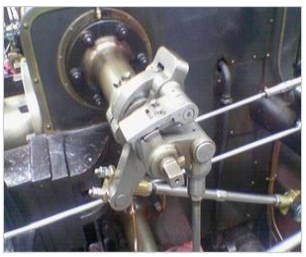
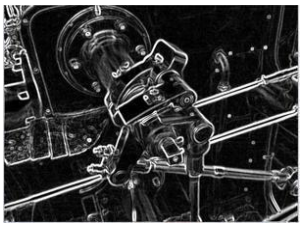


Figure 4 (a) Input image



(b) Edge detected output image

After we run edge detection technique, we get a black and white image as output whichhas white edges traced along all the edges, of the input image, on a black background. The Sobel edge operator we used was:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

Where, G_x and G_y are two images where each point contains the DV and DH respectively. A is referred to as the source image. Here, we are convoluting the input image with the Sobel edge operator. The "*" refers to convolution operation. The above matrices have 2 different filters which can be separated and can be written as:
By reducing it to this format, the number of computations can be reduced leading to enhanced speed and efficiency.

$$G_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * A \right) \quad \text{and} \quad G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * A \right)$$

2.2 Green color interpolation

Bayer Filter CFA is used in this project. The Bayer filter contains 50% of Green information, 25% of Red and 25% of Blue information and is overlaid on an image sensor in most modern digital Cameras. It is arranged in such a way that it has green-red information in odd rows and blue-green Information in even rows. Several methods, including edge detection, green color interpolation, and red or blue color interpolation approaches, are utilized to enhance the quality of the interpolated images. Different interpolation techniques are used for each color depending on its relative location and references to nearby samples. We go over the green color interpolation method in this part.

Let us first consider the diagrams shown below to understand the basics of green color interpolation.

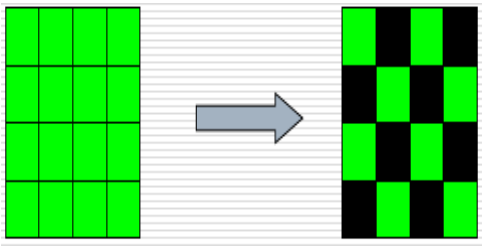


Figure 5 Formation of Green Color Plane

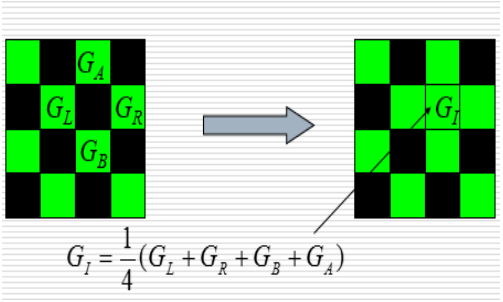


Figure 6 Estimation Using Neighboring Values

Every digital Image has 3 color planes namely green, red and blue. The above diagram (3.2.1) shows the formation of the green plane. After the green plane is formed, one of the missing pixel values (GI in this example) will be calculated using neighboring pixel values as shownin Figure 3.2.2.

There are a lot of methodologies to approximate a particular green pixel value by knowing the values of other pixels. This process of approximating the values at unknown points using known data is called interpolation.

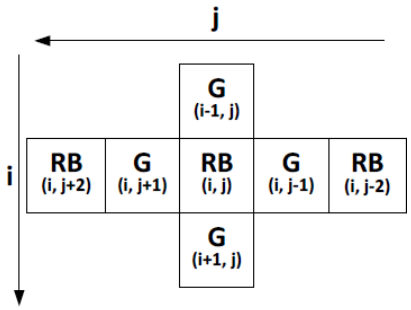


Figure 7 The matrix form

When R(i, j) and B are interpolated, the values BRg I j) and RBg I j) show that the green colour pixel g I j) was prepared(i, j).

Figure 3.2.4 demonstrates that there are significantly more reference adjacent pixels in the horizontal direction than there are in the vertical direction. By using a 25% weighting in the vertical direction and a 75% weighting in the horizontal direction, it is possible to interpolate the green hue of the pixels.

In the same way, there are two alternative reasons for interpolating G to rebuild the green colour in an image in CFA format (i, j). According to Figure 2.a, the location of G I j) might be either the colour pixel of B I j) or R I j). Since P(i, j) represents the pixel at the positions of I and j in the yand x coordinates, respectively, the interpolated pixel G(i, j) is described as G(RB)(i, j) where the location of P(i, j) is R(i, j) or B(i, j) in the original CFA pictures

One of the three green interpolation models is adaptively chosen based on the values of TD, DH, and DV (described in the preceding sections) and the edge information around the pixel G I j).

Following are the three green interpolation models:

1. without edge augmentation
2. Improvement of the edges in thehorizontal direction
3. Edge improvement along the vertical axis

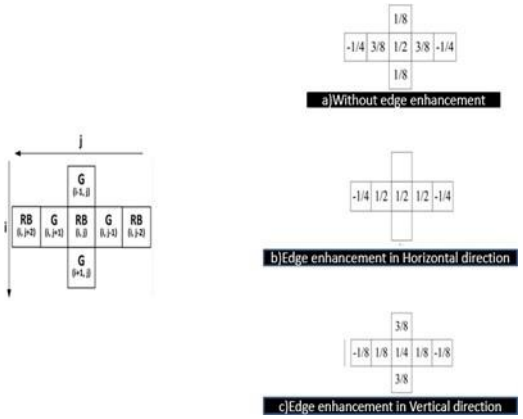


Figure 8 consists of the standard matrix on the left side and on the right side are the parameters of green interpolation: (a) Without an edge enhancement model, (b) Edge enhancement in horizontal directionmodel and (c) Edge enhancement in verticaldirection model.

(a) Without edge augmentation

The value of G (RB) I j) can be determined by the without edge enhancement model as follows if the value of total difference (TD)is smaller than the threshold value:

$$Gi,(RB) = 3/8\ (Gi,-1 + Gi,j+1) + 1/8(Gi-1,j+ Gi+1,)+RBi, - 1/2\ (1/2\ (RBi, + RBi,-2) + 1/2\ (RBi,j + RBi,j+2)) \quad 4$$

The RB(i, j) is the original sampled value of P(i, j) in the CFA image, which could be R(i, j) or B(i, j) depending on the location of P(i, j).

b) Edge enhancement in the horizontaldirection

The value of G(RB)(i, j) can be generated by the edge enhancement in the horizontal direction model as follows if the value of the total difference (TD) is greater than the threshold value and DH is smaller than DV:

$$Gi,\ (RB) = 1/2\ \ (Gi,-1 + Gi,j+1) + RBi,j - 1/2\ \ (\ 1\ 2\ (RBi,j + RBi,j-2)\ \ + 1/2\ (RBi,\ +RBi,+2)) \quad 5$$

(c) Edge enhancement in vertical direction

Edge enhancement in the vertical direction model can calculate the value of G(RB)(i, j) as follows if the total difference (TD) value is greater than the threshold value and the difference in height (DH) value is greater than the difference in distance (DV):

$$Gi,\ (RB) = \frac{1}{8}(Gi,-1 + Gi,j+1) + \frac{3}{8}\ (Gi-1,j + Gi+1,) + \frac{1}{2}\{RBi, - \frac{1}{2}(\frac{1}{2}(RBi, +RBi,-2)+\frac{1}{2}(RBi,j + RBi,j+2))\} \quad 6$$

Notice that the equations (4), (5), and (6) can be derived by multiplying the standard matrix (Figure 3.2.4) with the following three parameters of green interpolation: (a) without edge enhancement model, (b)Edge enhancement in horizontal direction modeland (c) edge enhancement in verticaldirection model.

The characteristic of the compensation for green color is a spatial sharpening filter, as it is evident from an analysis of the characteristics of these three green color interpolation models. Effectively reducing the blurring effect is this spatial sharpening filter. The suggested adaptive edge enhancement technique may also effectively improve the edge information.

2.3 Red and blue color interpolation

Let us first understand the basics of red color interpolation. Consider the diagrams below:

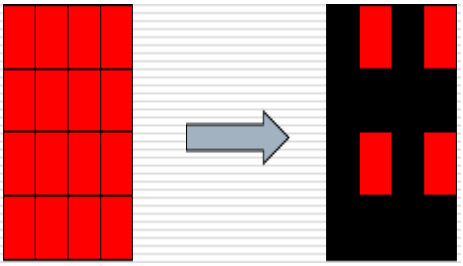


Figure 9 Formation of red color plane

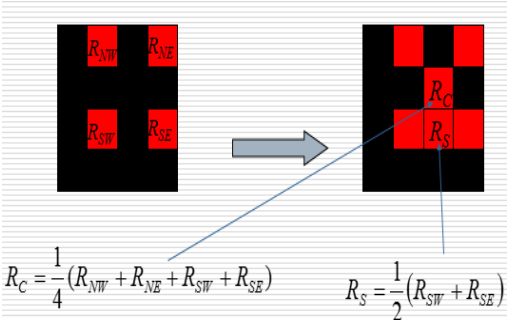


Figure 10 Estimation using neighboring values

After the red plane is formed, one of the missing pixel values (Rc in this example) will be calculated using neighboring pixel values as shown in Figure 2.5.b. Notice that Rs in the Figure 2.5.b can also be approximated with just two known values. Every aspect of approximation depends on the position of each pixel in the plane.

To recreate the red and blue colors in CFA design picture, it is important to utilize the data of the four neighboring green colors.

Consider the following matrices:

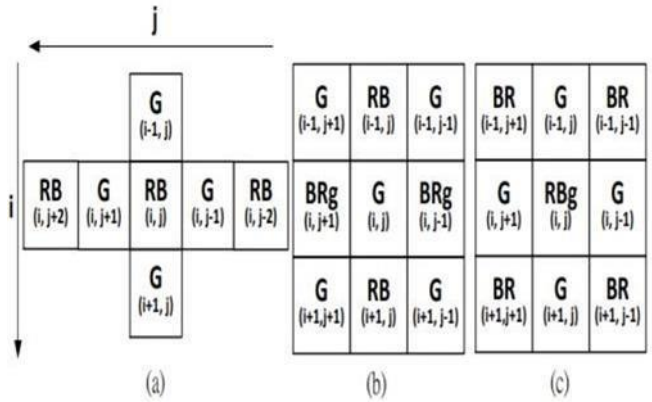


Figure 11 Relative locations and reference neighboring samples of G(i, j) in CFA format

Let's now examine how this project's red- or blue-color interpolation functions. When the pixel in CFA format is R(i, j) or B, the aforementioned figures display the relative locations and adjoining samples of G(i, j+1), G(i-1), G(i, j-1), and G(i+1, j) (i, j).

There are three red and blue interpolation models, and one of the three models can be selected adaptively to improve edge recognition.

1) Without edge enhancement.
2) Improvement of the edges in a horizontaldirection.
3) Enhancement of the edges in a verticaldirection.

There are three parameters used to calculate which model has to be used adaptively, namely:

1) TD-total difference.
2) TH-difference in the horizontaldirection.
3) TV-difference in the vertical direction.

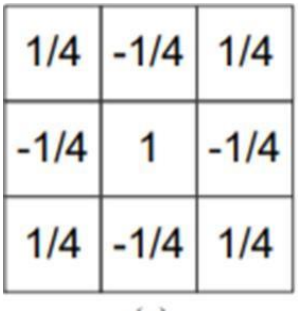
a) Model 1: Red and blue interpolationwithout edge enhancement

If the value of total difference (TD) is less than a threshold value, the value of R(B)(i, j) or B(R)(i, j) can be calculated by the without edge enhancement model as:

$$RBi, (BR) = \frac{1}{2}(RBi-1,-1 + RBi-1,j+1 + RBi+1,j-1 + RBi+1,+1) + gi,j - \frac{1}{4}(Gi-1,j + Gi+1,j + Gi,j-1 + Gi,j+1) \quad 7$$

where g(i, j) is produced by green color interpolation.

The below figure shows the parameters of red and blue color interpolation without edge enhancement:



1/4	-1/4	1/4
-1/4	1	-1/4
1/4	-1/4	1/4

Figure 12 without edge enhancement model

b) Model 2: Interpolation of the colors red and blue with horizontal edge enhancement.

The value of R(B)(i, j) or B(R)(i, j) can be produced by the edge enhancement in the horizontal direction model as follows if the value of TD is greater than the threshold value and DH is smaller than DV:

$$RBi,(BR) = \frac{1}{4}(RBi-1,-1 + RBi-1,j+1 + RBi+1,j-1 + RBi+1,j+1) + gi, -\frac{1}{8}(3 * (Gi-1,+ Gi+1,j) + Gi,j-1 + Gi,j+1) \quad 8$$

where g(i, j) is produced by green colorinterpolation.

The below figure shows the parameters of red and blue color interpolation with edge enhancement in the horizontal direction:

1/4	-3/8	1/4
-1/8	1	-1/8
1/4	-3/8	1/4

Figure 13 with edge enhancement in horizontal direction model

c) Model 3: Red and blue interpolation with edge enhancement in the vertical direction.

The value of R(B)(i, j) or B(R)(i, j) can be produced by the edge enhancement in the vertical direction model as follows if the value of TD is greater than the threshold value and DH is greater than DV:

$$RBi,(BR) = \frac{1}{4}(RBi-1,-1 + RBi-1,j+1 + RBi+1,j-1 + RBi+1,j+1) + gi, -\frac{1}{8}(Gi-1,+ Gi+1,j + 3 * (Gi,j-1 + Gi,j+1)) \quad 9$$

where g(i, j) is produced by green color interpolation

The below figure shows the parameters of red and blue color interpolation with edge enhancement in the horizontal direction:

1/4	-1/8	1/4
-3/8	1	-3/8
1/4	-1/8	1/4

Figure 14 with edge enhancement in verticaldirection model

By analyzing the parameters of these three interpolation models having three R and B colors, it can be seen that the characteristics of reconstruction for R and B colors are Laplacian filters.

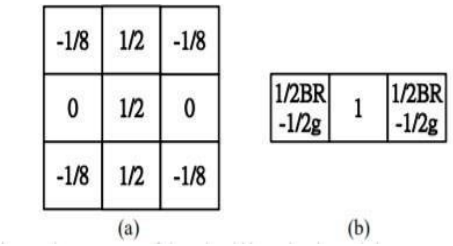


Figure 15 The parameters of the red and blue colorsinterpolation at G(i, j)

(a) Laplacian sharpening filter

(b) spatial sharpening filter

It is possible to calculate the red or blue color reconstructed pixel RB(i, j), whose top and bottom pixels are RB(i-1, j) and RB(i+1, j), respectively, at G(i, j), as follows:

$$RBi, (G) = 1\ 2\ (RBi{-}1, + RBi{+}1,) + 1\ 2\ Gi, \ -\ 1\ 8\ (Gi{-}1,{-}1\ Gi{-}1,j{+}1\ {+}Gi{+}1,j{-}1\ {+}Gi{+}1,j{+}1) \quad (10)$$

The red or blue color reconstructed pixel RB (i, j) where its left and right pixels are RB (i-1, j) and RB(i+1, j) at G(i, j) can be calculated as:

$$BRi, (G) = 1\ 2\ (BRi,{-}1 + BRi,j{+}1) + Gi,j - 1\ 2\ (gi,j{-}1 + gi,j{+}1) \quad (11)$$

3. Implementation

Initially, we need to modify the input image to get the desired CFA image. We use MATLAB for this conversion. Firstly, we resize the image and the convert it to the CFA image. This CFA image is then converted to a text file where each pixel is represented by a hexa-decimal value. Now, this text-file is then read by the VHDL codes. While we convert the input image to the CFA image, each CFA pixel holds the information regarding the color of the corresponding pixel. This is the reason why we do not lose the color part of the information.

After using the color interpolation techniques to enhance the quality of the image, we reconstruct it back using an in- built function in MATLAB. The VHDL codes create another text-file which is then read by MATLAB to reconstruct the final output image and then displays it. We can also see the PSNR as one of the outputs present in the command window.

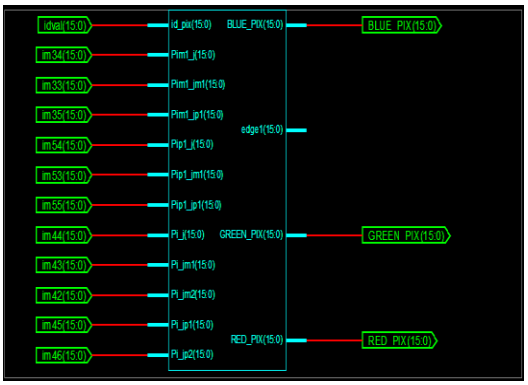


Figure 16 General RTL schematic of the Color interpolation processor

In the above VLSI design of the color- interpolation processor, 7 main blocks are used namely: edge detector, register bank, three red-blue interpolators, a controller and a green interpolator. These architectures cover the three main algorithms used which is the red-blue interpolator, low-complexity edge detection and the green interpolation techniques. For the color interpolators, an anisotropic weighting model was used. Now, the quality can improve since more information can be acquired by horizontal component of the information from the edge detector than the vertical component, eliminating the need of line buffer memory. This reduces the memory requirement. The proposed filter can produce images of good quality due to the use of various colors of actual CFA image and double-interpolated green pixels.

Register block

The register block is constructed so that just one-pixel value from memory is read during each cycle, and then output a total of fifteen values throughout the course of fifteen cycles. The color interpolation uses these pixel values as input. The processor will access the memory with this register bank by using pixel-in and pixel-out.

Edge Detector

The register bank supplies the edge pixels as input signals. The edge detector has eight inputs, and three outputs. The red, blue, and green color interpolators' first model (RB_M1) and the green color interpolator produce the values of TD, DH, and DV at the point of P after being processed by the edge detector (i, j). The controller has access to the edge detector's output signals as well. Information regarding the edge intensity is provided by the TD output signal. The other two output signals, DH and DV, give the edges' direction information. These pixel values are fed as input to Green interpolator and also red-blue interpolators.

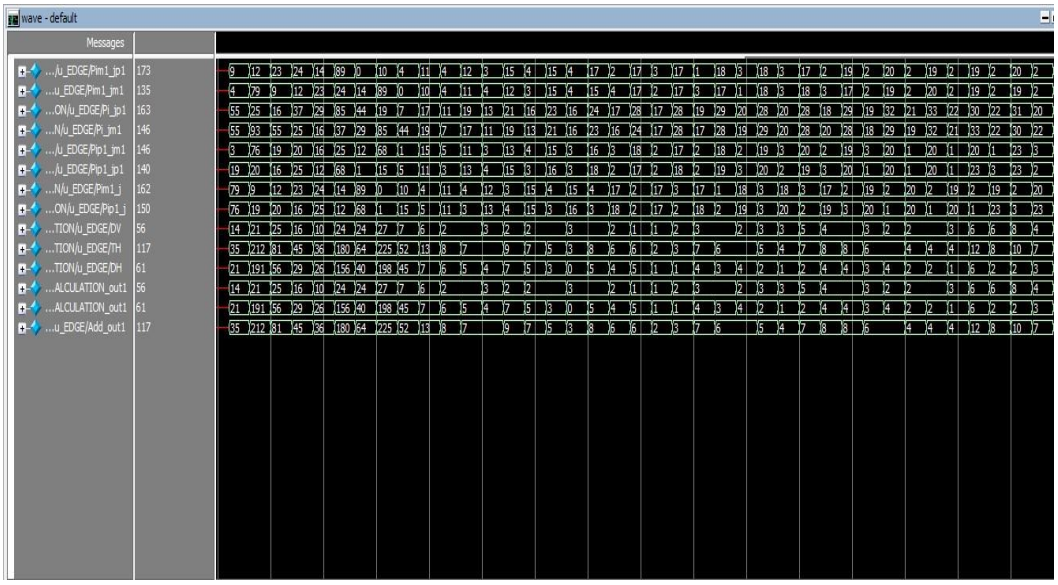


Figure 18 pixel values of edge pixels obtained after simulation

We used the Sobel Edge Detector to create the edge detector in our processor. It is straightforward to put into practice and is adequate as a low-complexity edge detector. Let's now have a look at a 3x3 arrangement of pixels as seen below:

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

The partial derivatives of the given matrices are written as:

$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

Constant ‘c’ is such that it is the co-efficient of pixels closer to the [i, j] pixel and by setting c=2, we get the matrices which are used as the Sobel operators.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The above matrices represent Horizontal and vertical directions respectively. After this phase, the pixels are off to the color interpolators.

Green Color Interpolator (G Interpolator)

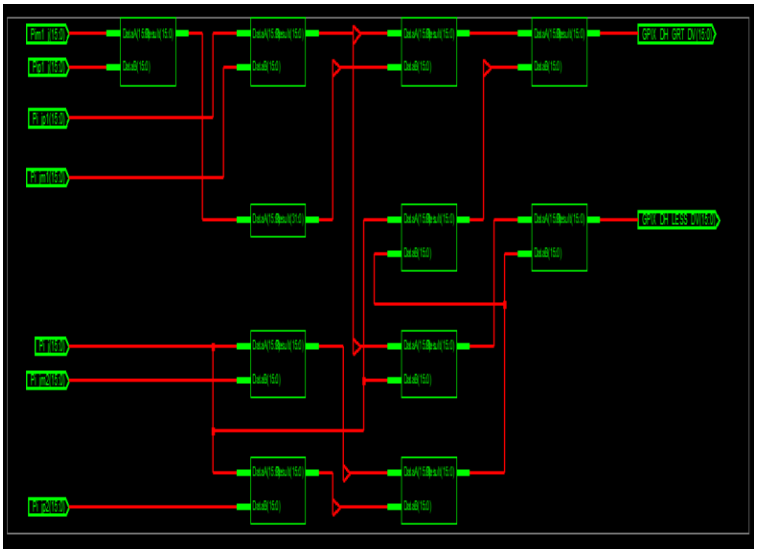


Figure 19 the RTL schematic of Green Color Interpolator (G Interpolator)

Analysis of equations (4), (5), and (6) presented in the green interpolation section reveals that all input signals, with the exception of the parameter values, have the same values. The green interpolator design's schematic is depicted in Figure 5.3.1. As we can see, the design consists of four multiplexers, one 3-bit shifter, four 1-bit shifters, eight adders, and one subtractor.

This architecture can be modified throughout each processing cycle as a result of the equations (4), (5), and (6) stated in the section on green interpolation, depending on the multiplexer choice features like high performance, great flexibility, and low cost. The controller delivers control signals to the multiplexer based on the values of TD, DH, and DV. This reconfigurable method is utilized, and as a result, the green interpolator that is created has good values.

In this architecture, three registers were used to simplify the critical path and enhance design performance. The output of G interpolator is kept in the first register (i, j). This register provides the input signal for the red-blue interpolation models 1 and 2.

The output of g(i, j) that generates the input signal g(i, j-1) for the red-blue interpolation model 3 is kept in the second register. The edge data for the TD, DH, and DV is primarily stored in the third register, which is the last one. Keep in mind that this is a pipeline architecture.

Red and blue Color Interpolator (RB Interpolator):

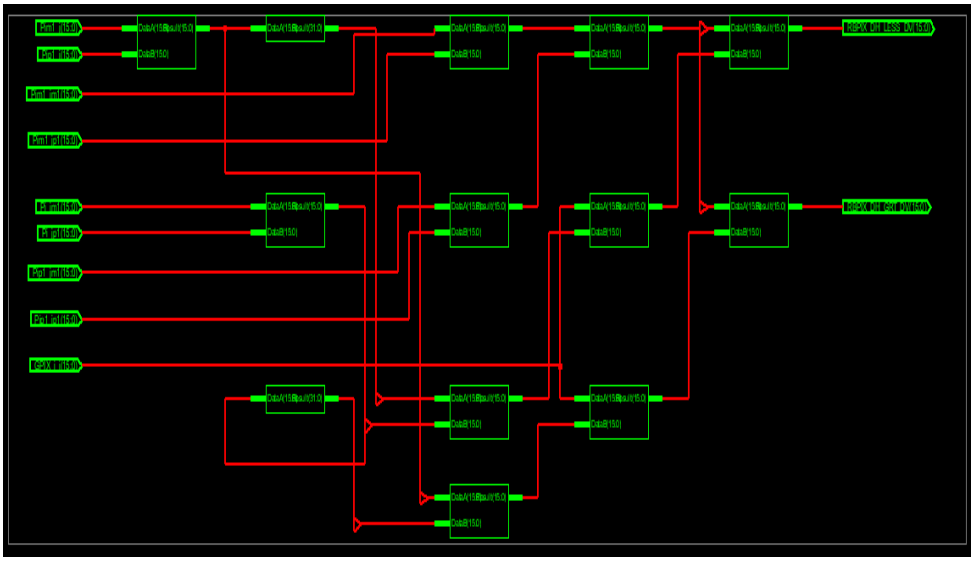


Figure 20 RTL schematic of M1 interpolator

All of the input signals must be the same, with the exception of the parameter values, in order to examine equations (7), (8), (9), (10), and (11) covered in the red and blue interpolation section. As a result, the hardware architecture of the red and blue interpolator must be designed using the reconfigurable technique.
Red and blue interpolation without edge enhancement as well as edge enhancement in the horizontal and vertical directions can all be evaluated using this model. The RB M1 interpolator consists of nine adders, one subtractor, two multiplexers, and five shifters.

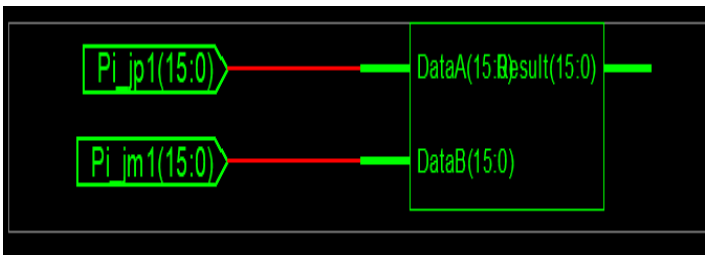


Figure 21 RTL schematic of R2 interpolator (left-right process)

The model that is used to rebuild upper and lower pixels is the RB M2 interpolator, which is made up of five adders, one subtractor, and two shifters.

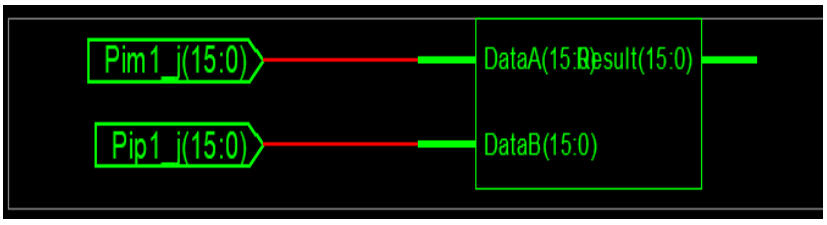


Figure 22 RTL schematic of R3 interpolator (upper_down process)

RB M3 interpolator functions similarly to RB M2. This model is used to reconstruct the left and right pixels and is composed of three adders, one subtractor, and one shifter.
The Green, Red, and Blue interpolation equations are constructed with parameters of 1/2, 1/4, 1/8, and 3/8. By substituting adders and dividers for multipliers and dividers, it creates a base for the VLSI implementation that is inexpensive to use.

Controller
A finite state machine (FSM) sequential circuit implements the controller. It can deliver reconfigurable control signals to change the interpolators' architecture and provides the control signals to the multiplexer for choosing the input for the interpolators. To optimize pixel-in and pixel-out speed, the controller must keep an eye on how much RAM is being accessed for input and output data. The color interpolation processor also has a high throughput and performance.

4. Results and discussion

The CPSNR (Clock Peak Signal to Noise Ratio) and RMSE (Root Mean Square Error) values by the original color image were computed using the Matlab (Mathworks, Natick, MA) tool to compare the performance of the prior low-complexity color interpolation algorithms with this work. Five color images of different resolutions have been taken for comparison. In color images it is difficult to visualize each and every pixels as in CFA image. To acquire the low resolution of color information of the image seen CFA (color filter array) is used. This CFA is placed on the top of the monochrome image sensor, usually a charge coupled device (CCD). The choices of CFA critically influences the accuracy of the single sensor imaging pipeline. In this project, to construct the CFA images, first the original color image is resized to 256×256 and then processed as a Bayer color filter array.
The input color images considered for constructing the CFA images are as shown in figure 6.1.

Figure 23 (a), (b), (c), (d), (e) reference images for testing

The input color image of size 2667×4000 is considered for interpolation is shown in figure 6.2. The input color image is resized to 256×256 using Matlab as shown in the figure 6.3.

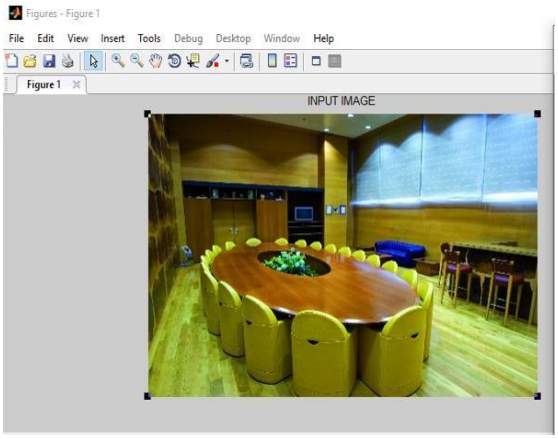


Figure 24 the original input color image

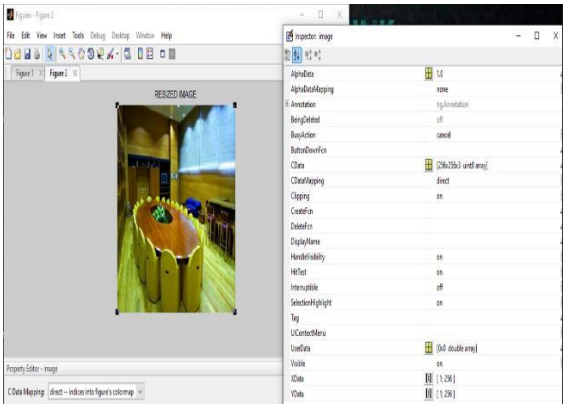


Figure 25 The resized image and its features

The resized color image is given as input to Matlab to obtain CFA (or Bayer) as shown in figure 6.4.

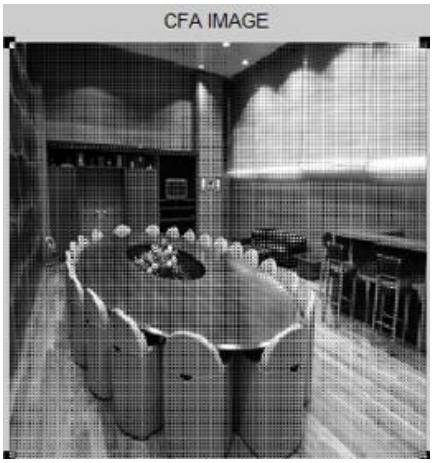


Figure 26 CFA image

The interpolation process is carried out using a hardware processor. Hardware description language Verilog is used to describe the interpolation processor hardware. The implementation is verified using Modelsim simulator.
The maximum number of rows and columns are configured and each and every pixel value of particular CFA image is loaded into Modelsim simulator via Matlab

using hdldaemon (socket, 4999). Using Modelsim simulator the horizontal and vertical values of pixels are estimated. The estimated values are stored in Register blocks.

6.1 Register block

The register block is constructed in Modelsim simulator so that just one-pixel value from memory is read during each clock cycle. A total of sixteen clock cycles are required to store pixel values of CFA image. For interpolation these pixel values are used as input. The processor will access the memory with this register block by using pixel-in and pixel-out.

6.2 Edge Detector

The edge detector takes the values from register block as input signals. The edge detector has eight inputs, and three outputs. After processing the pixels the edge detector produce values at the point of P (i, j) with respect to horizontal and vertical calculations of pixels. Then DH (Direction Horizontal), and DV (Direction Vertical) values of pixels are obtained and then sum of both DH and DV gives the TD (Total Difference). The two output signals, DH and DV gives the edge direction information. These pixel values are fed as input to Green interpolator and also red-blue interpolators.

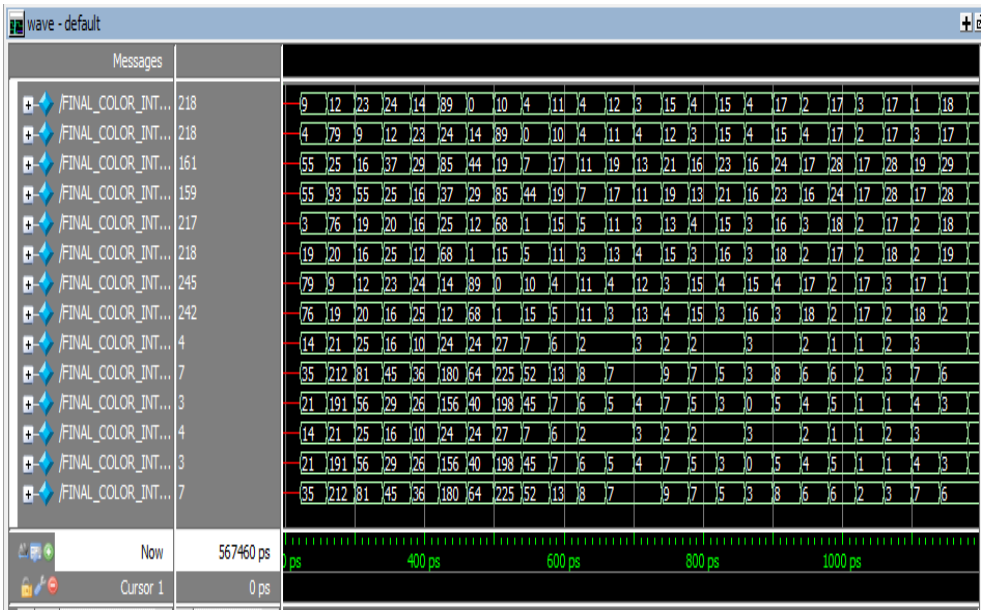


Figure 27 pixel values of edge pixels obtained after simulation

6.3 Green color interpolator (G interpolator)

The input signals are identical when analyzing equations (4), (5), and (6), with the exception of the different weightage values. In this project, the architecture of the green color interpolator was designed using a reconfigurable technique.

Eight adders, one subtractor, four multiplexers, and five shifters are the inputs considered for green color interpolation. Appropriate control signals are sent to multiplexers based on the values of TD, DH, and DV. The suggested green color interpolator also employs reconfigurable technology and benefits from low cost, and high performance.

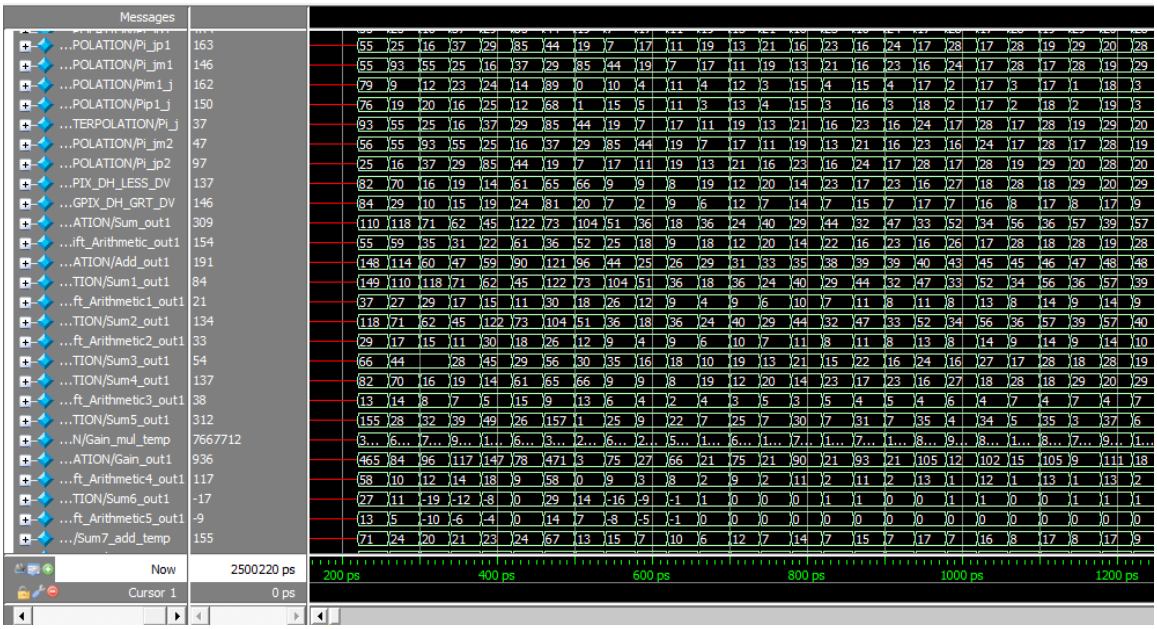


Figure 28 pixel values of green interpolation

The Green interpolator's output is kept in the first register g (i, j). It gives the RB_M1 (Red-Blue Model 1) and RB_M2 (Red-Blue Model 2) with input signal g (i, j) modules. The value of g is kept in the second register (i, j) that gives the RB_M3 (Red-Blue Model 3) the input signal g (i, j-1).

The edge information is lastly stored in the third register and the DH, DV, and TD. These three register values are added, and this implementation is the design of a pipelined architecture. It is succeeded in reducing the critical path and supplying interpolated R and B interpolators g (i, j) and g (i, j-1) to enhancing quality.

The Green color interpolated images have the same values except the different weightage. This interpolation is mainly done because of improving the performance of the image and also to shorten the critical paths. The pixel values obtained after simulation are stored in the form of text file. These values are again read by the matlab and the green color interpolated image is obtained.

Figure 29 Green color interpolation image

6.4 Red-Blue color interpolators

All input signals must be identical for equations (7), (8), and (9), with the exception of the different weightage values. Therefore, the hardware architecture of the red and blue colors interpolator can be designed using the reconfigurable technique.

6.4.1 RB_M1 interpolator (Red-Blue Model 1)

Figure 6.4.1.1 shows the pixel values of Red-blue Model 1 interpolator using equation (9)., which is composed of 9 adders, 1 subtractor, 2 multiplexers, and 5 shifters are used. The design can be changed to fulfill the requirements of Equation (7). (8), or (9) depending on the TD, DH, and DV values for each processing loop. All these pixel values are combined together and the red-blue interpolated image is obtained.

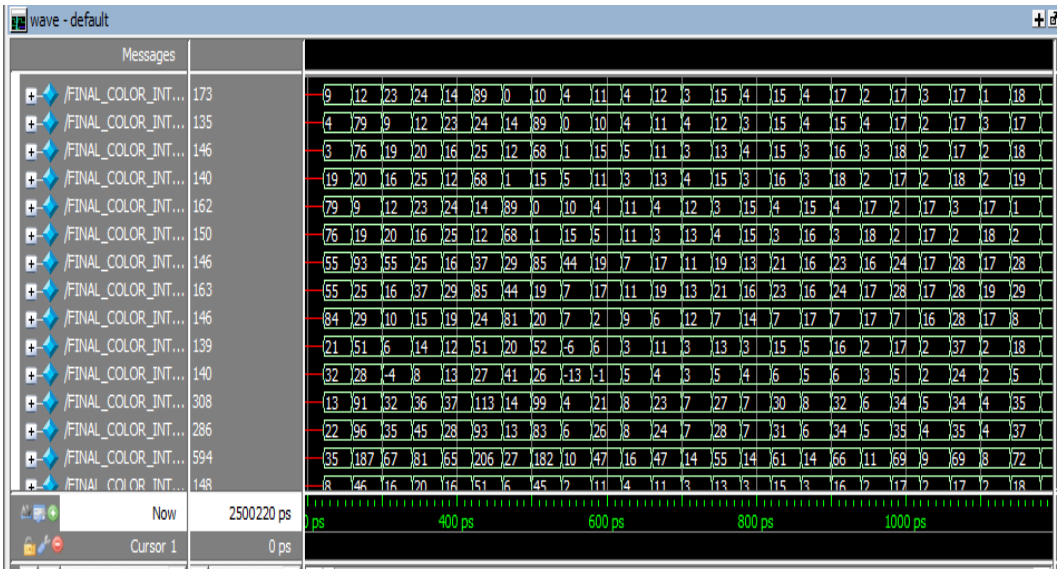


Figure 30 pixel values of RB_M1 interpolator

6.4.2 RB_M2 interpolator (Red-Blue Model 2)

Figure 6.4.2.1 shows the pixel values of Red-blue Model 2 interpolator using equation (10). The RB_M2 interpolates the pixels in horizontal direction. The RB_M2 interpolator has five adders, one multiplier, and one shifter, a subtractor, and two.

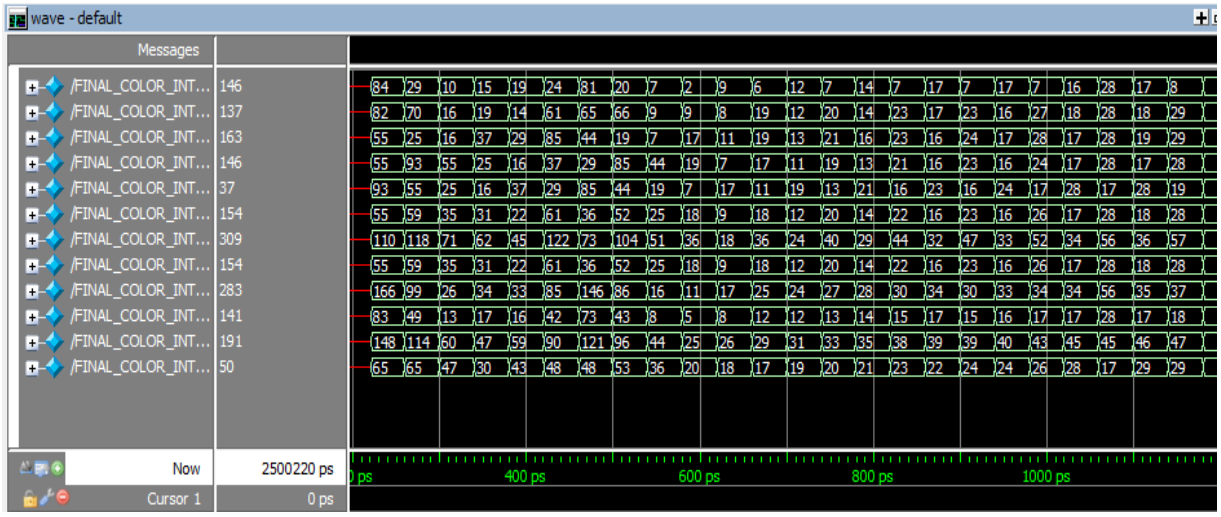


Figure 31 pixel values of RB_M2 interpolator

6.4.3 RB_M3 interpolator (Red-Blue Model 3)

Figure 6.4.3.1 shows the pixel values of Red-blue Model 3 interpolator using equation (11), the RB_M3 interpolates the pixels in vertical direction. The RB_M3 interpolator has three adders, and, one subtractor, and three 1×shifter**.**

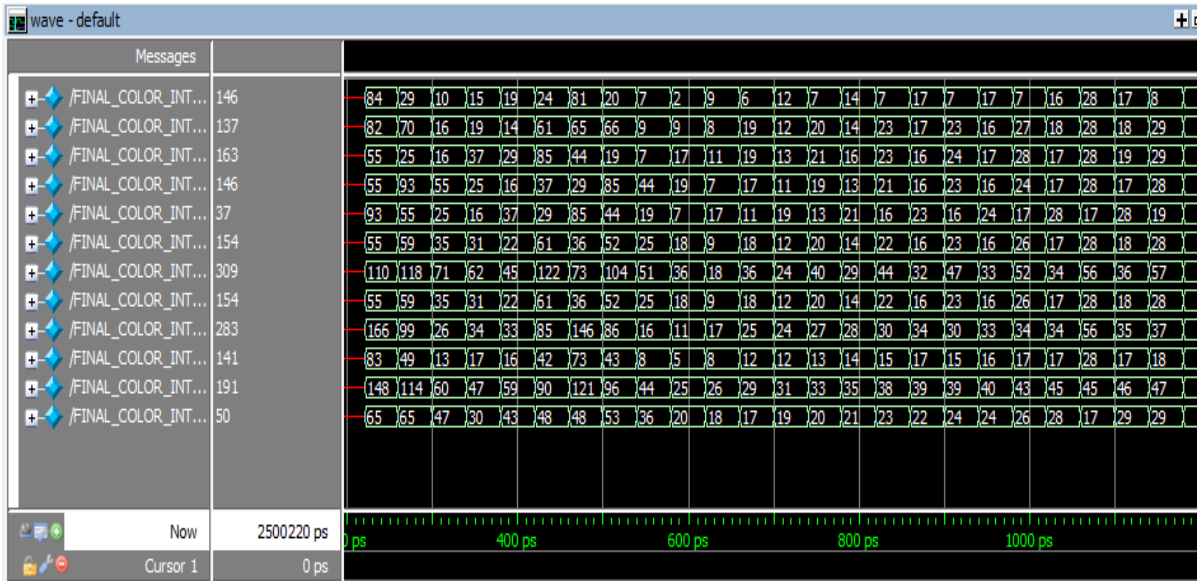


Figure 32 pixel values of RB_M3 interpolator

Figure 33 Red color interpolation image

Figure 34 Blue color interpolation image

The pixel values obtained from modelsim simulator after interpolation of Red-Blue interpolator models are stored in the form of text file. This text file is read by the matlab and the red and blue color interpolated image is obtained as shown in figure 6.6.1.4 and 6.6.1.5.

6.5 Controller

The controller gives control signals for the multiplexer for choosing input data for the interpolators. To fit the performance of both input and output pixels, the controller must also keep track of the memory access for its input and output data. Finally, great performance and high throughput are achieved using the suggested color interpolation processor**.**

After interpolation of all three models of Red-blue and green interpolation, the obtained pixel values are stored in Modelsim simulator. These stored values are again read by the Matlab and the color interpolated image is obtained.as shown in the figure 6.6.1.7.

Figure 35 Color interpolation image

In order to compare the values of RMSE and PSNR values of the previous values with this work, the Matlab tool is used to compute the values of both RMSE and PSNR.

To compare with the previous values with respect to performance, low complexity color interpolation processor is implemented in this work. The Matlab tool is used to compute the values of PSNR by considering 5 images and also the interpolated images.

Table 1 Comparison of PSNR values for the previous values

Images	Previous values of PSNR	PSNR
1	34.446	37.9525
2	31.760	35.5630
3	33.246	38.2763
4	27.476	38.5884
5	28.790	38.4856
Average PSNR	31.1436	37.7731

Compared to the previous values of PSNR the present values obtained are highly efficient. This is because, as the value of PSNR increases the quality of image is also increased.

AREA DELAY COMPARISON

The below table represents the comparison of the area and delay with the previous design. Compared to the previous design values the proposed design has increased gate count, and reduced Lookup table (LUTs). The maximum delay is also reduced from 91.53ns to 44.64ns which improves the quality of images and also the performance.

Table 2 Comparison of area and delay of the conventional and proposed design

S. No	Method Name	Area			Delay		
		Slice	Gate Count	LUT	Max Delay	Gate Delay	Path Delay
1	Previous Design	705	21575	1305	91.539ns	40.606ns	50.933ns
2	Proposed Design	516	21635	942	44.648ns	28.884ns	15.764ns
	Differences between 1 & 2	189	60	363	46.891ns	11.722ns	35.169ns

AREA

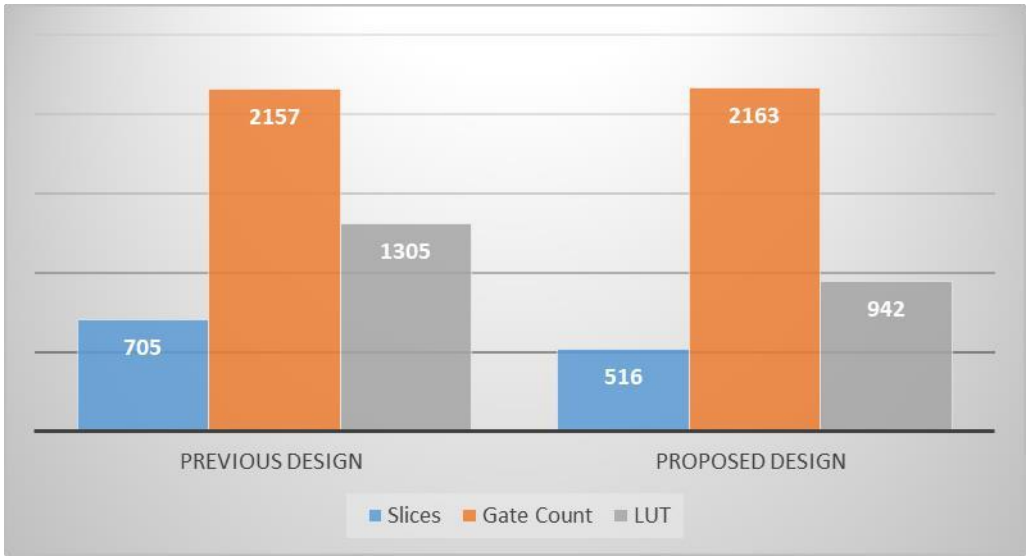


Figure 36 Comparison of area with the previous design

The gate count is increased from 21575 to 21635 where the increased gate count is 60. LUTs are reduced from 1305 to 942 where the reduced LUTs are 363. The slices are reduced from 705 to 516 where the reduced slice count is 189.

DELAY

The gate delay of the previous design is 40.606ns and proposed design is 28.884ns. So, the gate delay is reduced to 11.722ns. The maximum delay of the previous design is 91.539ns and proposed design is 44.891ns. The maximum delay is reduced to 46.891ns. The path delay of previous design is 50.933ns and

proposed delay is 15.764ns. The path delay is reduced to 35.169ns. The reduction in delay avoids the noise present in the images.

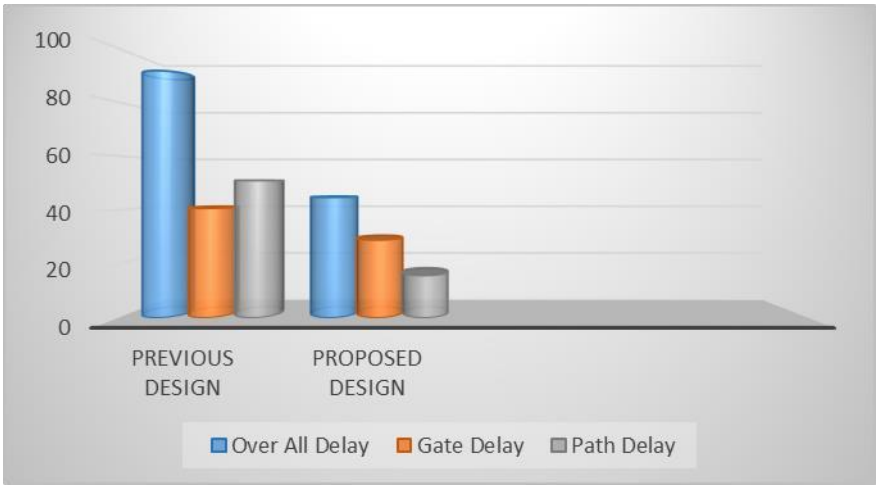


Figure 37 Comparison of delay with the previous design

5. Conclusion

In this project, Matlab is used to convert a color image into a Bayer image (CFA) using a filter array to separate RGB pixels which is necessary for interpolation of RGB pixels separately. The color interpolation specifies the color space for gradient interpolation, color animations, and alpha compositing.

The CFA image is used as input for the Verilog implementation of edge detection, red-blue, and green interpolators, as well as horizontal and vertical computations. Each pixel's value is transformed to a hexadecimal value during simulation. This attribute selects between color-related RGB-space operations.

The color interpolated image is obtained once the simulation is complete, and the RMSE and CPSNR values are determined. The cost function, which is frequently the mean square error between the desired signals and the output of the adaptive filter which serves as the optimization criterion. If interpolation process is carried out using hardware, then the computation will be faster.

An adaptive edge enhanced color interpolation method is implemented for a color image application with low cost, low power, high performance, and high quality. Using MATLAB, we can put the methods into practice and compare them. In MATLAB, the input image is adjusted before being converted to a CFA image and then a text file. The Verilog code written for interpolation subsequently saved in another text file, implement the three strategies. MATLAB then reads this new text file and creates the image with the improved edge characteristics for image applications.

The RMSE and the CPSNR values are computed from color interpolated image for defining the quality of image. Finally, the Peak signal- to-noise-ratio (PSNR) is increased from 31.1436 to 37.7731 which is increased by 6.6295. This is because, as the value of PSNR increases the quality of the image is also increased. The overall delay reduction is 4.000ns. The overall area reduction is 44% which includes slices, LUTs and the gate count.

6. REFERENCES

[1] R. Lukac, K. N. Plataniotis, and D. H.atzinakos, "Color image zooming on the Bayer pattern," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 15, no. 11, pp. 1475-1492, Nov. 2005

[2] S. C. Hsia, M. H. Chen, and P. S. Tsai, "VLSI implementation of low-power high-quality color interpolation processor for CCD camera," IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 14, no. 4, pp. 361-369, Apr. 2006.

[3] K. Jensen and D. Anastassiou, "Subpixel edge localization and the interpolation of still images," IEEE Trans. Image Process., vol. 4, no. 3, pp. 285–295, Mar. 1995.

[4] Y. H. Shiau, P. Y. Chen, and C. W. Chang, "An area-efficient color demosaicking scheme for VLSI architecture," International Journal of Innovative Computing, Information and Control, Vol.7, No.4, pp.1739-1752, Apr. 2011.

[5] S. C. Hsia, M. H. Chen, and P. S. Tsai, "VLSI implementation of low-power high-quality color interpolation processor for CCD camera," IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 14, no. 4, pp. 361-369, Apr. 2006

[6] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: color filter array interpolation," IEEE Signal Processing Magazine, vol. 22, no. 1, pp. 44–54, 2005.

[7] B. E. Bayer, "Color Imaging Array," U.S. patent 3 971 065, Jul. 1976.

[8] H. A. Chang, and H. H. Chen, "Stochastic color interpolation for digital cameras," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 17, no. 8, pp. 964-973, Aug. 2007.

[9] D. Doswald, J. Hafliger, P. Blessing, N. Felber, P. Niederer, and W. Fichtner, "A 30-frames/s megapixel real-time CMOS image processor," IEEE Journal of Solid-State Circuits, Vol. 35, no. 11, pp. 1732-1743, Nov. 2000.

[10] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10000 fps CMOS sensor with massively parallel image processing," IEEE Journal of Solid-State Circuits, Vol. 43, no. 3, pp. 706-717, Mar. 2008.