Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN IMPROVED VLSI ARCHITECTURE FOR CFA INTERPOLATION USING CARRY SKIP ADDER

CHATLA RAJA RAO[#1] AND DR. SOUMITRA KUMAR MANDAL[*2]

[1#]Dy. Director, Board of Practical Training (Eastern Region), Salt Lake City, Sector-I, Kolkata, India, [*]
[2]Professor, Department of Electrical Engineering, National Institute of Technical Teacher's Training and Research, Kolkata, West Bengal, India.

E-mail: [1]c.rajarao@gmail.com, [1]crrao@bopter.gov.in, [2]skmandal@nittrkol.ac.in, [3]mandal_soumitra@yahoo.com .

ABSTRACT

A wider range of digital devices, including as 4G/5G smart phones, digital cameras, digital notebooks, and consumer electrical items, will be able to function properly thanks to this Application of Color filter array. As a result, a linear deviation compensation approach that boosts correlation between interpolated and neighboring pixels is recommended to be used to this color filter array in an effort to enhance the performance of the reconstructed images with perfection. By prioritizing green in the color interpolation process and using a hardware-sharing methodology, we may enhance the image's resolution on both sides. Therefore, larger space in arithmetic operations and higher gate counts on VLSI architecture will be needed for the hardware sharing approach of red, green, and blue interpolation. In order to cut down on space, time, and energy requirements, the project would include a color demosaicking method that makes use of a carry skip adder as opposed to a standard ripple push adder into all current hardware sharing techniques. In this research, experiments are conducted using VHDL programming language and the synthesize capabilities of the Xilinx FPGA XC6SLX150-2CSG484 at a 200 MHz operating clock frequency to create a color demosaicking approach using a 256x256 pixel images.

Keywords: *CSKA (Carry Skip Adder), CFA (Color Filter Array), Boundary detection, Boundary Mirror Machine, VLSI.*

1. INTRODUCTION

Digital communication-based image processing systems are becoming more important as a result of their ability to support a wide range of electronic devices and consumer goods in the modern digital environment. This digital image processing task will have struggled to demonstrate performance with regards to image resolution, noise aberrations, color mismatches, color fading, and so on. As a result, the image color sensor will be used in cutting-edge machinery to lessen the prevalence of malfunctions across all electronic devices. Bayer color filter arrays are the most common kind of array-based color sensors used in today's electronic devices. Using the color filter array (CFA) technology, the current image's color sensors may be made more cost-effective and memory-efficient, hence reducing its footprint. The color interpolated method may decrease the interpolated missing value and rebuild a full-color image while losing just a third of the information at each set of image boundaries. These color interpolated techniques of color de-mosaicking method improved the de-mosaicking performance, providing high performance and high resolution with edge oriented filtering strength in all sets of image boundaries. Support for stochastic estimates on image interpolation and adjustable resolution on a color filter array are features of the proposed demosaicking technique. Similarly, gradient edge detections based on heterogeneity projection on the color filter array will be used to supplement the edge detection on demosaicking technique. Therefore, the proposed CFA will improve the interpolation algorithm with high quality real time video applications [1] by avoiding artefacts in all the image boundaries with zipper effect, color spots, image blurring, and demoralized to missing interpolations.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

When it comes to digital image processing, image filtering is at the top of the list. This is because it ensures that high-resolution images may be sent and received by all digital devices, including H.254, H.256, HDTV, Smart Phones, LED Projectors, and so on. Due to the matrix-based nature of the image processing, different hardware requirements will apply to images of different resolutions (32x32 pixels, 64x64 pixels, 1024x1024, and HEVC-based 4096x2048 pixels, respectively), with memory and arithmetic operations also playing a role. This study proposes developing a hardware-oriented de-mosaicking interpolated technique using a color filter array. Because of the increased need for basic operations like shifting, addition, and subtraction, the VLSI implementation of this concept for color de-mosaicking will need additional chip space. Existing methods will make an attempt in pipelined design, adaptive edge improvement, and an anisotropic weighting model to deal with this increased area complexity, but their findings will not be relevant to efficient performance. Therefore, the suggested technique will focus on area reduction by using arithmetic operations in a color de-mosaicking algorithm with fewer logic gates, garbage signals, memory logic components, and power consumptions [2]. In this specific situation, the suggested color de-mosaicking architecture would make use of three distinct hardware sharing machines for performing arithmetic operations on input colors. As a result, the carry skip adder, a high-speed and efficient adds technique, will be used in place of more traditional full adders in this proposed work.

This study proposes a digital image filtering technique for color de-mosaicking utilizing a carry skip adder, which is designed in VHDL languages and synthesized in FPGA implementations for a 256 x 256-pixel image. The suggested work aims to minimize the area, latency, and power consumptions [3, 4] caused by the FPGA's logic gates.

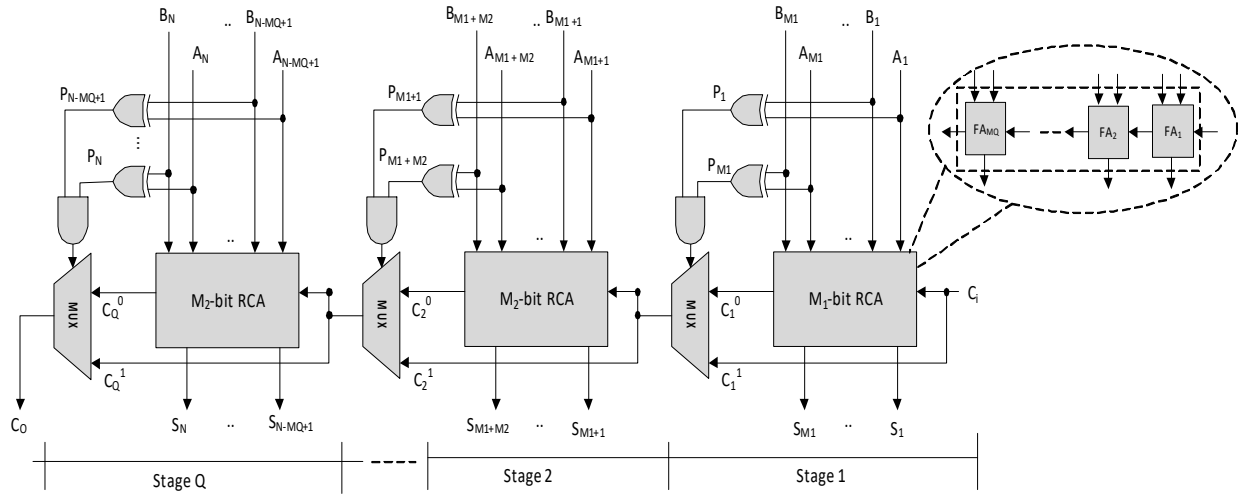

Figure 1 : Architecture of Conventional Structure of the Carry Skip Adder

This study's Section II will demonstrate how a carry skip adder operates with various bit sizes. While Section III will provide a concise summary of a suggested color de-mosaicking technique that makes use of such additions. High-quality image results will be shows once Section IV integrates a color de-mosaicking technique into an FPGA implementation. In Section V, we'll show how we plan to improve upon and wrap up this effort.

2. MULTI-BIT OPERATIONS OF CARRY SKIP ADDER

An arithmetic logic unit is the most important component of a digital image processing programmer. Because mathematical operations like as addition, subtraction, and multiplication are performed on images' pixels, higher resolutions are necessary. Power and energy consumption for arithmetic operations will increase in digital image processing; this is because the number of additions

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

will serve as a multiplication and the number of subtraction operations will serve as a division. There are many different types of conventional adders that can optimize area and power usage in arithmetic operations, a number of carry adders, including the "1. Ripple carry adder, 2. Carry choose adder, 3. Carry increment adder, 4. Manchester carry chain adder, 5. Conditional sum, and 6. Parallel prefix adder". However, the carry skip adder also performs more quickly while performing arithmetic operations. Figure 1 shows the typical carry skip adder structure, which is based on RCA blocks that contain a series of complete adders and has an N-bit size. However, the operation speed and delay of this architecture based on multiplexers will be improved, resulting in a smaller number of critical path delays and a higher maximum operating speed. The multiplexer's output choice is determined by the input and carry logic, as well as any additional XOR & AND gate structure operations. This RCA block has a series of full-wave conventional cascaded amplifiers, adders, having inputs A and B that are multi-bit in size (from A1 to AN and BN, respectively) and producing S1 to SN as outputs. In this procedure, after A and B inputs are provided, they are combined in the RCA block, and a selection signal is generated by the EXOR gate operations according to the following equation (1).

$$P_i = (A_i \oplus B_i) \quad (1)$$

where, the P_i represent the propagation signal which generated from Ai $_{XOR}$ B_i and all the $P_{i....n}$ bit will AND_{ed} finally and given a input of multiplexer selection process. This multiplexer will select skip operation which based on carry generation C^0_1 to C^0_Q.

3. PROPOSED COLOR DE-MOSAICKING ALGORITHM WITH CARRY SKIP ADDER

We recommend utilizing a unique linear deviation approach that compensates for and quickly interpolates using green interpolation, as well as a to produce use a boundary detectors and a boundary mirror device to demonstrate an icing process in color. The suggested color de-mosaicking method is shows in a block diagram form in Fig. 2. After the CFA (Color Filter Array Pixel) is processed, the boundary mirror machine and the boundary detector receive it, which together detect the boundary information and separate it into red, blue, and green interpolated pixels (RB' and G', respectively) using the linear deviation compensation method.

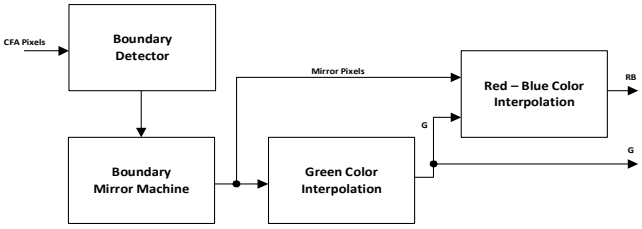

Figure 2 : Architecture of Color Demosaicking interpolation

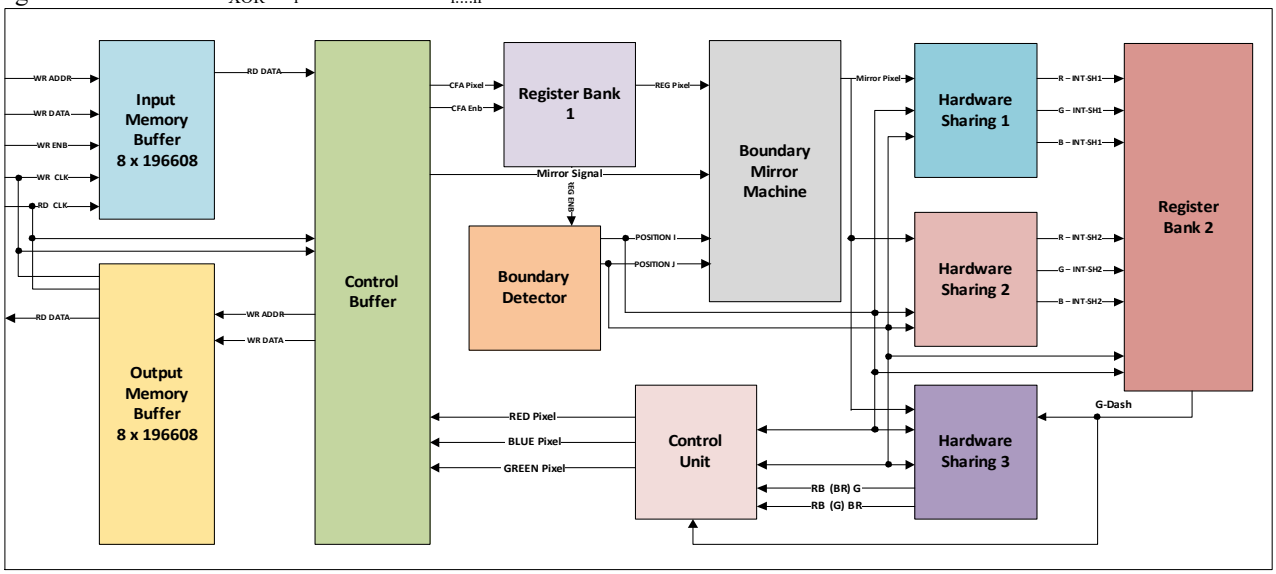

Figure 3: Color De-mosaicking Top module architecture

Journal of Theoretical and Applied Information Technology
15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The proposed color de-mosaicking design in VLSI is seen in Fig. 3. The input images will be processed in MATLAB, and the architecture's input will come from Buffer Memory 19660x8. In the first stage, a 256x256 image is imported into the MATLAB GUI for conversion. Hexadecimal format, these hexadecimal values will be put into memory, and the total number of pixels will be 65536 [5] times 3, as required by the image size of 256 by 256. The input and output buffer memory is 196608 x 8 bit since each pixel has three values—red, green, and blue. The control buffer's next-highest-priority block handles each individual pixel by retrieving its value from the input buffer's memory at each successive address increment and passing it on to other blocks like the register bank 1, the boundary mirror, the boundary detector, control unit, register bank 2, hardware sharing 1, hardware sharing 2, and hardware sharing 3. Consequently, by substituting a shifter for multipliers and divisions, we may drastically lower the hardware cost of this design. This color de-mosaicking algorithm begins with a register bank 1 composed of sixteen shift registers that supply sixteen samples of CFA pixels to the image boundary mirror machine. Simultaneously, a boundary detector determines the coordinates of i and j values using the Bayer color filter array, which has 64 blocks and 24 bits per block (8 bits each for red, green, and blue) (see Fig. 4).

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,8
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,8
3,0	3,1	3,2	3,3	3,4	3,5	3,6	
4,0	4,1	4,2	4,3	4,4	4,5	4,6	
5,0	5,1	5,2	5,3	5,4	5,5	5,6	
6,0	6,1	6,2	6,3	6,4	6,5	6,6	
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

Figure 4 : Pixel Identification with Bayer CFA

Here green color pixels will have been used on the green
interpolation as an equation shows in below (1). More information on the vertical as well as horizontal scaling will be included in the pixel reading, but the order, here the green interpolation avoiding different rows, and its update the results on G' which captures CFA pixels from horizontal directions. A method of green interpolation equation will have given on (2) and pixel based reference will have given on Fig.5, [6].

$$G'_{i,j} = \frac{1}{2}(G'_{i,j-1} + G'_{i,j+1}) + \frac{1}{2}RB_{i,j} - \frac{1}{4}(RB_{i,j-2} + RB_{i,j+2}) \quad (2)$$

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

Figure 5 : Method of Green Interpolation

By using the same green interpolation technique utilized in the previously described method, we could improve the red and blue color interpolation that is offered inside the structure of the more traditional bilinear method. Furthermore, the red and blue interpolation approach will apply a unique linear deviation compensation methodology to rebuild the CFA pixels with a sum of the surrounding green color pixels, so make up for the interpolation of the red and blue. Equation (3) shows the interpolation of red and blue. With four average CFA pixels compensating for the interpolation of green color [7]. As illustrated in Fig.5, the linear deviation of the green interpolation will fall within a range of eight values, and the suggested linear compensation methodology will be used to improve the red and blue interpolations' quality.

$$RB_{i,j}^{(BR)G} = \frac{1}{4}(RB_{i-1,j-1} + RB_{i-1,j+1} + RB_{i+1,j-1} + RB_{i+1,j+1}) + G'_{i,j} - \frac{1}{8}[(G'_{i-1,j-1} + G'_{i-1,j+1} + G'_{i+1,j-1} + G'_{i+1,j+1}) + (G'_{i-1,j} + G_{i,j-1} + G_{i,j+1} + G_{i+1,j})] \quad (3)$$

According to the linear deviation compensation method, the solution to equation (4) will be shows in green, with red and blue interpolation along the vertical axis. Using equations (3) and (4), we may interpolate in both the horizontal and vertical directions, improving image quality while decreasing hardware costs.

Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$RB'^{(G)BR}_{i,j} = \frac{1}{2}(RB_{i-1,j} + RB_{i+1,j}) + G_{i,j} - \frac{1}{8}(G'_{i-1,j} + G'_{i,j-1} + G'_{i,j+1} + G'_{i+1,j}) + (G_{i-1,j-1} + G_{i-1,j+1} + G_{i+1,j-1} + G_{i+1,j+1})] \quad (4)$$

By employing the linear deviation compensation approach, the final equation of CFA color interpolation is presented in equation (4), which will provide a high-quality, low-complexity color de-mosaicking process. Equation (3) provides access to data on the eight values of G' in close proximity to the given value. These values were calculated using equations (4) and (5).

$$RB'^{(G)BR}_{i,j} = \frac{1}{2}(RB_{i,j-1} + RB_{i,j+1}) + \frac{1}{2}G_{i,j} - \frac{1}{4}(G_{i,j-2} + G_{i,j+2}) \quad (5)$$

The linear deviation based color interpolation technique will apply in VLSI architecture with different sub module as per the supporting equation (2), (3), (4), (5). As per this equation a hardware sharing machine 1 will It will appear in Fig. 7 if you obtain the values G'(i-1,j), G'(i+1,j-1), and G'(i+1,j+1). Machine 1 in this equipment sharing will have three carry skip adders. 1 sub tractors and 6 multiplexers. Additionally, it has three pipelined registers due to reducing the critical path delay and also reduced the hardware cost. The input of "$M_{i,j}$ will be differ on each input multiplexer which point out CFA pixel points as per the boundary configuration, in Fig.7 will shows the hardware sharing pixel point and the format will obtain the inputs on $M_{(i-1,j-1)}$, $M_{(i+1,j)}$, $M_{(i+1,j-2)}$, $M_{(i-1,j+2)}$, $M_{(i+1,j)}$, $M_{(i-1,j)}$, $M_{(i+1,j+1)}$, $M_{(i+1,j-1)}$, $M_{(i-1,j-2)}$, $M_{(i+1,j-1)}$, $M_{(i+1,j-3)}$, $M_{(i-1,j-3)}$, $M_{(i-1,j+2)}$, $M_{(i+1,j+3)}$, $M_{(i+1,j+1)}$. The output of Hardware sharing M1 will generate $G'_{(i-1,j)}$ - Green interpolation, $G'_{(i+1,j-1)}$ - Red interpolation, $G'_{(i+1,j+1)}$ - Blue interpolation".

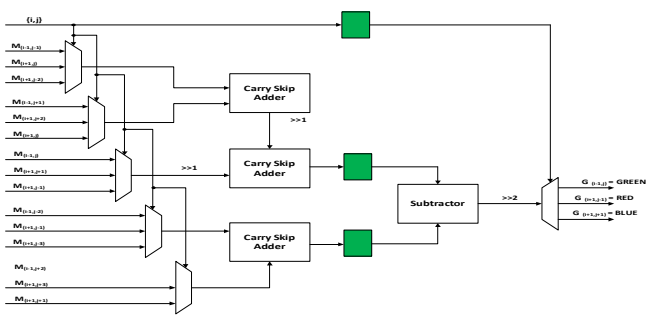

Figure 6 : VLSI Architecture of the Hardware Sharing M1 with using Carry Skip Adder

M(i+3,j-3)	M(i+3,j-2)	M(i+3,j-1)	M(i+3,j)	M(i+3,j+1)	M(i+3,j+2)	M(i+3,j+3)	M(i+4,j+4)
M(i+2,j-3)	M(i+2,j-2)	M(i+2,j-1)	M(i+2,j)	M(i+2,j+1)	M(i+2,j+2)	M(i+2,j+2)	M(i+2,j+4)
M(i+1,j-3)	M(i+1,j-2)	M(i+1,j-1)	M(i+1,j)	M(i+1,j+1)	M(i+1,j+2)	M(i+1,j+3)	M(i+1,j+4)
M(i,j-3)	M(i,j-2)	M(i,j-1)	M(i,j)	M(i,j+1)	M(i,j+2)	M(i,j+3)	M(i,j+4)
M(i-1,j-3)	M(i-1,j-2)	**M(i-1,j-1)**	M(i-1,j)	**M(i-1,j+1)**	**M(i-1,j+2)**	M(i-1,j+3)	M(i-1,j+4)
M(i-2,j-3)	M(i-2,j-2)	M(i-2,j-1)	M(i-2,j)	M(i-2,j+1)	M(i-2,j+2)	M(i-2,j+3)	M(i-2,j+4)
M(i-3,j-3)	M(i-3,j-2)	M(i-3,j-1)	M(i-3,j)	M(i-3,j+1)	M(i-3,j+2)	M(i-3,j+3)	M(i-3,j+4)
M(i-4,j-4)	M(i-4,j-2)	M(i-4,j-1)	M(i-4,j)	M(i-4,j+1)	M(i-4,j+2)	M(i-4,j+3)	M(i-4,j+4)

Figure 7 : Hardware Sharing M1 pixel points

In the Fig.8 architecture will shows the hardware sharing machine 2, Equations (2) and (5) can be used to realize the results of the calculation, which are G'(i,j+1) - Green interpolation and RB'i,j(G)RB. In the architecture it's having three adders, two sub-tractors and one multiplexer. The three pipelining registers will be reducing the critical path delay's. The input of CFA pixel point in the hardware sharing M2 will shows in Fig.9, Five input pixel points, M(i,j-1), M(i,j+1), M(i,j), M(i,j-2), M(i,j+2, are available.

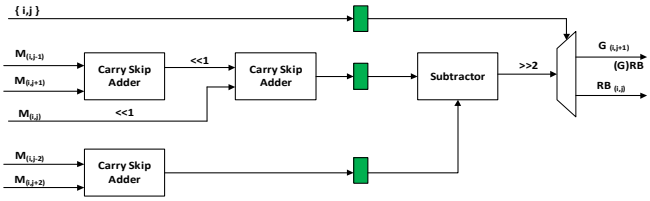

Figure 8 : VLSI Architecture of the Hardware Sharing M2 with Carry Skip Adder

M(i+3,j-3)	M(i+3,j-2)	M(i+3,j-1)	M(i+3,j)	M(i+3,j+1)	M(i+3,j+2)	M(i+3,j+3)	M(i+4,j+4)
M(i+2,j-3)	M(i+2,j-2)	M(i+2,j-1)	M(i+2,j)	M(i+2,j+1)	M(i+2,j+2)	M(i+2,j+2)	M(i+2,j+4)
M(i+1,j-3)	M(i+1,j-2)	M(i+1,j-1)	M(i+1,j)	M(i+1,j+1)	M(i+1,j+2)	M(i+1,j+3)	M(i+1,j+4)
M(i,j-3)	M(i,j-2)	M(i,j-1)	M(i,j)	M(i,j+1)	M(i,j+2)	M(i,j+3)	M(i,j+4)
M(i-1,j-3)	M(i-1,j-2)	M(i-1,j-1)	M(i-1,j)	M(i-1,j+1)	M(i-1,j+2)	M(i-1,j+3)	M(i-1,j+4)
M(i-2,j-3)	M(i-2,j-2)	M(i-2,j-1)	M(i-2,j)	M(i-2,j+1)	M(i-2,j+2)	M(i-2,j+3)	M(i-2,j+4)
M(i-3,j-3)	M(i-3,j-2)	M(i-3,j-1)	M(i-3,j)	M(i-3,j+1)	M(i-3,j+2)	M(i-3,j+3)	M(i-3,j+4)
M(i-4,j-4)	M(i-4,j-2)	M(i-4,j-1)	M(i-4,j)	M(i-4,j+1)	M(i-4,j+2)	M(i-4,j+3)	M(i-4,j+4)

Figure 9 : Hardware Sharing M2 pixel points

In the Fig.10 architecture will shows the hardware sharing machine 3, it will calculate $RB'^{(BR)G}_{i,j}$ - Green interpolation, and $RB'^{(G)BR}_{i,j}$ - Blue and Red interpolation, and it can realize by the equation (3) and (4). Its consists of five pipelined registers to avoid critical path and eleven carry skip adder, a sub-tractors and three shifters. This Hardware sharing M3 method will capture green interpolation pixels from the CFA inputs, with the help of boundary mirror machine and G' from register bank 2 [1]. The input of CFA pixel point in the hardware sharing M3 will shows in Fig.11, it's having twenty

six pixel point such as "$M_{(i-1,j+2)}$, $M_{(i-1,j+3)}$, $M_{(i-1,j+4)}$, $M_{(i-1,j+3)}$, $M_{(i+1,j+2)}$, $M_{(i+1,j+3)}$, $M_{(i+1,j+4)}$, $M_{(i+1,j+3)}$, $G'_{(i,j)}$, $M_{(i,j+3)}$, $M_{(i-1,j+3)}$, $M_{(i-1,j+2)}$, $M_{(i,j+2)}$, $M_{(i-1,j+4)}$, $M_{(i,j+4)}$, $M_{(i+1,j+2)}$, $M_{(i+1,j+3)}$, $M_{(i+1,j+4)}$, $G'_{(i-1,j-1)}$, $G'_{(i-1,j)}$, $G'_{(i-1,j+1)}$, $G'_{(i,j-1)}$, $G'_{(i+1,j-1)}$, $G'_{(i,j+1)}$, $G'_{(i+1,j+1)}$, $G'_{(i+1,j)}$,".

M(i+3,j-3)	M(i+3,j-2)	M(i+3,j-1)	M(i+3,j)	M(i+3,j+1)	M(i+3,j+2)	M(i+3,j+3)	M(i+4,j+4)
M(i+2,j-3)	M(i+2,j-2)	M(i+2,j-1)	M(i+2,j)	M(i+2,j+1)	M(i+2,j+2)	M(i+2,j+3)	M(i+2,j+4)
M(i+1,j-3)	M(i+1,j-2)	M(i+1,j-1)	M(i+1,j)	M(i+1,j+1)	M(i+1,j+2)	M(i+1,j+3)	M(i+1,j+4)
M(i,j-3)	M(i,j-2)	M(i,j-1)	M(i,j)	M(i,j+1)	M(i,j+2)	M(i,j+3)	M(i,j+4)
M(i-1,j-3)	M(i-1,j-2)	M(i-1,j-1)	M(i-1,j)	M(i-1,j+1)	M(i-1,j+2)	M(i-1,j+3)	M(i-1,j+4)
M(i-2,j-3)	M(i-2,j-2)	M(i-2,j-1)	M(i-2,j)	M(i-2,j+1)	M(i-2,j+2)	M(i-2,j+3)	M(i-2,j+4)
M(i-3,j-3)	M(i-3,j-2)	M(i-3,j-1)	M(i-3,j)	M(i-3,j+1)	M(i-3,j+2)	M(i-3,j+3)	M(i-3,j+4)
M(i-4,j-4)	M(i-4,j-2)	M(i-4,j-1)	M(i-4,j)	M(i-4,j+1)	M(i-4,j+2)	M(i-4,j+3)	M(i-4,j+4)

Figure 10 : Hardware Sharing M3 pixel points

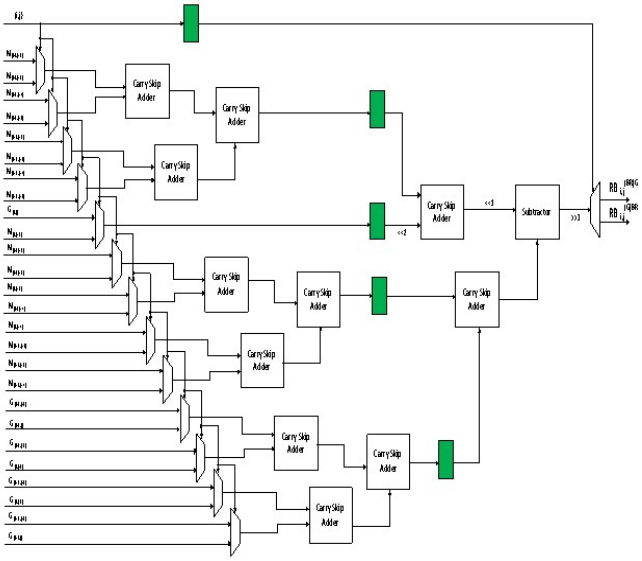

Figure 11 : VLSI Architecture of the Hardware Sharing M3 with Carry Skip Adder

In the Fig.12 architecture will shows CFA Pixel register bank 2, it will calculate G' value from the input of "G'(i-1,j), RB'(G)RB form Hardware sharing M2, G'(i-1,j), G'(i+1,j-1), G'(i+1,j+1)" from Hardware sharing M1. This Register bank will pipeline the interpolation value and give it to G' output as per the boundary detector position changes, those positioning pixel point values are given in Fig.13 [8], [11], [12]. In this register bank will contain six registers and it's having three level upper, middle and lower, the upper part will consider green interpolation, middle and lower part will consider red and blue interpolation from hardware sharing machine M1 and M2. Finally, the control unit and control buffer block, will arranging all the CFA pixel point positions (i,j) and stored into the output block memory buffer 8x196608 [9]. Once the output memory block filled out, the output image data will read out and converted into hex format. Finally, the MATLAB GUI will have shown the output of input image and output image with PSNR comparisons, thus, one of 256 x 256 Akiyo image result analysis will shows Fig.14. Its contain two image left image is input it will take PSNR 44.0693, and right side image is output it will take PSNR 44.1796 [10].

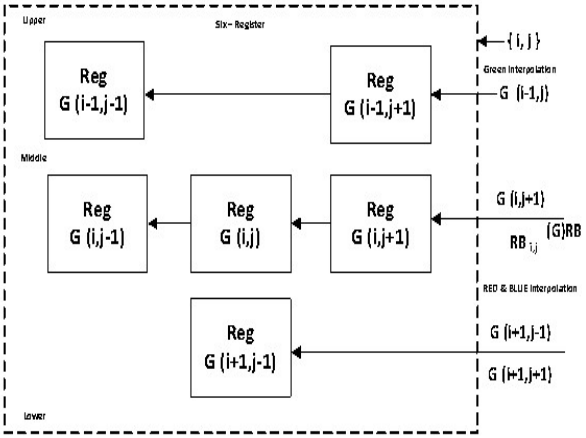

Figure 12 : VLSI Architecture of the register bank 2

M(i+3,j-3)	M(i+3,j-2)	M(i+3,j-1)	M(i+3,j)	M(i+3,j+1)	M(i+3,j+2)	M(i+3,j+3)	M(i+4,j+4)
M(i+2,j-3)	M(i+2,j-2)	M(i+2,j-1)	M(i+2,j)	M(i+2,j+1)	M(i+2,j+2)	M(i+2,j+3)	M(i+2,j+4)
M(i+1,j-3)	M(i+1,j-2)	G(i+1,j-1)	G(i+1,j)	G(i+1,j+1)	M(i+1,j+2)	M(i+1,j+3)	M(i+1,j+4)
M(i,j-3)	M(i,j-2)	G(i,j-1)	G(i,j)	G(i,j+1)	M(i,j+2)	M(i,j+3)	M(i,j+4)
M(i-1,j-3)	M(i-1,j-2)	G(i-1,j-1)	G(i-1,j)	G(i-1,j+1)	M(i-1,j+2)	M(i-1,j+3)	M(i-1,j+4)
M(i-2,j-3)	M(i-2,j-2)	M(i-2,j-1)	M(i-2,j)	M(i-2,j+1)	M(i-2,j+2)	M(i-2,j+3)	M(i-2,j+4)
M(i-3,j-3)	M(i-3,j-2)	M(i-3,j-1)	M(i-3,j)	M(i-3,j+1)	M(i-3,j+2)	M(i-3,j+3)	M(i-3,j+4)
M(i-4,j-4)	M(i-4,j-2)	M(i-4,j-1)	M(i-4,j)	M(i-4,j+1)	M(i-4,j+2)	M(i-4,j+3)	M(i-4,j+4)

Figure 13: Register Bank Pixel point values

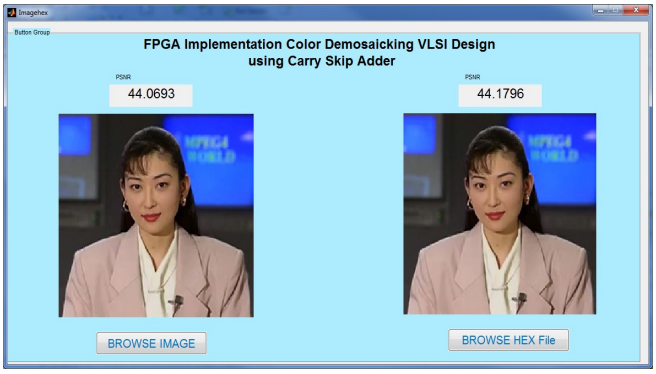

Figure 14 : 256X256 Image result analysis on MATLAB with PSNR

3. FPGA RESULTS AND IMPLEMENTATIONS

In this paper, we offer a high-quality color de-mosaicking technique that makes use of a carry skip adder to cut down on hardware expenses in digital image processing software. This construction of a color filter array is supported in the simulation process by the MATLAB distribution, which allows for the conversion of images to and from hexadecimal notation as well as the inspection of PSNR and SSIM values [13], [14]. Initial steps included implementing a high-quality color de-mosaicking algorithm using an 8x8 image size and two adders, both a carry skip adder and a typical RCA (ripple carry adder). This 8x8 image size will yield better results using the proposed method of carry skip adder implementation, so the numbers in Table.1 will be updated to reflect the latest values for things like slice register count (497 vs. 579), look-up table count (343 vs. 516), and delay performance (3.617 ns vs. 3.617 ns) [15] for both the conventional and proposed implementations of the 8x8 image.

Table 1: Comparison Table Of High Quality Color DE Mosaicking VLSI Design - Image Size 8x8

	High Quality Color DE mosaicking VLSI Design	
	CFA - Conventional Adder	CFA- Carry Skip Adder
Image Size	8x8	8x8
Slice Register	579	497
LUT	516	343
IOB	52	52
Delay (ns)	4.027	3.617
Power (mW)	361	353

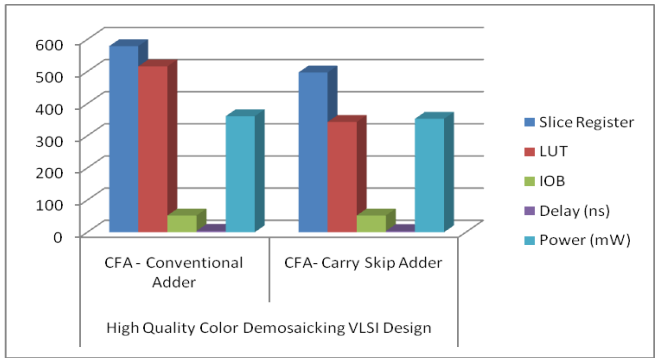

Figure 15 : Comparisons Analysis Of High Quality Color DE Mosaicking VLSI Design - Image Size 8 X 8

Following that, the work that is presented for this study will show a high grade color demosaicking approach that makes use of a 256 × 256 picture size using a multi bit carry skip adder method. According to this, the design was created using the VHDL programming language; the simulation was run in Modelsim 6.5b; and the results of the simulation were synthesized on Xilinx 14.2 by using an FPGA with the model number XC6LX150-2CSG484. In conclusion, the work that was presented demonstrated a satisfactory performance in terms of area (slice register, LUT), IOB (Input output block), latency, and power. On Table.2 you'll find an updated version of the comparison table with high-quality 256 x 256 color demosaicking designs. It will take 679, and the LUT will take 758, and the delay will take 15.114 nanoseconds, and the power will take 184 milliwatts. This will demonstrate that the number of slices register will decrease in the CFA - Carry skip adder architecture. The results of the analysis will be shows in Fig.16 for these comparisons. The RTL schematic of the 256 x 256 CFA demosaicking technique will be given in Figure 19, and the simulation results will be given in Figure 20. For synthesizing result analysis of 256 × 256 image sizes using carry skip adder method will provide in Fig.17, and Fig.18 will updated with delay report.

Table 2 : Comparison Table Of High Quality Color De-Mosaicking VLSI Design - Image Size 256 X 256

	High Quality Color De-mosaicking VLSI Design 256 x256	
	CFA - Conventional Adder	CFA - Carry Skip Adder
PSNR	44.0693	44.1796
Slice Register	689	679
LUT	754	758
IOB	82	82
Delay (ns)	33.904	15.114
Power (mW)	198	184

Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

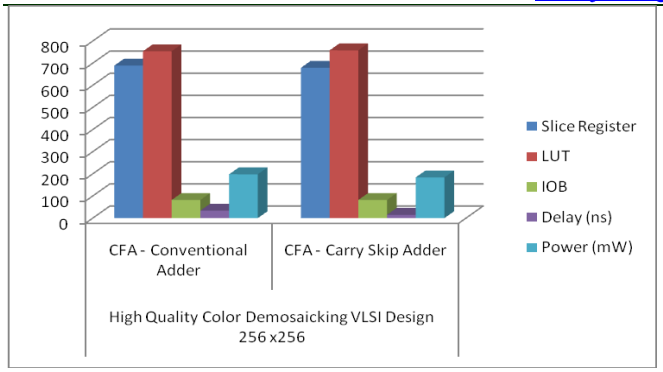

Figure 16 : Comparisons Analysis Of High Quality Color De-Mosaicking VLSI Design - Image Size 256x256

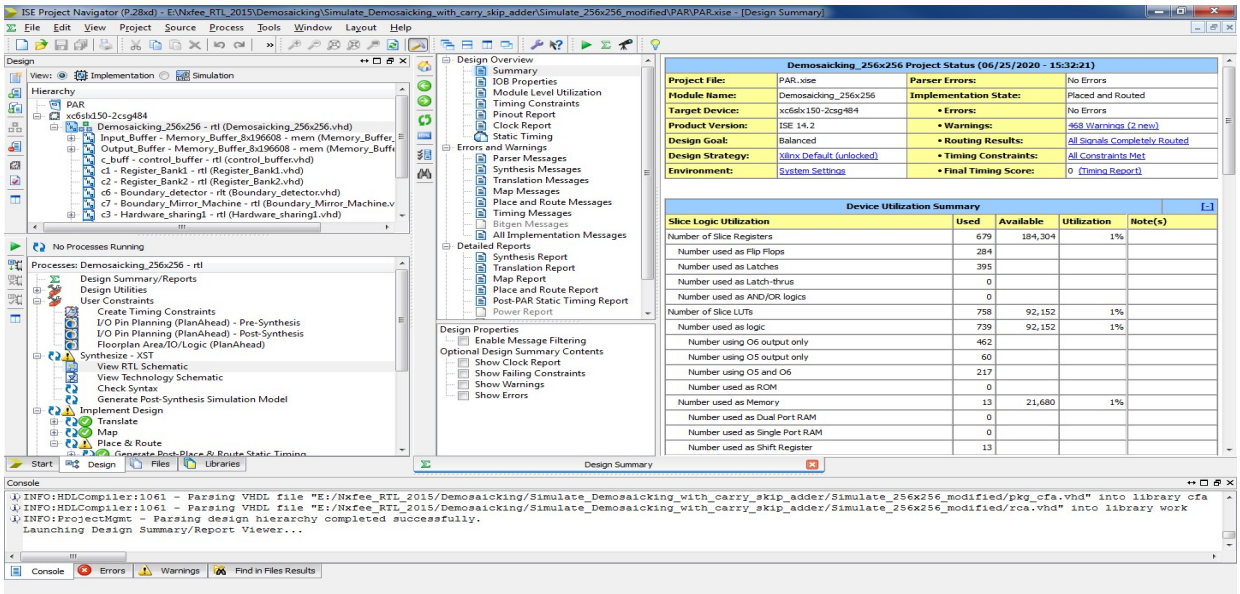

Figure 17 : Synthesize Result Of Proposed Color De-Mosaicking Method With Using Carry Skip Adder

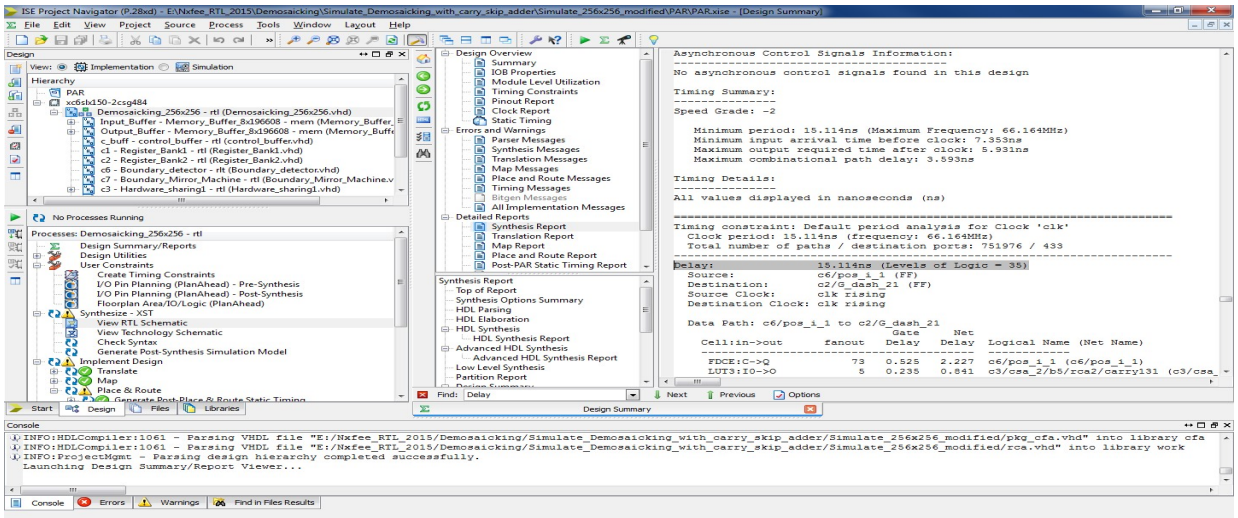

Figure 18 : Delay report of Proposed Color De-mosaicking method with using Carry skip Adder

Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

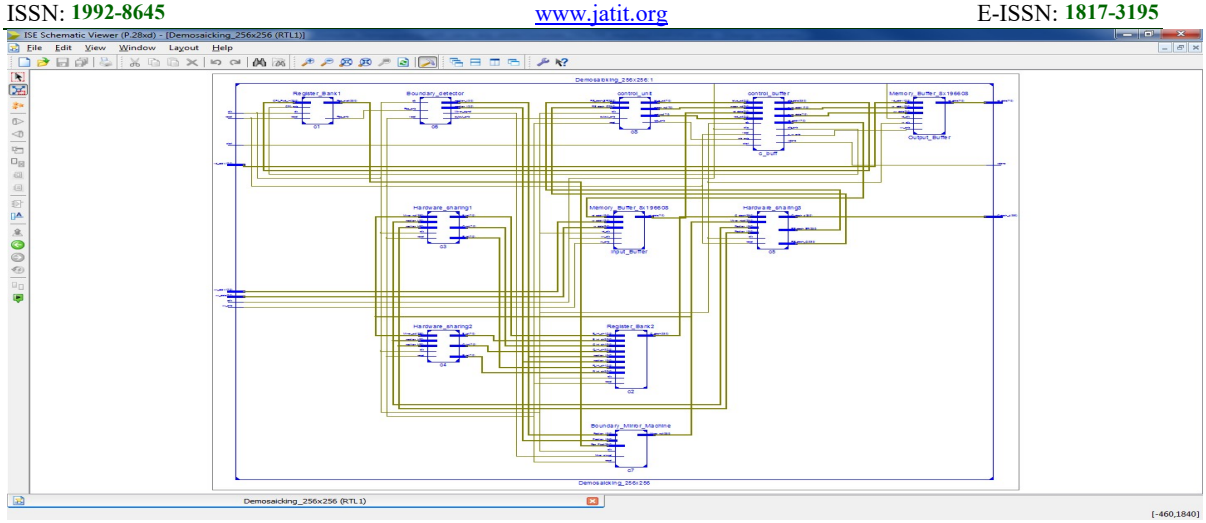

Figure 19 : RTL Schematic of Proposed Color De-mosaicking method with using Carry skip Adder

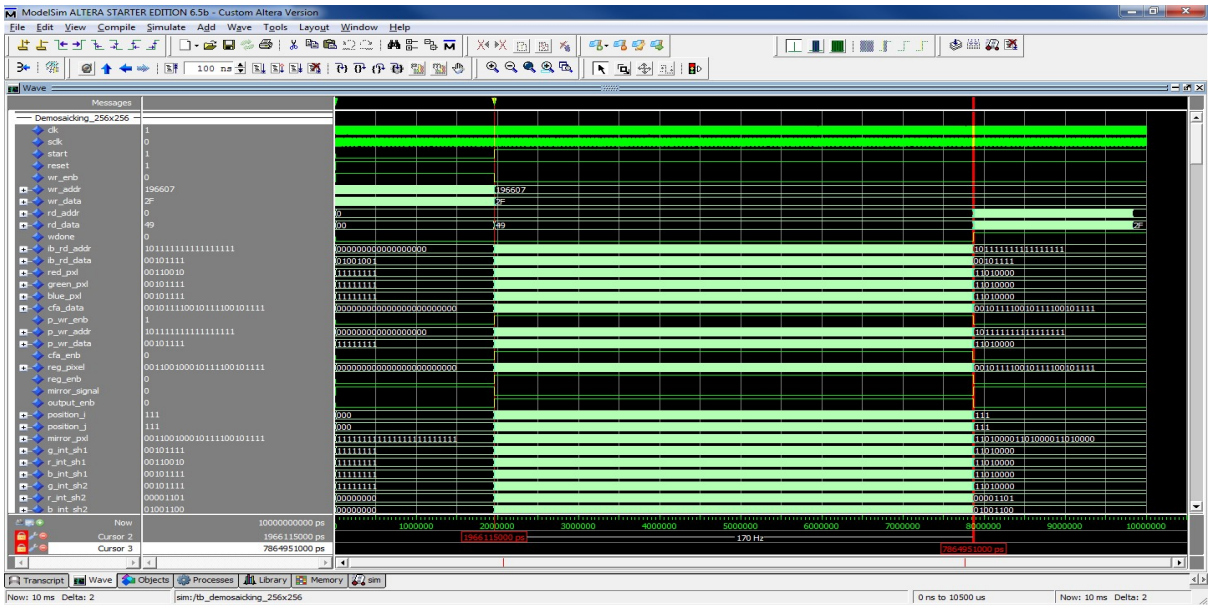

Figure 20 : Simulation result of Proposed Color De-mosaicking method with using Carry Skip Adder

4. CONCLUSION

The purpose of this study is to accomplish the goal of reducing the hardware logic size as well as the power consumptions of the color de-mosaicking approach while maintaining a low cost and good performance in image processing applications. The previous approach of a color filter array will occupy more logic size in VLSI System design due to large number of arithmetic operations. With the proposed color filter array architecture of hardware sharing machines, boundary mirror machines, and boundary detectors with register banks, this approach for color interpolation will reduced a number of arithmetic operations pertaining to the filtering of all sets of image resolutions. Here, the approach that is being proposed would incorporate a linear deviation based color interpolation methodology using a carry skip adder as instead of an traditional ripple carry adder. This work was produced on a Xilinx FPGA with a model number of XC6SLX150-2CSG484 and an operational clock frequency of 200 MHz. As a result, the performance of the area (Slice registers, LUT, and IOB), latency, and power was proven.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES

[1] Shih-Lun Chen, Huan-Rui Chang, IEEE Member, Fully Pipelined Low Cost and High Quality Color De-mosaicking VLSI Design for Real Time Video Applications, 2015, IEEE Transactions on Circuit and Systems-II.

[2] Milad Bahadori, Mehdi Kamal, Ali Afzali-Kusha, Massoud Pedram, IEEE Member, High Speed and Energy Efficient Carry Skip Adder Operating Under a Wide Range of Supply Voltage Levels, 2015 IEEE Transactions on Very Large Scale Integration systems.

[3] K.Chirca et al,. "A Static low power high performance 32-bit carry skip adder," in Proc. Euromicro Symp, Digit, Syst, Design (DSD), Aug/Sep. 2004, pp. 615-619.

[4] M. Alioto and G. Palumbo, "A Simple strategy for optimized design of one-level carry skip adders," IEEE Trans, Circuits Syst. I, Fundam, Theory Appl,. Vol. 50, no.1, pp. 141-148, Jan 2003.

[5] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, Apr. 2004.

[6] S. L. Chen and E. D. Ma, "VLSI Implementation of an Adaptive EdgeEnhanced Color Interpolation Processor for Real-Time Video Applications," IEEE Transaction on Circuits and Systems for Video Technology, 2014. (Accepted).

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

[8] S. C. Hsia, M. H. Chen, and P. S. Tsai, "VLSI implementation of lowpower 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.

[9] C. Hu, L. Cheng, and Y. M. Lu, "Graph-based regularization for color image de-mosaicking," in Proc. IEEE Int. Conf. Image Processing (ICIP), Oct. 2012, pp. 2769–2772.

[10] X. Chen, G. Jeon, and J. Jeong, "Voting-based directional interpolation method and its application to still color image demosaicking," IEEE Transaction on Circuits and Systems for Video Technology, vol. 24, no. 2, pp. 255-262, Feb. 2014.

[11] K. L. Chung, W. J. Yang, W. M. Yan, and C. C. Wang, "Demosaicing of Color Filter Array Captured Images Using Gradient Edge Detection Masks and Adaptive Heterogeneity-Projection," IEEE Trans. Image Processing, vol. 17, no. 12, pp. 2356–2367, Dec. 2008.

[12] D. Harris and I. Sutherland, "Logical effort of carry propagate adders," Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, vol. 1, pp. 873-878, November, 2003.

[13] V. Kantabutra, "Accelerated two-level carry-skip addersa type of very fast adders," IEEE Transactions on Computers, vol. 42, no. 11, pp. 1389-1393, November 1993.

[14] P. Chan, M. Schlag, C. Thomborson, and V. Oklobdzija, "Delay optimization of carry-skip adders and block carry-lookahead adders using multidimensional dynamic programming," IEEE Transactions on Computers, vol. 41, no. 8, pp. 920-930, August 1992.

[15] S. Turrini, "Optimum group distribution in carry-skip adders," in Proceedings of the 9th IEEE Symposium on Computer Arithmetic, pp. 96–103, September, 1989.

[16] M. Alioto and G. Palumbo, "A simple strategy for optimized design of one-level carry-skip adders,'' IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 50, no. 1, pp. 141-148, January 2003.

[17] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high-performance microprocessor VLSI adders,'' Proceedings of the 16th IEEE Symposium on Computer Arithmetic, pp. 272-279, June 2003.

[18] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-timepower tradeoffs in parallel adders,'' IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 53, no. 10, pp. 689-702, October 1996.

[19] I Koren, Computer Arithmetic Algorithms, 2nd edition A. K. Peters, Ltd., Natick, MA, 2002.

[20] K. Uming , T. Balsara, and W. Lee, "Low-power design techniques for high-performance CMOS adders,'' IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 3, no. 2, pp. 327-333, June 1995.