# HIGH PERFORMANCE HIGH QUALITY IMAGE DEMOSAICING HARDWARE DESIGNS

by

HASAN AZGIN

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

Spring 2015

HIGH PERFORMANCE HIGH QUALITY IMAGE DEMOSAICING HARDWARE DESIGNS

APPROVED BY

Assoc. Prof. Dr. İlker HAMZAOĞLU ..........[illegible]..........

(Thesis Supervisor)

Assoc. Prof. Dr. Gözde ÜNAL ..........[illegible]..........

Assist. Prof. Dr. Yakup GENÇ ..........[illegible]..........

DATE OF APPROVAL: 03.09.2015

HIGH PERFORMANCE HIGH QUALITY IMAGE DEMOSAICING HARDWARE DESIGNS

Hasan AZGIN

EE, MS Thesis, 2015

Thesis Supervisor: Assoc. Prof. Dr. İlker HAMZAOĞLU

Keywords: Image demosaicing, alternating projections, enhanced effective color interpolation, hardware implementation, FPGA.

## Abstract

Since capturing three color channels (red, green, and blue) per pixel increases the cost of digital cameras, most digital cameras capture only one color channel per pixel using a single image sensor. The images pass through a color filter array before being captured by the image sensor. Demosaicing is the process of reconstructing the missing color channels of the pixels in the color filtered image using their available neighboring pixels. There are many image demosaicing algorithms with varying reconstructed image quality and computational complexity. In this thesis, high performance hardware architectures are designed for two high quality image demosaicing algorithms with high computational complexity. The proposed hardware architectures are implemented on an FPGA.

A high performance Alternating Projections (AP) image demosaicing hardware is proposed. This is the first AP image demosaicing hardware in the literature. A high performance Enhanced Effective Color Interpolation (EECI) image demosaicing hardware is proposed. This is the first EECI image demosaicing hardware in the literature. The proposed hardware architectures are implemented using Verilog HDL. The Verilog RTL codes are mapped to a Xilinx Virtex 6 FPGA. The proposed FPGA implementations are verified with post place & route simulations. They can process 31 and 94 full HD (1920x1080) images per second, respectively.

YÜKSEK PERFORMANSLI YÜKSEK KALİTELİ RESİM DEMOZAİKLEME DONANIM TASARIMLARI

Hasan AZGIN

EE, Yüksek Lisans Tezi, 2015

Tez Danışmanı: Doç. Dr. İlker HAMZAOĞLU

Anahtar Kelimeler: Resim demozaikleme, değişken projeksiyon, pekiştirilmiş etkin renk interpolasyonu, donanım tasarımı, FPGA.

## Özet

Piksel başına üç renk kanalını (kırmızı, yeşil, ve mavi) yakalamak dijital fotoğraf makinesinin maliyetini arttırdığı için, çoğu dijital fotoğraf makinesi sadece bir resim sensörü kullanarak piksel başına sadece bir renk kanalı yakalar. Resimler sensörde yakalanmadan önce renk filtresi düzeneğinden geçer. Demozaikleme, renk filtresinden geçmiş resmin piksellerindeki eksik renk kanallarının mevcut komşu piksellerden elde edilmesi işlemidir. Farklı yeniden oluşturulan resim kalitesi ve işlemsel karmaşıklığa sahip birçok demozaikleme algoritması vardır. Bu tezde, yüksek işlemsel karmaşıklığa sahip iki yüksek kaliteli resim demozaikleme algoritması için yüksek performanslı donanım mimarileri tasarlandı. Önerilen donanımlar FPGA üzerinde gerçeklendi.

Yüksek performanslı Değişken Projeksiyon resim demozaikleme algoritması donanımı önerildi. Bu, literatürdeki ilk Değişken Projeksiyon resim demozaikleme donanımıdır. Zenginleştirilmiş Efektif Renk İnterpolasyonu resim demoazikleme algoritması önerildi. Bu, literatürdeki ilk Zenginleştirilmiş Efektif Renk İnterpolasyonu resim demozaikleme donanımıdır. Önerilen donanım mimarileri Verilog donanım tanımlama dili kullanılarak gerçeklendi. Verilog kodları Xilinx Virtex 6 FPGA'sına yerleştirildi. FPGA gerçeklemeleri yerleştirme ve yönlendirme sonrası simülasyonlar ile doğrulandı. Bu FPGA gerçeklemeleri, sırasıyla saniyede 31 ve 94 tam yüksek çözünürlüklü (1920x1080) resim işleyebilmektedir.

# Acknowledgements

I would like to thank my advisor, Dr. İlker Hamzaoğlu for his guidance, support and teachings during my masters' education. His suggestions, corrections and reviews were immensely helpful throughout my thesis.

I am grateful to my thesis committee members, Dr. Gözde Ünal and Dr. Yakup Genç, for their precious feedback and time.

I would like to thank members of System-on-Chip Design and Test Lab, Ercan Kalalı and Serkan Yalıman for their support. I also would like to thank Ömer Seçmeler, Onur Girgin, Hamza Kazanci, Gökay Çoruhlu and Ali Gemci for their friendship throughout my studies.

I am also thankful to my family for their support during my educational path.

Lastly, I would like to thank Sabancı University and TÜBİTAK for supporting me during my masters studies.

# TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| AP | : | Alternating Projections |
| BRAM | : | Block RAM |
| CFA | : | Color Filter Array |
| ECI | : | Effective Color Interpolation |
| EECI | : | Enhanced Effective Color Interpolation |
| HD | : | High Definition |
| HVS | : | Human Visual System |
| IPS | : | Images per Second |
| LUT | : | Lookup Table |
| MSE | : | Mean Squared Error |
| POCS | : | Projection onto Convex Sets |
| PSNR | : | Peak Signal-to-Noise Ratio |
| RGB | : | Red, Green, Blue |

# Chapter 1

# INTRODUCTION

Since capturing three color channels (red(R), green(G), and blue(B)) per pixel increases the cost of digital cameras, most digital cameras capture only one color channel per pixel using a single image sensor. The images pass through a color filter array (CFA) before being captured by the image sensor. Several CFA patterns are shown in Figure 1.1. Bayer pattern, shown in Figure 1.1 (a), is the most commonly used CFA pattern in digital cameras [1]. Bayer pattern takes the human vision systems relatively higher sensitivity to green into account by sampling green channel at twice the rate of red and blue channels [1]. CFA interpolation, also known as demosaicing (or demosaicking), is the process of reconstructing the missing color channels of the pixels in the color filtered image using their available neighboring pixels. Demosaicing process is shown in Figure 1.2. There are many image demosaicing algorithms with varying reconstructed image quality and computational complexity.

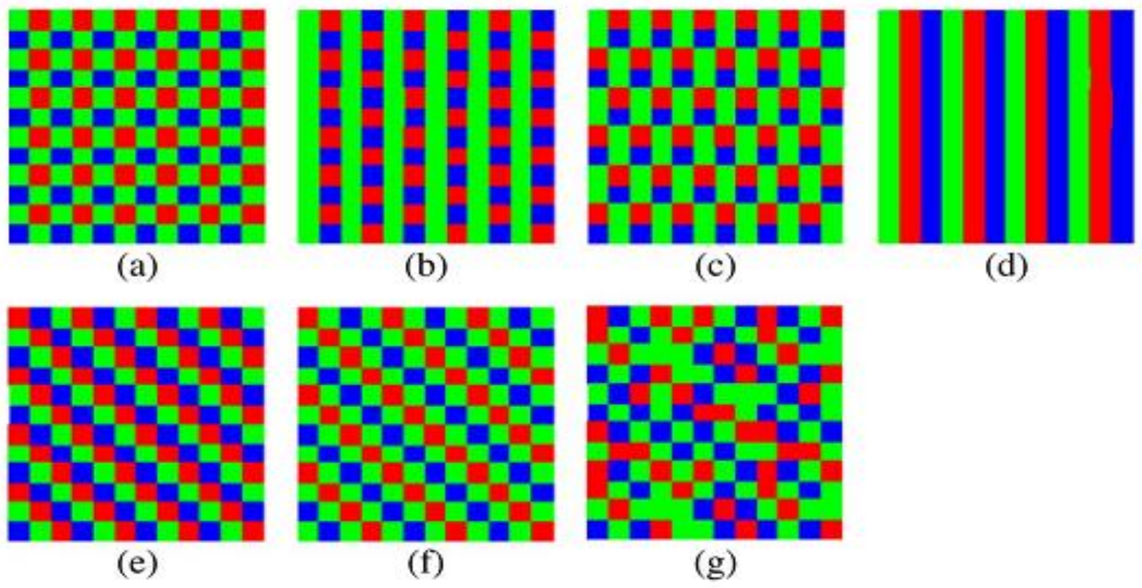
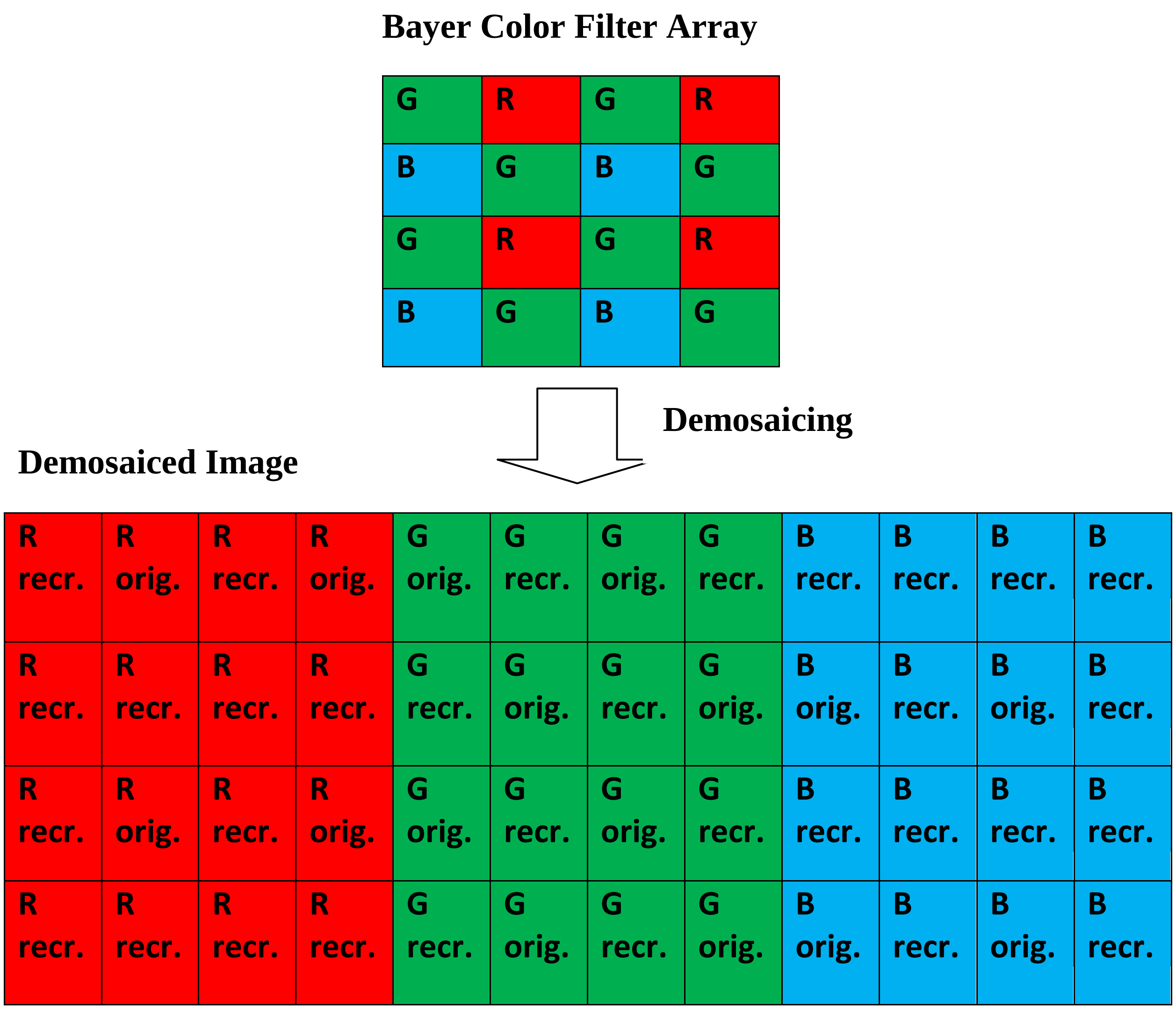


Figure 1.1 RGB color filter arrays (a) Bayer [1] (b)Yamanaka [2] (c) Lukac and Plataniotis [3] (d) Vertical-stripe [5] (e) Diagonal-stripe [5] (f) Modified Bayer [5] (g) HVS-based [6]



Bayer Color Filter Array
G R G R
B G B G
G R G R
B G B G
Demosaicing
Demosaiced Image
R recr. R orig. R recr. R orig. G orig. G recr. G orig. G recr. B recr. B recr. B recr. B recr.
R recr. R recr. R recr. R recr. G recr. G orig. G recr. G orig. B orig. B recr. B orig. B recr.
R recr. R orig. R recr. R orig. G orig. G recr. G orig. G recr. B recr. B recr. B recr. B recr.
R recr. R recr. R recr. R recr. G recr. G orig. G recr. G orig. B orig. B recr. B orig. B recr.

Figure 1.2 Image demosaicing

As shown in Figure 1.3(b), simple image demosaicing algorithms such as bilinear interpolation produce low quality reconstructed images often accompanied with aliasing problems around edges. Therefore, complex image demosaicing algorithms such as Effective Color Interpolation (ECI) [7], Enhanced ECI (EECI) [8] and Alternating Projections (AP) [9] are proposed. These algorithms produce higher quality reconstructed images at the expense of increased computational complexity. They use larger number of neighboring pixels for color interpolation and the correlation between different color channels of the neighboring pixels.

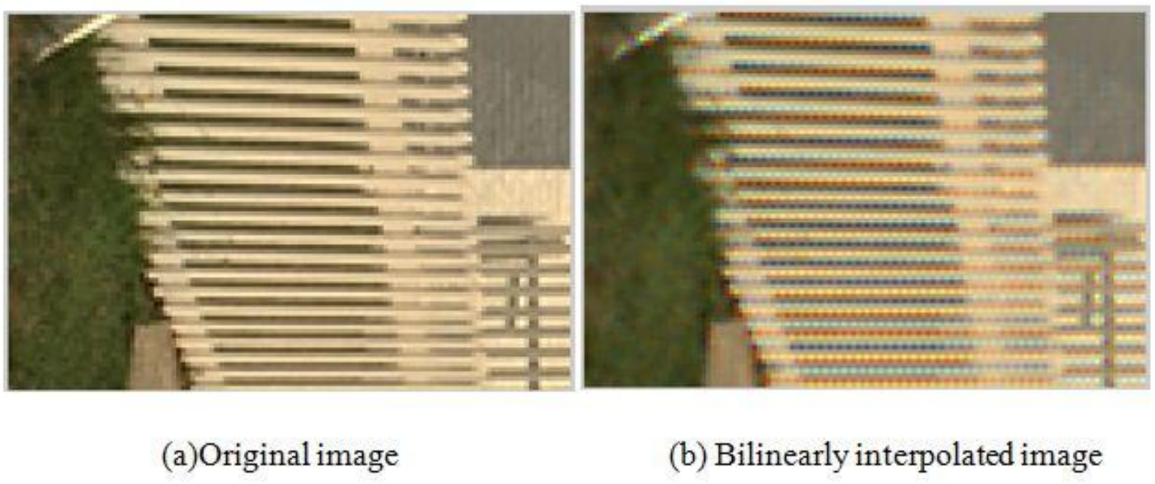


Figure 1.3 Aliasing problem in bilinearly interpolated image

## 1.1. Thesis Contribution

Alternating Projections (AP) and Enhanced Effective Color Interpolation (EECI) image demosaicing algorithms have very high computational complexity. Their software implementations cannot achieve real-time performance even on desktop processors. Therefore, high performance hardware architectures should be designed for real-time implementation of these high quality image demosaicing algorithms.

In this thesis, first, we propose a high performance AP image demosaicing hardware [16]. This is the first AP image demosaicing hardware in the literature. It uses parallelism and pipelining to achieve real-time performance. The proposed hardware architecture is implemented using Verilog HDL. The Verilog RTL code is mapped to a Xilinx Virtex 6 FPGA. The proposed FPGA implementation is verified with post place & route simulations. It can process 166 768x512 images per second or 31 full HD (1920x1080) images per second. The proposed FPGA implementation is 136 times faster than a C++ software implementation of AP algorithm running at 2.4 GHz on an Intel Core i7 processor with 16 GB DRAM.

Then, we propose a high performance EECI image demosaicing hardware. This is the first EECI image demosaicing hardware in the literature. It uses parallelism and pipelining to achieve real-time performance. The proposed hardware architecture is implemented using Verilog HDL. The Verilog RTL code is mapped to a Xilinx Virtex 6 FPGA. The proposed FPGA implementation is verified with post place & route simulations. It can process 498

768x512 images per second or 94 full HD (1920x1080) images per second. The proposed FPGA implementation is 105 times faster than a C++ software implementation of EECI algorithm running at 2.4 GHz on an Intel Core i7 processor with 16 GB DRAM.

In this thesis, image demosaicing algorithms are evaluated as follows. Images in the Kodak Image Suite shown in Figure 1.4 [10], which include all three color channels, are filtered through Bayer pattern. The color filtered images are reconstructed by the image demosaicing algorithm. Each color channel (R, G, B) of each reconstructed image is then compared to the corresponding color channel of the original image using Peak-Signal-to-Noise Ratio (PSNR) metric. Higher PSNR value indicates that the reconstructed image more closely matches the original image. PSNR is calculated using Mean Squared Error (MSE), as shown in Equations 1.1 and 1.2. *N* and *M* denote the image height and width respectively. *R* is the reconstructed image and *O* is the original image. MAX is the maximum value a pixel can take. In this thesis, this value is set to 255.

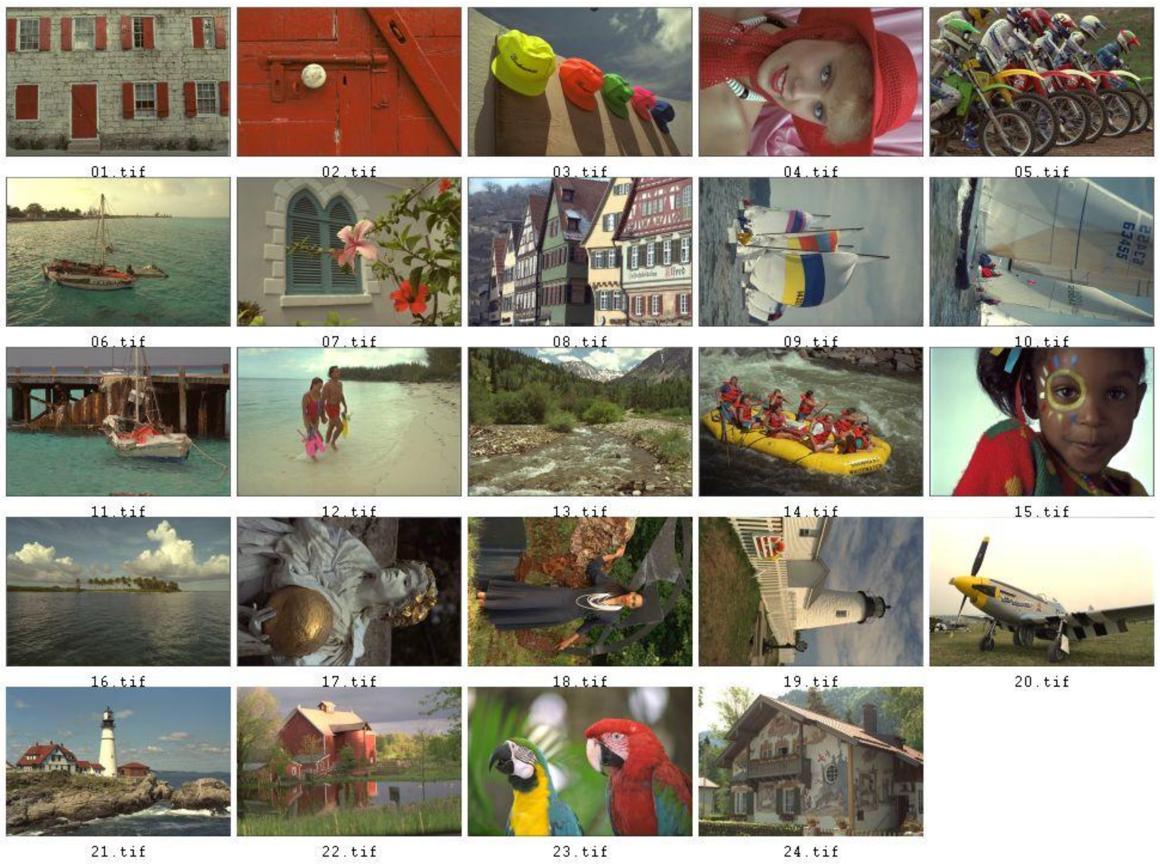


Figure 1.4 Kodak image suite

$$MSE = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(R(i,j) - O(i,j))^2 \tag{1.1}$$

$$PSNR = 10.\,log_{10}\left(\frac{MAX^2}{MSE}\right) = 20.\,log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \tag{1.2}$$

The rest of the thesis is organized as follows. In Chapter 2, the proposed high performance AP image demosaicing hardware is presented, and its implementation results are given. In Chapter 3, the proposed high performance EECI image demosaicing hardware is presented, and its implementation results are given. In Chapter 4, the thesis is concluded.

# Chapter 2

# PROPOSED AP IMAGE DEMOSAICING HARDWARE

## 2.1 Alternating Projections Algorithm

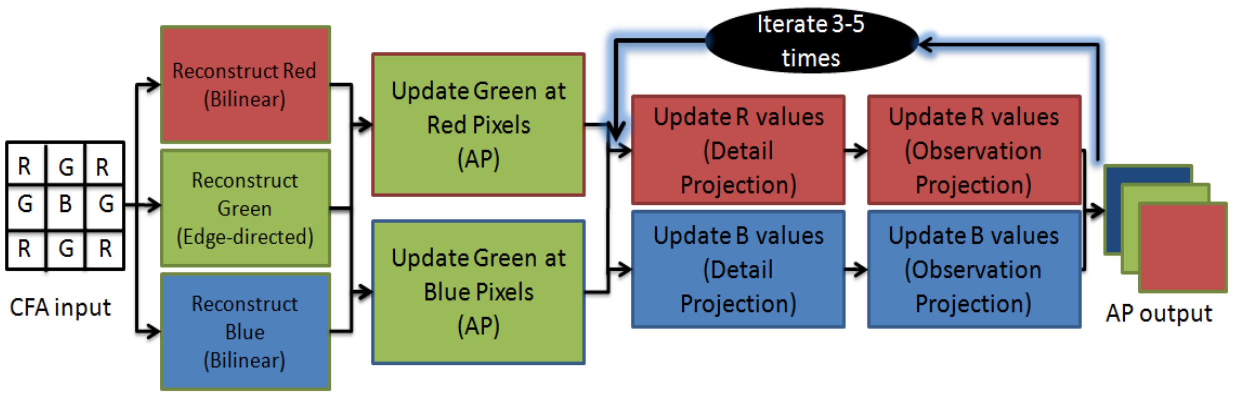
Alternating Projections (AP) image demosaicing algorithm uses the correlation between color (R, G, B) channels of the neighboring pixels for producing high quality reconstructed images [9]. It uses the higher correlation available in the high frequency components when reconstructing the high frequency parts of the missing color channels. Therefore, it performs better than the demosaicing algorithms which do not perform effective interpolation around the edges present in the image that correspond to the high frequency parts of the image.

AP algorithm defines two constraint sets on the reconstructed image. Applying these constraint sets to the reconstructed image is known as Projection onto Convex Sets (POCS).

*Observation Projection*: Reconstructed image must be consistent with the CFA input image

*Detail Projection*: High frequency components of red and blue channels must be similar to that of green channel

Detail projection is obtained by using combinations of low-pass (h0 = [1 2 1] / 4) and high-pass (h1 = [1 -2 1] / 4) filters on red, blue and green channels. Resulting subbands are named as *LL* where both rows and columns are low-pass filtered, *LH* where rows are low-pass filtered and columns are high-pass filtered, *HL* where rows are high-pass filtered and columns are low-pass filtered, and *HH* where both rows and colums are high-pass filtered. The missing color channels are reconstructed by inverse filtering the appropriate subbands with the filters g0 = [-1 2 6 2 -1] / 8 and g1 = [1 2 -6 2 1] / 8.



R G R
G B G
R G R
CFA input
Reconstruct Red (Bilinear)
Reconstruct Green (Edge-directed)
Reconstruct Blue (Bilinear)
Update Green at Red Pixels (AP)
Update Green at Blue Pixels (AP)
Iterate 3-5 times
Update R values (Detail Projection)
Update R values (Observation Projection)
Update B values (Detail Projection)
Update B values (Observation Projection)
AP output

Figure 2.1 Alternating Projections algorithm

As shown in Figure 2.1, AP algorithm works as follows.

1) Green channel is reconstructed by edge-directed demosaicing. Red and blue channels are reconstructed by bilinear demosaicing.

2) Down-sampled red and down-sampled blue channels are formed.

3) Green channel is then reconstructed by inverse filtering LL subband of green and LH, HL and HH subbands of down-sampled red or down-sampled blue respectively. The reconstructed green values are replaced with the original green values available in the CFA input image.

4) After green channel is updated using detail projection and observation projection, similar process is performed for red and blue channels. Red channel is reconstructed by inverse filtering LL subband of red and LH, HL and HH subbands of green. Blue channel is reconstructed by inverse filtering LL subband of blue and LH, HL and HH subbands of green. The reconstructed red and blue values are replaced with the original ones available in the CFA input image.

5) Step 4 is repeated 3-5 times.

AP is an iterative algorithm, and it usually converges after 5 iterations.

## 2.2 Proposed AP Demosaicing Hardware

Two minor modifications are done to AP algorithm to reduce the complexity of the proposed AP hardware. A threshold value is used in AP algorithm to determine the impact of the correlation between color channels to reconstructed image. Although the threshold value is set to 0.02 in [9], it is stated that it is acceptable to set it to zero for natural images, since they have highly correlated subbands. Therefore, the threshold value is set to zero.

Double precision variables are used for AP algorithm in [9]. Fixed point variables are used in all modules of the proposed AP hardware to reduce its area. The size of the integer part of each fixed point variable is set to the minimum value required to avoid overflow, and the size of the fractional part of all fixed point variables is set to 3 bits. In addition, the results produced by each module of the proposed AP hardware (Bilinear Interpolation, Edge-directed Interpolation, Update Green Channel, POCS for Red Channel and POCS for Blue Channel) are truncated to integer to reduce the amount of required on-chip and off-chip memory.

The PSNR (dB) performances of the original AP algorithm and the slightly modified AP algorithm implemented in the proposed AP hardware for the 24 images in the Kodak Image Suite [10] are shown in Table 2.1. These results show that the modifications have negligible impact on the performance of the AP algorithm.

TABLE 2.1 PSNR (dB) COMPARISON OF IMAGE DEMOSAICING ALGORITHMS

| **Image** | **Bilinear** | **ECI** | **AP (5 ITERATIONS)** | **AP HARDWARE (5 ITERATIONS)** |
|---:|---:|---:|---:|---:|
| 1 | 25.31 | 33.21 | 36.64 | 36.65 |
| | 29.58 | 35.65 | 40.42 | 40.38 |
| | 35.35 | 33.38 | 37.15 | 37.18 |
| 2 | 31.89 | 36.85 | 37.33 | 37.36 |
| | 36.26 | 41.28 | 42.43 | 42.39 |
| | 32.36 | 39.35 | 40.74 | 40.70 |
| 3 | 33.45 | 40.51 | 40.93 | 40.72 |
| | 37.17 | 43.11 | 43.51 | 43.33 |
| | 33.83 | 40.07 | 40.84 | 40.94 |
| 4 | 32.61 | 36.88 | 37.22 | 37.19 |
| | 36.53 | 42.16 | 43.78 | 43.74 |
| | 32.91 | 40.77 | 42.23 | 42.19 |
| 5 | 25.75 | 35.04 | 36.91 | 36.93 |
| | 29.32 | 36.84 | 39.70 | 39.53 |
| | 25.92 | 34.74 | 36.65 | 36.47 |
| 6 | 26.70 | 34.65 | 37.81 | 37.84 |
| | 31.05 | 36.98 | 41.48 | 41.39 |
| | 27.00 | 34.12 | 37.52 | 37.47 |
| 7 | 32.57 | 40.31 | 41.37 | 41.31 |
| | 36.47 | 42.16 | 43.90 | 43.84 |
| | 32.58 | 39.69 | 40.78 | 40.74 |
| 8 | 22.53 | 30.16 | 33.98 | 34.12 |
| | 27.40 | 33.09 | 38.55 | 38.46 |
| | 22.48 | 29.87 | 34.19 | 34.32 |
| 9 | 31.56 | 38.66 | 40.60 | 40.57 |
| | 35.72 | 41.37 | 43.40 | 43.43 |
| | 31.38 | 39.32 | 41.99 | 41.93 |
| 10 | 31.84 | 39.37 | 40.95 | 40.94 |
| | 35.37 | 42.02 | 44.31 | 44.23 |
| | 31.23 | 39.43 | 41.41 | 41.40 |
| 11 | 28.17 | 35.62 | 37.86 | 37.86 |
| | 32.22 | 38.06 | 41.63 | 41.60 |
| | 28.34 | 36.09 | 39.02 | 39.02 |
| 12 | 32.67 | 39.22 | 41.52 | 41.48 |
| | 36.82 | 42.40 | 45.16 | 44.97 |
| | 32.35 | 39.38 | 41.97 | 41.86 |

TABLE 2.1 (cont) PSNR (dB) COMPARISON OF IMAGE DEMOSAICING ALGORITHMS

| Image | Bilinear | ECI | AP (5 ITERATIONS) | AP HARDWARE (5 ITERATIONS) |
|---|---|---|---|---|
| 14 | 23.09 | 31.43 | 33.97 | 34.07 |
| | 26.49 | 32.64 | 36.84 | 36.79 |
| | 23.00 | 30.64 | 33.08 | 33.03 |
| 15 | 28.15 | 34.78 | 34.60 | 34.67 |
| | 32.01 | 37.87 | 38.09 | 38.07 |
| | 28.60 | 35.15 | 35.40 | 35.34 |
| 16 | 28.15 | 36.33 | 37.15 | 37.07 |
| | 32.01 | 41.11 | 42.26 | 42.06 |
| | 28.60 | 39.31 | 40.42 | 40.33 |
| 17 | 30.30 | 37.51 | 41.06 | 41.08 |
| | 34.73 | 40.14 | 44.76 | 44.70 |
| | 30.39 | 37.32 | 40.76 | 40.76 |
| 18 | 31.63 | 39.14 | 40.81 | 40.47 |
| | 34.54 | 40.35 | 43.03 | 43.01 |
| | 30.86 | 38.59 | 40.42 | 40.33 |
| 19 | 27.32 | 34.92 | 36.07 | 36.03 |
| | 30.54 | 36.54 | 39.32 | 39.27 |
| | 26.82 | 34.82 | 36.98 | 36.97 |
| 20 | 26.78 | 34.53 | 37.91 | 37.84 |
| | 31.74 | 37.23 | 42.08 | 42.11 |
| | 26.95 | 34.76 | 39.49 | 39.47 |
| 21 | 30.81 | 38.93 | 40.78 | 40.51 |
| | 34.58 | 40.56 | 43.45 | 43.29 |
| | 30.59 | 37.48 | 39.01 | 38.85 |
| 22 | 27.63 | 35.63 | 38.24 | 38.20 |
| | 31.54 | 37.57 | 41.55 | 41.46 |
| | 27.53 | 34.97 | 37.45 | 37.37 |
| 23 | 29.84 | 36.41 | 37.14 | 36.58 |
| | 33.34 | 38.64 | 39.73 | 39.70 |
| | 29.22 | 35.74 | 36.75 | 36.74 |
| 24 | 34.43 | 40.84 | 40.96 | 40.86 |
| | 37.98 | 43.84 | 43.98 | 43.93 |
| | 34.12 | 41.68 | 41.58 | 41.55 |
| **Ave. R** | **29.15** | **36.47** | **38.20** | **38.12** |
| **Ave. G** | **33.03** | **39.05** | **41.70** | **41.62** |
| **Ave. B** | **29.49** | **36.61** | **38.70** | **38.66** |

The proposed AP hardware is shown in Figure 2.2. It uses parallelism and pipelining to achieve real-time performance. It has four modules (Bilinear Interpolation, Edge-directed Interpolation, Update Green Channel, POCS for Red Channel (R_POCS) and POCS for Blue Channel (B_POCS)) working in parallel. All the data buses between memories and these modules are 8-bit wide. Bilinear Interpolation module has 1 pipeline stage. Edge-directed Interpolation module has 2 pipeline stages. Update Green Channel, R_POCS and B_POCS modules have 8 pipeline stages. The pipeline registers are not shown in the figures.

R_POCS hardware is shown in Figure 2.4. First, rows are filtered with forward filters h0 and h1. Then, columns are filtered with forward filters h0 and h1 which are shown as h0' and h1' in Figure 2.4. Then, rows are filtered with inverse filters g0 and g1. Finally, columns are filtered with inverse filters g0 and g1 which are shown as g0' and g1' in Figure 2.4. B_POCS hardware is the same except B_bil input is used instead of R_bil and B_iter output is produced instead of R_iter. Forward (h0, h1) and inverse (g0, g1) filter hardware are shown in Figure 2.3. Edge-directed Interpolation hardware is shown in Figure 2.5.

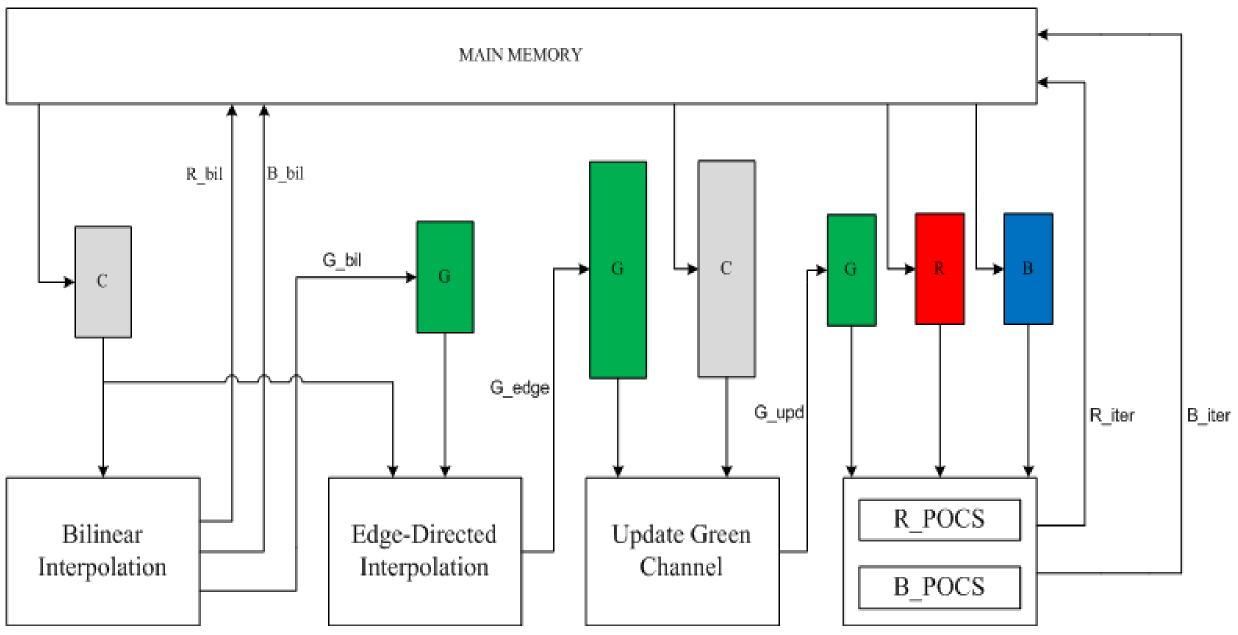


Figure 2.2 Proposed Alternating Projections demosaicing hardware

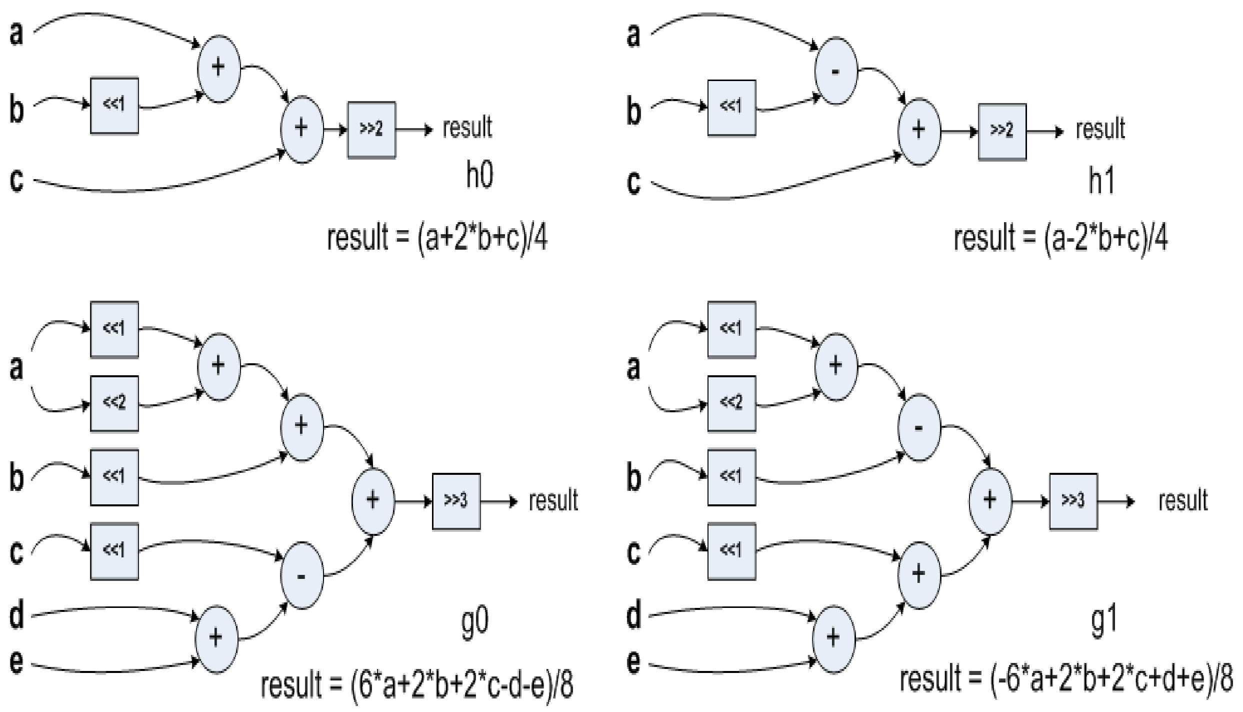


Figure 2.3 Forward and inverse filter hardware

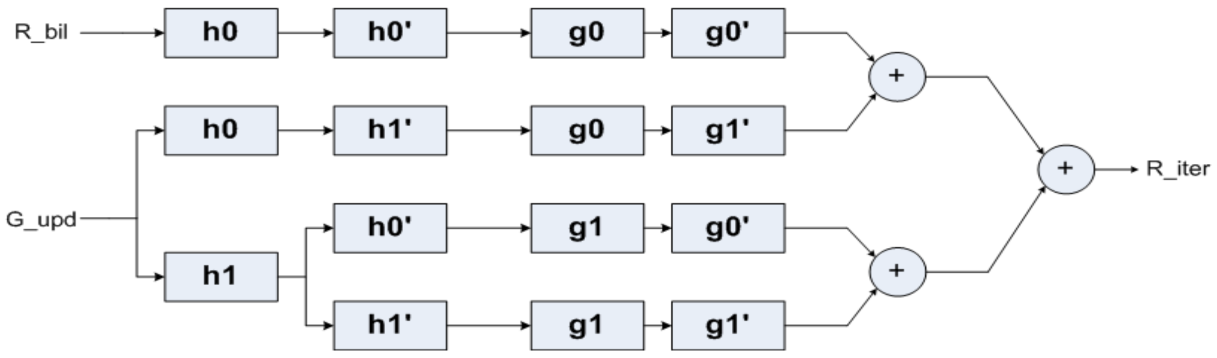


Figure 2.4 POCS for red channel (R_POCS) hardware

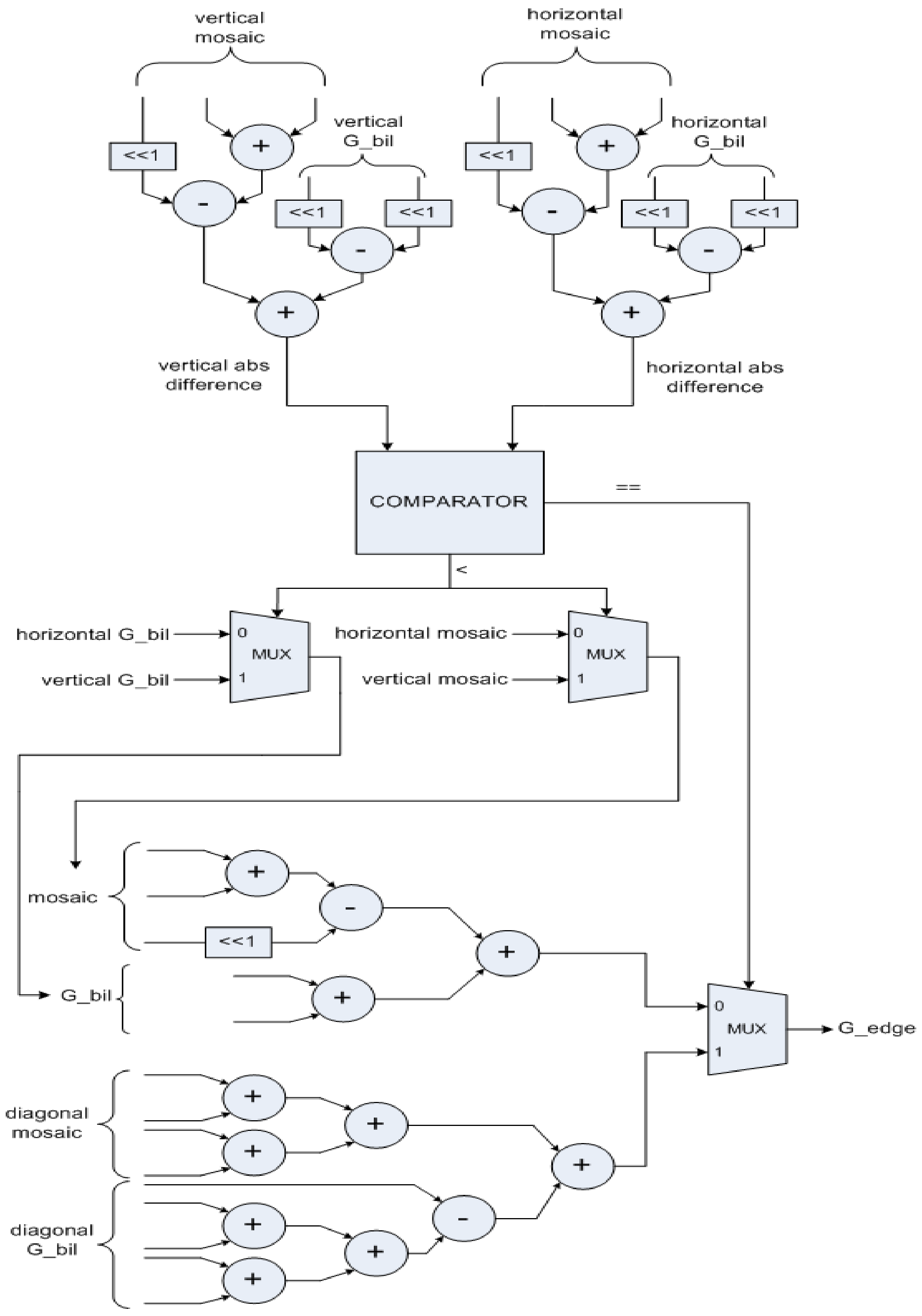


Figure 2.5 Edge-Directed Interpolation hardware

Each module starts working after necessary amount of input data is available in its input on-chip memory, and stores its output data into proper on-chip memory or off-chip main memory. First Bilinear Interpolation, then Edge-directed Interpolation, then Update Green Channel, and finally R_POCS and B_POCS modules start working. They work in parallel until the end of first R_POCS and B_POCS iteration. After the first iteration, only R_POCS and B_POCS modules work in parallel until the end of fifth iteration.

Bilinear interpolation requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. Bilinear Interpolation module starts working after first 2 rows of CFA (mosaic) input image are stored in its C on-chip memory. Edge-directed interpolation uses two input images, CFA (mosaic) input image and green channel reconstructed by bilinear interpolation. It requires the pixels in its 5x5 neighborhood (two neighbors in each of four directions) for reconstructing a pixel.

Edge-directed Interpolation module starts working after 3 rows of bilinear interpolation output are stored in its G on-chip memory. Update Green Channel module uses two input images, CFA (mosaic) input image and green channel reconstructed by edge-directed interpolation. It requires the pixels in its 17x17 neighborhood for reconstructing a pixel. POCS modules use 3 input images, green channel reconstructed by edge-directed interpolation, red and blue channels reconstructed in its previous iteration. It requires the pixels in its 7x7 neighborhood in 3 channels for reconstructing a pixel.

The sizes of the on-chip memories in Figure 2.2 are drawn to scale. In the FPGA implementation, Block RAMs (BRAMs) are used as on-chip memories. As shown in Table 2.2 for an MxN CFA (mosaic) input image, using on-chip BRAMs significantly reduces the amount of necessary off-chip main memory accesses. C input on-chip memory of Bilinear Interpolation module stores 8x768x8 bits (8 rows of CFA (mosaic) input image). G input on-chip memory of Update Green Channel module stores 18x768x8 bits (18 rows of output of Edge-directed Interpolation module), and C input on-chip memory stores 18x768x8 bits (18 rows of CFA (mosaic) input image). The sizes of the remaining 4 on-chip memories are the same as the size of C input on-chip memory of Bilinear Interpolation module, i.e. 8x768x8 bits.

TABLE 2.2 AMOUNT OF OFF-CHIP MAIN MEMORY ACCESS

| **Modules** | **Bilinear Interpolation** | **Edge-Directed Interpolation** | | **Update Green Channel** | | **R_POCS and B_POCS** | | |
|---|---|---|---|---|---|---|---|---|
| Input Images | CFA (Mosaic) | CFA (Mosaic) | Green | CFA (Mosaic) | Green | Red | Green | Blue |
| With Block RAMs | MN | 0 | 0 | MN | 0 | MN | 0 | MN |
| Without Block RAMs | 6 MN | 2.5 MN | 2.5 MN | 49 MN | 49 MN | 49 MN | 49 MN | 49 MN |

8 BRAMs are used to implement C on-chip memory of Bilinear Interpolation module. One of these 8 BRAMs is used to implement a rotating BRAM structure for storing the next row. Similarly, 8 BRAMs are used to implement G on-chip memory of Edge-directed Interpolation module. 2x18=36 BRAMs are used to implement on-chip memories of Update Green Channel module. Two of these 36 BRAMs are used to implement the rotating BRAM structure. 3x8=24 BRAMs are used to implement on-chip memories of POCS modules. Three of these 24 BRAMs are used to implement the rotating BRAM structure. Therefore, 76 BRAMs are used in the proposed AP hardware.

The input data necessary for interpolating the next row are stored in rotating BRAMs, while the current row is being interpolated. Therefore, the rotating BRAM structure lets each module continue interpolating the next row after interpolating the current row without waiting for the input data.

The proposed hardware is implemented using Verilog HDL. The Verilog RTL code is synthesized and mapped to a Xilinx XC6VLX240T FF1156 FPGA with speed grade 1 using Xilinx ISE 12.3. The FPGA implementation works at 83 MHz, and it uses 53584 LUTs, 22573 DFFs and 76 BRAMs. The FPGA implementation is verified to work correctly in a Xilinx Virtex 6 FPGA. Its results matched the results of MATLAB implementation of the same AP algorithm. The FPGA implementation of the AP algorithm with 5 iterations for a 768x512 CFA (mosaic) input image takes 5.95 ms. Therefore, it can process 166 768x512 images per second or 31 full HD (1920x1080) images per second.

In order to determine the speedup achieved by the proposed AP hardware, a C++ software implementation of the same AP algorithm is developed in Microsoft Visual Studio. The results of this software implementation matched the results of the FPGA implementation.

The execution time of the software implementation is measured at 2.4 GHz on an Intel Core i7-3630QM processor with 16 GB DRAM and 64-bit Windows 8 operating system. Text file input and output times are not included in the execution time of the software implementation. The software implementation of the AP algorithm with 5 iterations takes 820 ms. Therefore, the FPGA implementation is 136 times faster than the software implementation.

There are several lower quality image demosaicing hardwares in the literature. The proposed AP image demosaicing hardware produces reconstructed images with much higher quality than them at the expense of increased computational complexity.

# Chapter 3

# PROPOSED EECI IMAGE DEMOSAICING HARDWARE

## 3.1 Effective Color Interpolation Algorithm

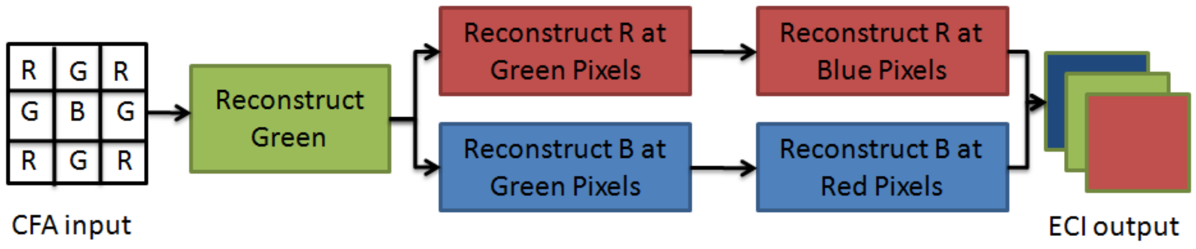
Effective color interpolation (ECI) image demosaicing algorithm takes advantage of the inter-channel correlation between green and blue channels, as well as between green and red channels [7]. ECI flow is shown in Figure 3.1. The reference CFA pattern used for the following ECI calculations and EECI calculations is shown in Figure 3.2. ECI calculates $K_R$ and $K_B$ values for green pixels as shown in Equations 3.1, 3.2, 3.3 and 3.4.

$$\mathrm{K_R} = \mathrm{G} - \mathrm{R} \tag{3.1}$$

$$\mathrm{K_B} = \mathrm{G} - \mathrm{B} \tag{3.2}$$

$$K'_R3 = G3 - R'3 = G3 - \frac{1}{2}(R1 + R7) \tag{3.3}$$

$$K'_R6 = G6 - R'6 = G6 - \frac{1}{2}(R5 + R7) \tag{3.4}$$



R G R
G B G
R G R
CFA input
Reconstruct Green
Reconstruct R at Green Pixels
Reconstruct B at Green Pixels
Reconstruct R at Blue Pixels
Reconstruct B at Red Pixels
ECI output

Figure 3.1 ECI flow

| | | | | |
|---|---|---|---|---|
| R | G | R1 | G | R |
| G | B2 | G3 | B4 | G |
| R5 | G6 | R7 | G8 | R9 |
| G | B10 | G11 | B12 | G |
| R | G | R13 | G | R |

Figure 3.2 Reference CFA pattern for ECI calculations

ECI algorithm first interpolates missing green channels as shown in Equation 3.5. Missing green channels at blue pixels are interpolated similarly using K'$_B$ values. It then interpolates missing red and blue channels as shown in Equations 3.6 and 3.7.

$$G'7 = R7 + \frac{1}{4}(K'_R3 + K'_R6 + K'_R8 + K'_R11) \tag{3.5}$$

$$R'3 = G3 - \frac{1}{2}(K'_R1 + K'_R7) \tag{3.6}$$

$$B'7 = G'7 - \frac{1}{4}(K'_B2 + K'_B4 + K'_B10 + K'_B12) \tag{3.7}$$

## 3.2 Enhanced Effective Color Interpolation Algorithm

ECI algorithm does not take into account the non-uniformity around the edges. This causes zipper effect artifacts. Enhanced ECI (EECI) image demosaicing algorithm uses an adaptive weighted interpolation scheme in order to reduce the zipper effect artifacts around the edges [8]. The objective qualities of the reconstructed images by several image demosaicing algorithms including ECI and EECI are shown in Table 3.1.

TABLE 3.1 PSNR (dB) COMPARISON OF IMAGE DEMOSAICING ALGORITHMS

| Image | Bilinear | AP | ECI | EECI |
|---|---|---|---|---|
| 1 | 25.31 | 36.64 | 33.21 | 37.00 |
| | 29.58 | 40.42 | 35.65 | 40.65 |
| | 35.35 | 37.15 | 33.38 | 37.45 |
| 2 | 31.89 | 37.33 | 36.85 | 38.30 |
| | 36.26 | 42.43 | 41.28 | 43.68 |
| | 32.36 | 40.74 | 39.35 | 41.41 |
| 3 | 33.45 | 40.93 | 40.51 | 42.05 |
| | 37.17 | 43.51 | 43.11 | 45.66 |
| | 33.83 | 40.84 | 40.07 | 41.15 |
| 4 | 32.61 | 37.22 | 36.88 | 39.64 |
| | 36.53 | 43.78 | 42.16 | 42.12 |
| | 32.91 | 42.23 | 40.77 | 41.18 |
| 5 | 25.75 | 36.91 | 35.04 | 37.48 |
| | 29.32 | 39.70 | 36.84 | 40.98 |
| | 25.92 | 36.65 | 34.74 | 36.79 |
| 6 | 26.70 | 37.81 | 34.65 | 37.28 |
| | 31.05 | 41.48 | 36.98 | 41.19 |
| | 27.00 | 37.52 | 34.12 | 37.16 |
| 7 | 32.57 | 41.37 | 40.31 | 41.99 |
| | 36.47 | 43.90 | 42.16 | 45.43 |
| | 32.58 | 40.78 | 39.69 | 41.83 |
| 8 | 22.53 | 33.98 | 30.16 | 34.53 |
| | 27.40 | 38.55 | 33.09 | 38.50 |
| | 22.48 | 34.19 | 29.87 | 34.36 |
| 9 | 31.56 | 40.60 | 38.66 | 42.81 |
| | 35.72 | 43.40 | 41.37 | 45.47 |
| | 31.38 | 41.99 | 39.32 | 41.01 |
| 10 | 31.84 | 40.95 | 39.37 | 42.67 |
| | 35.37 | 44.31 | 42.02 | 45.18 |
| | 31.23 | 41.41 | 39.43 | 41.06 |
| 11 | 28.17 | 37.86 | 35.62 | 38.01 |
| | 32.22 | 41.63 | 38.06 | 42.29 |
| | 28.34 | 39.02 | 36.09 | 39.31 |
| 12 | 32.67 | 41.52 | 39.22 | 41.86 |
| | 36.82 | 45.16 | 42.40 | 45.84 |
| | 32.35 | 41.97 | 39.38 | 41.88 |
| 13 | 23.09 | 33.97 | 31.43 | 34.17 |
| | 26.49 | 36.84 | 32.64 | 36.76 |
| | 23.00 | 33.08 | 30.64 | 33.19 |

TABLE 3.1 (cont) PSNR (dB) COMPARISON OF IMAGE DEMOSAICING ALGORITHMS

| **Image** | **Bilinear** | **AP** | **ECI** | **EECI** |
|---:|---:|---:|---:|---:|
| 14 | 28.15 | 34.60 | 34.78 | 35.30 |
| | 32.01 | 38.09 | 37.87 | 40.56 |
| | 28.60 | 35.40 | 35.15 | 37.01 |
| 15 | 28.15 | 37.15 | 36.33 | 37.98 |
| | 32.01 | 42.26 | 41.11 | 42.22 |
| | 28.60 | 40.42 | 39.31 | 40.12 |
| 16 | 30.30 | 41.06 | 37.51 | 40.27 |
| | 34.73 | 44.76 | 40.14 | 44.05 |
| | 30.39 | 40.76 | 37.32 | 40.23 |
| 17 | 31.63 | 40.81 | 39.14 | 42.35 |
| | 34.54 | 43.03 | 40.35 | 44.24 |
| | 30.86 | 40.42 | 38.59 | 39.70 |
| 18 | 27.32 | 36.07 | 34.92 | 37.86 |
| | 30.54 | 39.32 | 36.54 | 39.60 |
| | 26.82 | 36.98 | 34.82 | 35.76 |
| 19 | 26.78 | 37.91 | 34.53 | 40.00 |
| | 31.74 | 42.08 | 37.23 | 42.83 |
| | 26.95 | 39.49 | 34.76 | 39.05 |
| 20 | 30.81 | 40.78 | 38.93 | 41.59 |
| | 34.58 | 43.45 | 40.56 | 44.37 |
| | 30.59 | 39.01 | 37.48 | 39.65 |
| 21 | 27.63 | 38.24 | 35.63 | 38.66 |
| | 31.54 | 41.55 | 37.57 | 41.83 |
| | 27.53 | 37.45 | 34.97 | 37.72 |
| 22 | 29.84 | 37.14 | 36.41 | 38.07 |
| | 33.34 | 39.73 | 38.64 | 41.16 |
| | 29.22 | 36.75 | 35.74 | 37.61 |
| 23 | 34.43 | 40.96 | 40.84 | 42.27 |
| | 37.98 | 43.98 | 43.84 | 45.61 |
| | 34.12 | 41.58 | 41.68 | 41.96 |
| 24 | 26.40 | 35.10 | 34.37 | 35.16 |
| | 29.38 | 37.46 | 35.54 | 37.85 |
| | 25.29 | 32.88 | 31.94 | 33.04 |
| **Average Red** | **29.15** | **38.20** | **36.47** | **39.05** |
| **Average Green** | **33.03** | **41.70** | **39.05** | **42.42** |
| **Average Blue** | **29.49** | **38.70** | **36.61** | **38.73** |

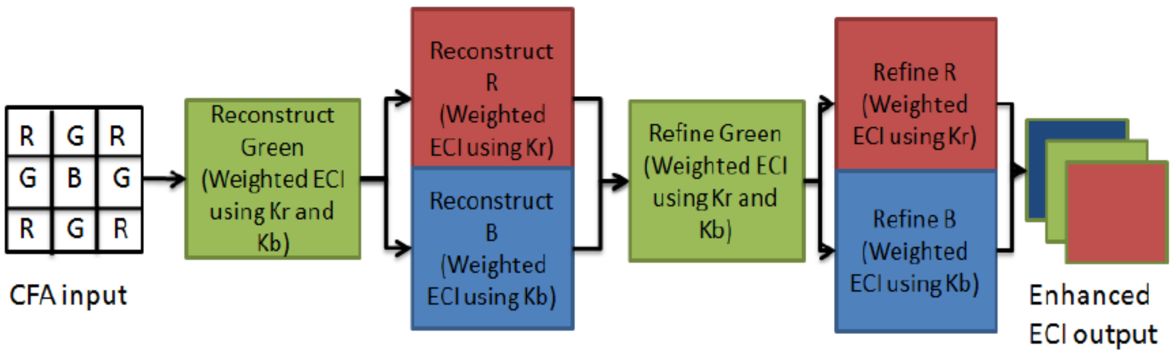

R G R
G B G
R G R
CFA input
Reconstruct Green (Weighted ECI using Kr and Kb)
Reconstruct R (Weighted ECI using Kr)
Reconstruct B (Weighted ECI using Kb)
Refine Green (Weighted ECI using Kr and Kb)
Refine R (Weighted ECI using Kr)
Refine B (Weighted ECI using Kb)
Enhanced ECI output

Figure 3.3 EECI flow

The algorithm consists of two very similar steps, initial step and refinement step. The main difference between them is the color channel values that are used as inputs. In initial step, CFA image is used as input. Therefore, missing color channel values are initially estimated. In refinement step, three fully populated color channels, which are outputs of initial step, are used as input.

**Initial Step**

Initial step starts with interpolation of green channel at red and blue pixels on the CFA image. The pixels on the CFA image are represented as R, G or B indicating that CFA image has red, green or blue channel value at that pixel, respectively. First, weights for four adjacent horizontal and vertical directions are calculated as shown in Equations 3.8, 3.9, 3.10 and 3.11.

$$a_{i-1,j} = |R_{i-2,j} - R_{i,j}| + |G_{i-1,j} - G_{i+1,j}| \tag{3.8}$$

$$a_{i+1,j} = |R_{i+2,j} - R_{i,j}| + |G_{i+1,j} - G_{i-1,j}| \tag{3.9}$$

$$a_{i,j-1} = |R_{i,j-2} - R_{i,j}| + |G_{i,j-1} - G_{i,j+1}| \tag{3.10}$$

$$a_{i,j+1} = |R_{i,j+2} - R_{i,j}| + |G_{i,j+1} - G_{i,j-1}| \tag{3.11}$$

Next, color channel differences for four directions are calculated as shown in Equations 3.12, 3.13, 3.14 and 3.15. The missing red channels are initially estimated as average of neighboring red channels, since they are not present in CFA image. The estimated red channel values are used to calculate the color channel differences for four adjacent directions.

$$K_{R(i-1,j)} = G_{i-1,j} - \frac{1}{2}(R_{i-2,j} + R_{i,j}) \tag{3.12}$$

$$K_{R(i+1,j)} = G_{i+1,j} - \frac{1}{2}(R_{i+2,j} + R_{i,j}) \tag{3.13}$$

$$K_{R(i,j-1)} = G_{i,j-1} - \frac{1}{2}(R_{i,j-2} + R_{i,j}) \tag{3.14}$$

$$K_{R(i,j+1)} = G_{i,j+1} - \frac{1}{2}(R_{i,j+2} + R_{i,j}) \tag{3.15}$$

Next, weighted color channel difference, $K_R$, is calculated as shown in Equation 3.16.

$$K_{R(i,j)} = \frac{\frac{K_{R(i-1,j)}}{1+a_{i-1,j}} + \frac{K_{R(i+1,j)}}{1+a_{i+1,j}} + \frac{K_{R(i,j-1)}}{1+a_{i,j-1}} + \frac{K_{R(i,j+1)}}{1+a_{i,j+1}}}{\frac{1}{1+a_{i-1,j}} + \frac{1}{1+a_{i+1,j}} + \frac{1}{1+a_{i,j-1}} + \frac{1}{1+a_{i,j+1}}} \tag{3.16}$$

Finally, green channel at red pixel is calculated by adding this color channel difference to the red channel value in the CFA image as shown in Equation 3.17.

$$G_{i,j} = R_{i,j} + K_{R(i,j)} \tag{3.17}$$

After green channels at red pixels are interpolated, green channels at blue pixels are interpolated in the same way using blue channel values instead of red channel values.

Initial step continues with interpolation of red and blue channels. Red and blue channels interpolations consist of two different stages. First stage is interpolation of red/blue channels at blue/red pixels on the CFA image. Second stage is interpolation of red and blue channels at green pixels on the CFA image. Both stages are similar to green channel interpolation.

Red channels at blue pixels are interpolated as follows. The outputs of green channel interpolation are used as input. Since green channels are fully populated in green interpolation, there is no need for estimating green channels. Weights for four adjacent diagonal directions are calculated as shown in Equations 3.18, 3.19, 3.20 and 3.21. Since red and blue pixels in CFA image have a diagonal structure, these directions are not horizontal and vertical as in green interpolation.

$$a_{i-1,j-1} = |B_{i-2,j-2} - B_{i,j}| + |R_{i+1,j+1} - R_{i-1,j-1}| \tag{3.18}$$

$$a_{i+1,j+1} = |B_{i+2,j+2} - B_{i,j}| + |R_{i+1,j+1} - R_{i-1,j-1}| \tag{3.19}$$

$$a_{i+1,j-1} = |B_{i+2,j-2} - B_{i,j}| + |R_{i+1,j-1} - R_{i-1,j+1}| \tag{3.20}$$

$$a_{i-1,j+1} = |B_{i-2,j+2} - B_{i,j}| + |R_{i+1,j-1} - R_{i-1,j+1}| \tag{3.21}$$

Color channel differences for four directions are calculated as shown in Equations 3.22, 3.23, 3.24 and 3.25. Red channel values at diagonal directions are obtained from CFA image, and green channel values are obtained from green interpolation output.

$$K_{R(i-1,j-1)} = \mathrm{G}_{i-1,j-1} - R_{i-1,j-1} \tag{3.22}$$

$$K_{R(i+1,j+1)} = \mathrm{G}_{i+1,j+1} - R_{i+1,j+1} \tag{3.23}$$

$$K_{R(i+1,j-1)} = \mathrm{G}_{i+1,j-1} - R_{i+1,j-1} \tag{3.24}$$

$$K_{R(i-1,j+1)} = \mathrm{G}_{i-1,j+1} - R_{i-1,j+1} \tag{3.25}$$

Weighted color channel difference is calculated as shown in Equation 3.26.

$$K_{R(i,j)} = \frac{\frac{K_{R(i-1,j-1)}}{1+a_{i-1,j-1}} + \frac{K_{R(i+1,j+1)}}{1+a_{i+1,j+1}} + \frac{K_{R(i+1,j-1)}}{1+a_{i+1,j-1}} + \frac{K_{R(i-1,j+1)}}{1+a_{i-1,j+1}}}{\frac{1}{1+a_{i-1,j-1}} + \frac{1}{1+a_{i+1,j+1}} + \frac{1}{1+a_{i+1,j-1}} + \frac{1}{1+a_{i-1,j+1}}} \tag{3.26}$$

Red channel values at blue pixels are calculated as shown in Equation 3.27.

$$R_{i,j} = G_{i,j} - K_{R(i,j)} \tag{3.27}$$

Blue channels at red pixels are interpolated in the same way but with different orientation of inputs, and with red channel values used as inputs instead of blue channel values and vice versa.

Red and blue channels at green pixels are interpolated similar to green channel interpolation at red and blue pixels, red/blue channel interpolation at blue/red pixels. Weights for four adjacent horizontal and vertical directions are calculated as shown in Equations 3.28, 3.29, 3.30 and 3.31. Since green, red and blue channels, except red and blue channels at green pixels, are fully populated in previous interpolations, there is no need for estimating channel values.

$$a_{i-1,j} = \left|G_{i-2,j} - G_{i,j}\right| + \left|R_{i-1,j} - R_{i+1,j}\right| \tag{3.28}$$

$$a_{i+1,j} = \left|G_{i+2,j} - G_{i,j}\right| + \left|R_{i-1,j} - R_{i+1,j}\right| \tag{3.29}$$

$$a_{i,j-1} = \left|G_{i,j-2} - G_{i,j}\right| + \left|R_{i,j-1} - R_{i,j+1}\right| \tag{3.30}$$

$$a_{i,j+1} = \left|G_{i,j+2} - G_{i,j}\right| + \left|R_{i,j-1} - R_{i,j+1}\right| \tag{3.31}$$

Color channel differences for four directions are calculated as shown in Equations 3.32, 3.33, 3.34 and 3.35.

$$K_{R(i-1,j)} = \mathrm{G}_{i-1,j} - R_{i-1,j} \tag{3.32}$$

$$K_{R(i+1,j)} = \mathrm{G}_{i+1,j} - R_{i+1,j} \tag{3.33}$$

$$K_{R(i,j-1)} = \mathrm{G}_{i,j-1} - R_{i,j-1} \tag{3.34}$$

$$K_{R(i,j+1)} = \mathrm{G}_{i,j+1} - R_{i,j+1} \tag{3.35}$$

Weighted color channel difference is calculated as shown in Equation 3.36.

$$K_{R(i,j)} = \frac{\frac{K_{R(i-1,j)}}{1+a_{i-1,j}} + \frac{K_{R(i+1,j)}}{1+a_{i+1,j}} + \frac{K_{R(i,j-1)}}{1+a_{i,j-1}} + \frac{K_{R(i,j+1)}}{1+a_{i,j+1}}}{\frac{1}{1+a_{i-1,j}} + \frac{1}{1+a_{i+1,j}} + \frac{1}{1+a_{i,j-1}} + \frac{1}{1+a_{i,j+1}}} \tag{3.36}$$

Red channel values at green pixels are calculated as shown in Equation 3.37.

$$R_{i,j} = G_{i,j} - K_{R(i,j)} \tag{3.37}$$

Blue channels at green pixels are interpolated in the same way using blue channel values instead of red channel values.

**Refinement Step**

After initial step, the three color channels are fully populated, and the results can be refined by using the same interpolation scheme for the three color channels to generate higher quality output image. There are minor differences between initial step and refinement step. In green channel interpolation of initial step, missing color channel values must be initially estimated since they are not fully populated yet. This is not needed in green channel interpolation of refinement step since the three color channels are fully populated in initial step. Input color channels used in initial step and refinement step are different. Initial step uses CFA image while refinement step uses three fully populated color channels which are outputs of initial step.

## 3.3 Proposed EECI Demosaicing Hardware

Several minor modifications are done to EECI algorithm to increase the performance and reduce the area and power consumption of proposed EECI hardware. First modification is to change the double precision variables used for storing the results of division operations to fixed point variables. The size of integer part of each fixed point variable is set to the minimum value required to avoid overflow, and the size of fractional parts of all fixed point variables is set to 10 bits.

Second modification is to change input and output connections of some modules. As shown in Figure 3.4, in EECI algorithm, output of Refine Red at Blue Pixel module is used as input of Refine Red at Green Pixel module. As shown in Figure 3.5, in refinement step of the modified EECI algorithm, red channel output of initial step (R2) is used instead of output of Refine Red at Blue Pixel module. The input of Refine Blue at Green Pixel module is changed similarly. These changes remove the data dependencies among the modules in refinement step other than Refine Green module. Therefore, these modules work in parallel and there is no need to store outputs of Refine Red at Blue Pixel and Refine Blue at Red Pixel modules in Block RAMs, and send them to Refine Red at Green Pixel and Refine Blue at Green Pixel modules.

Last modification is to reduce the number of inputs of several modules to avoid corresponding multiplexers, registers and input connections for these modules: In Reconstruct Red at Green Pixel, Reconstruct Blue at Green Pixel, Refine Red at Green Pixel and Refine Blue at Green Pixel modules, EECI algorithm uses two different green values to compute weights and color channel differences for four adjacent directions. G(i-2,j), G(i+2,j), G(i,j-2) and G(i,j+2) are used to compute weights for four adjacent directions. G(i-1,j), G(i+1,j), G(i,j-1) and G(i,j+1) are used to compute color channel differences for four adjacent directions. In the modified EECI algorithm, G(i-1,j), G(i+1,j), G(i,j-1) and G(i,j+1) are used to compute both weights and color channel differences for four adjacent directions. These 4 green inputs are used instead of the removed green inputs in the calculations. Similarly, the number of red and blue inputs is reduced in Refine Green module.

The PSNR (dB) performances of original EECI algorithm and modified EECI algorithm implemented in the proposed EECI hardware for the 24 images in the Kodak Image Suite [10] are shown in Table 3.2. These results show that the modifications have negligible impact on the performance of the EECI algorithm.

The proposed EECI hardware is shown in Figure 3.5. The modules in the hardware are shown in Figures 3.6, 3.7, 3.8 and 3.9. All data buses between memories and these modules are 8-bit wide. All modules in the hardware work in parallel except for a short time at the beginning and at the end because of the data dependencies between them. Each module starts working after necessary amount of input data is available in its input on-chip memory, and stores its output data into on-chip input memory of the next module. First, Reconstruct Green module starts working. The output of the refinement step is stored into off-chip main memory.

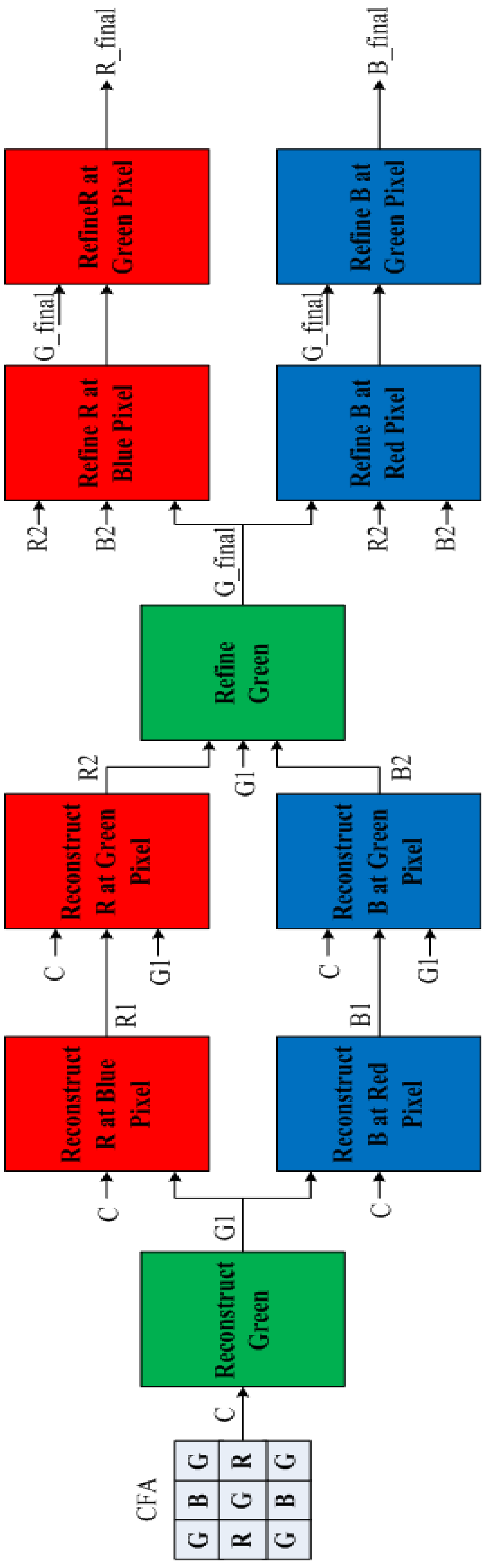


Figure 3.4 EECI algorithm

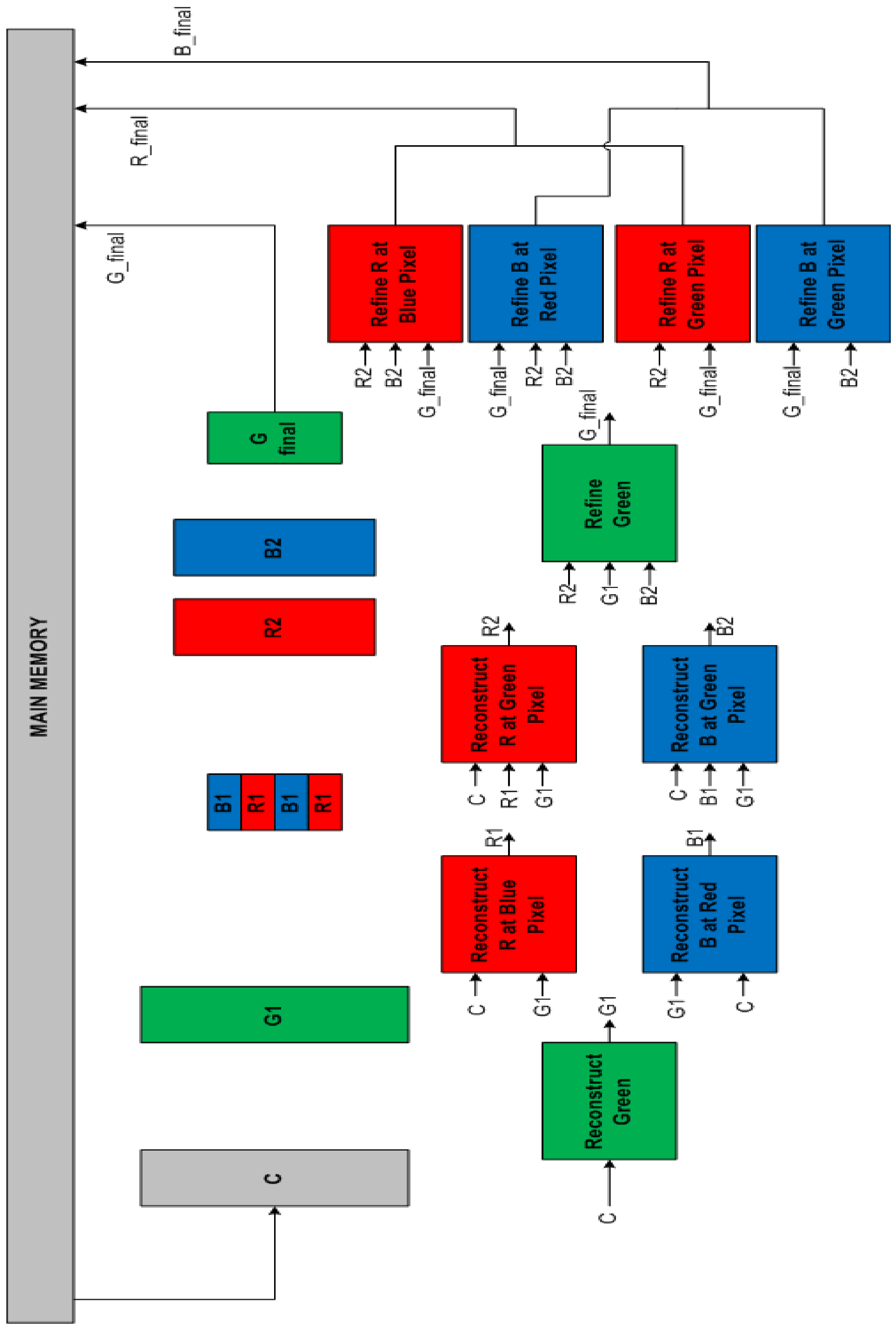


Figure 3.5 Proposed EECI demosaicing hardware

TABLE 3.2 PSNR (dB) COMPARISON OF EECI DEMOSAICING ALGORITHM AND EECI DEMOSAICING HARDWARE

| Image | EECI | EECI Hardware | | Image | EECI | EECI Hardware |
|---|---|---|---|---|---|---|
| 1 | 37.00 | 36.66 | | 13 | 34.17 | 33.80 |
| | 40.65 | 40.85 | | | 36.76 | 37.00 |
| | 37.45 | 37.03 | | | 33.19 | 32.86 |
| 2 | 38.30 | 38.10 | | 14 | 35.30 | 35.42 |
| | 43.68 | 43.47 | | | 40.56 | 40.67 |
| | 41.41 | 40.97 | | | 37.01 | 36.97 |
| 3 | 42.05 | 41.88 | | 15 | 37.98 | 37.87 |
| | 45.66 | 45.90 | | | 42.22 | 42.22 |
| | 41.15 | 41.26 | | | 40.12 | 39.66 |
| 4 | 39.64 | 39.79 | | 16 | 40.27 | 40.09 |
| | 42.12 | 41.91 | | | 44.05 | 44.27 |
| | 41.18 | 40.67 | | | 40.23 | 40.05 |
| 5 | 37.48 | 37.31 | | 17 | 42.35 | 42.08 |
| | 40.98 | 41.08 | | | 44.24 | 44.34 |
| | 36.79 | 36.57 | | | 39.70 | 39.41 |
| 6 | 37.28 | 37.09 | | 18 | 37.86 | 37.73 |
| | 41.19 | 41.38 | | | 39.60 | 39.85 |
| | 37.16 | 36.88 | | | 35.76 | 35.59 |
| 7 | 41.99 | 41.88 | | 19 | 40.00 | 39.76 |
| | 45.43 | 45.50 | | | 42.83 | 42.93 |
| | 41.83 | 41.83 | | | 39.05 | 38.70 |
| 8 | 34.53 | 34.40 | | 20 | 41.59 | 41.32 |
| | 38.50 | 38.87 | | | 44.37 | 44.33 |
| | 34.36 | 34.13 | | | 39.65 | 39.48 |
| 9 | 42.81 | 42.67 | | 21 | 38.66 | 38.37 |
| | 45.47 | 45.61 | | | 41.83 | 41.95 |
| | 41.01 | 40.74 | | | 37.72 | 37.44 |
| 10 | 42.67 | 42.61 | | 22 | 38.07 | 37.71 |
| | 45.18 | 45.34 | | | 41.16 | 41.14 |
| | 41.06 | 40.75 | | | 37.61 | 37.33 |
| 11 | 38.01 | 37.89 | | 23 | 42.27 | 42.18 |
| | 42.29 | 42.45 | | | 45.61 | 45.63 |
| | 39.31 | 39.04 | | | 41.96 | 41.96 |
| 12 | 41.86 | 41.60 | | 24 | 35.16 | 34.88 |
| | 45.84 | 45.88 | | | 37.85 | 37.92 |
| | 41.88 | 41.53 | | | 33.04 | 32.77 |
| | | | | **Red** | **39.05** | **38.88** |
| | | | | **Green** | **42.42** | **42.52** |
| | | | | **Blue** | **38.73** | **38.48** |

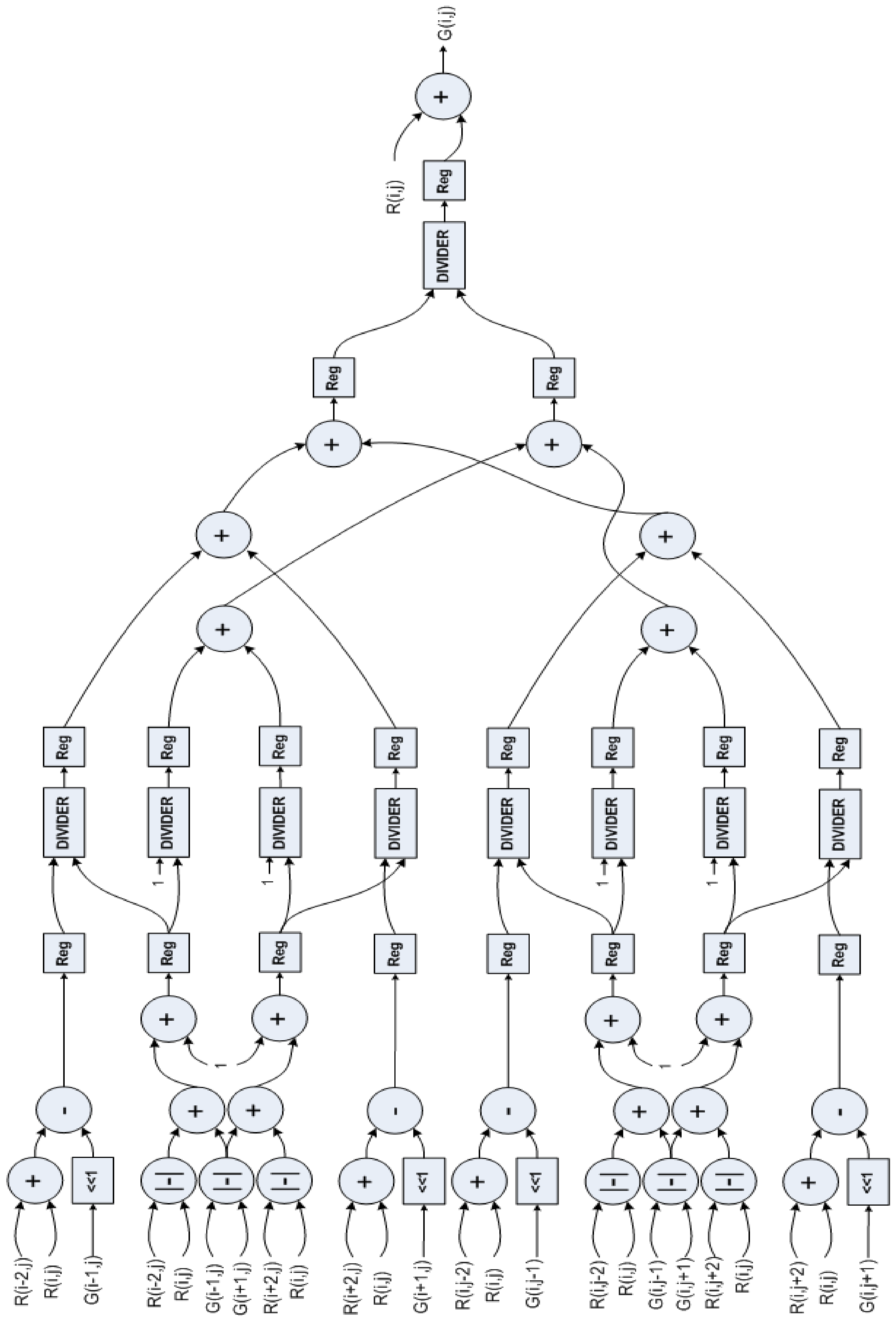


Figure 3.6 Reconstruct Green hardware

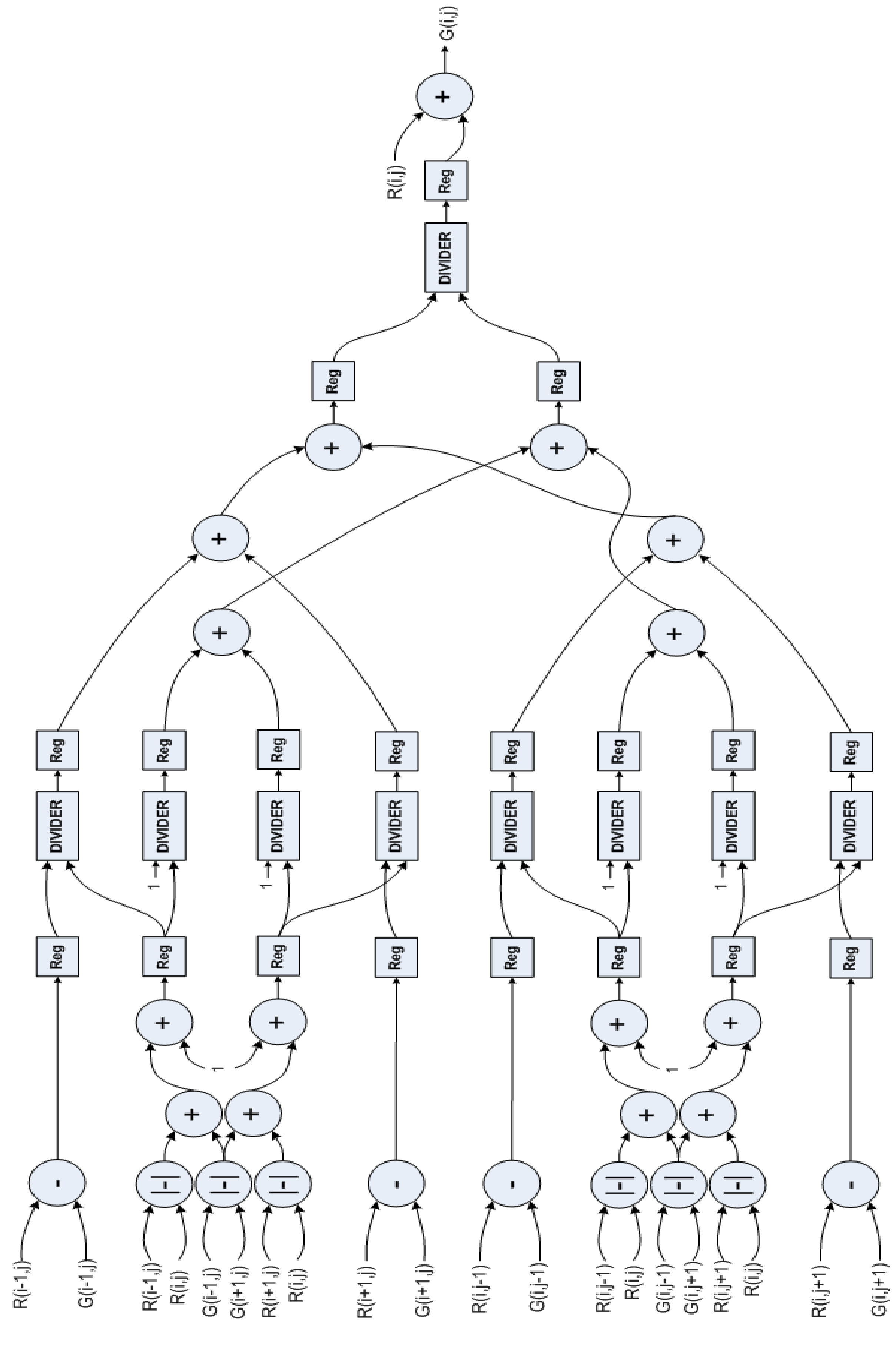


Figure 3.7 Refine Green hardware

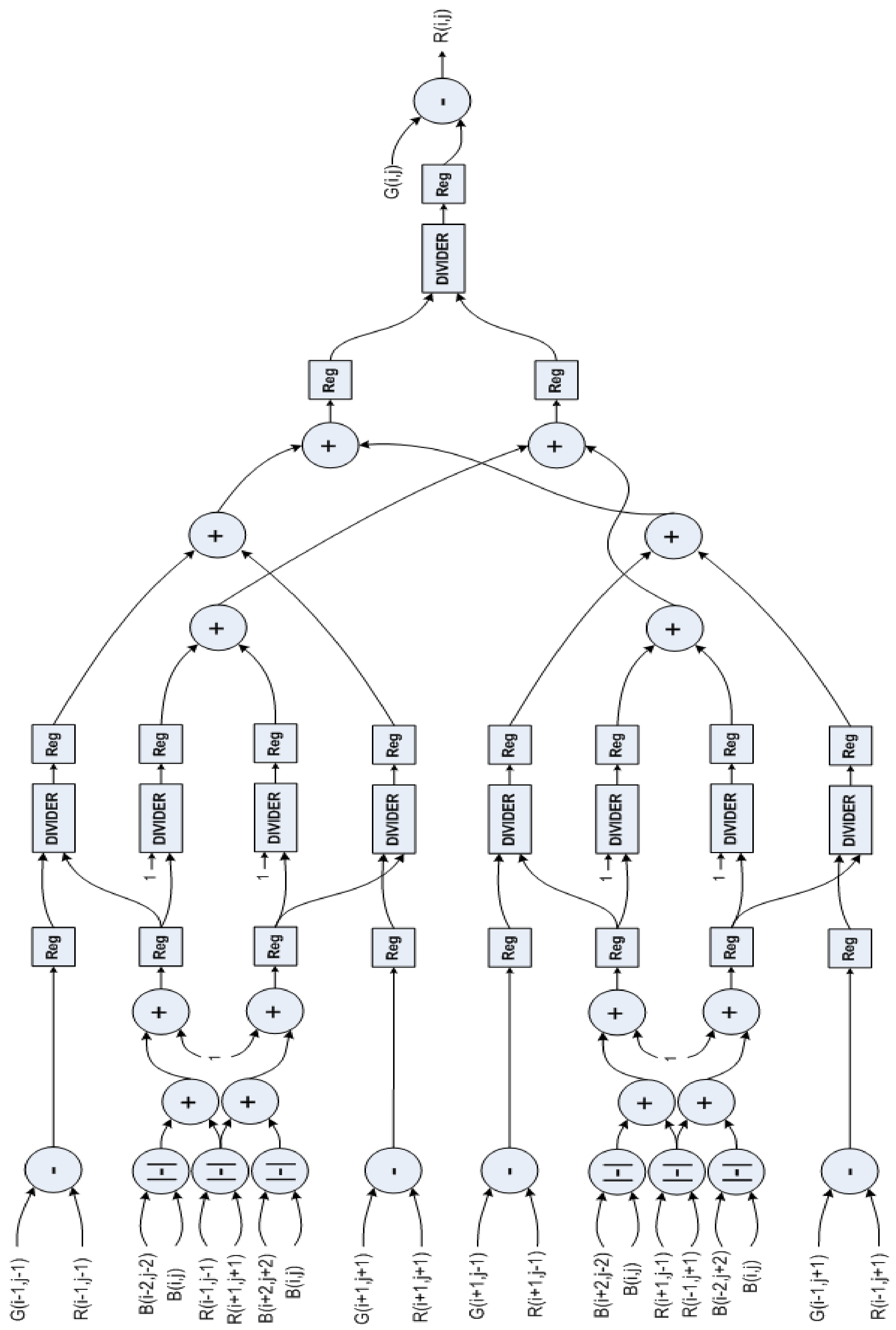


Figure 3.8 Reconstruct / Refine Red at Blue Pixel hardware

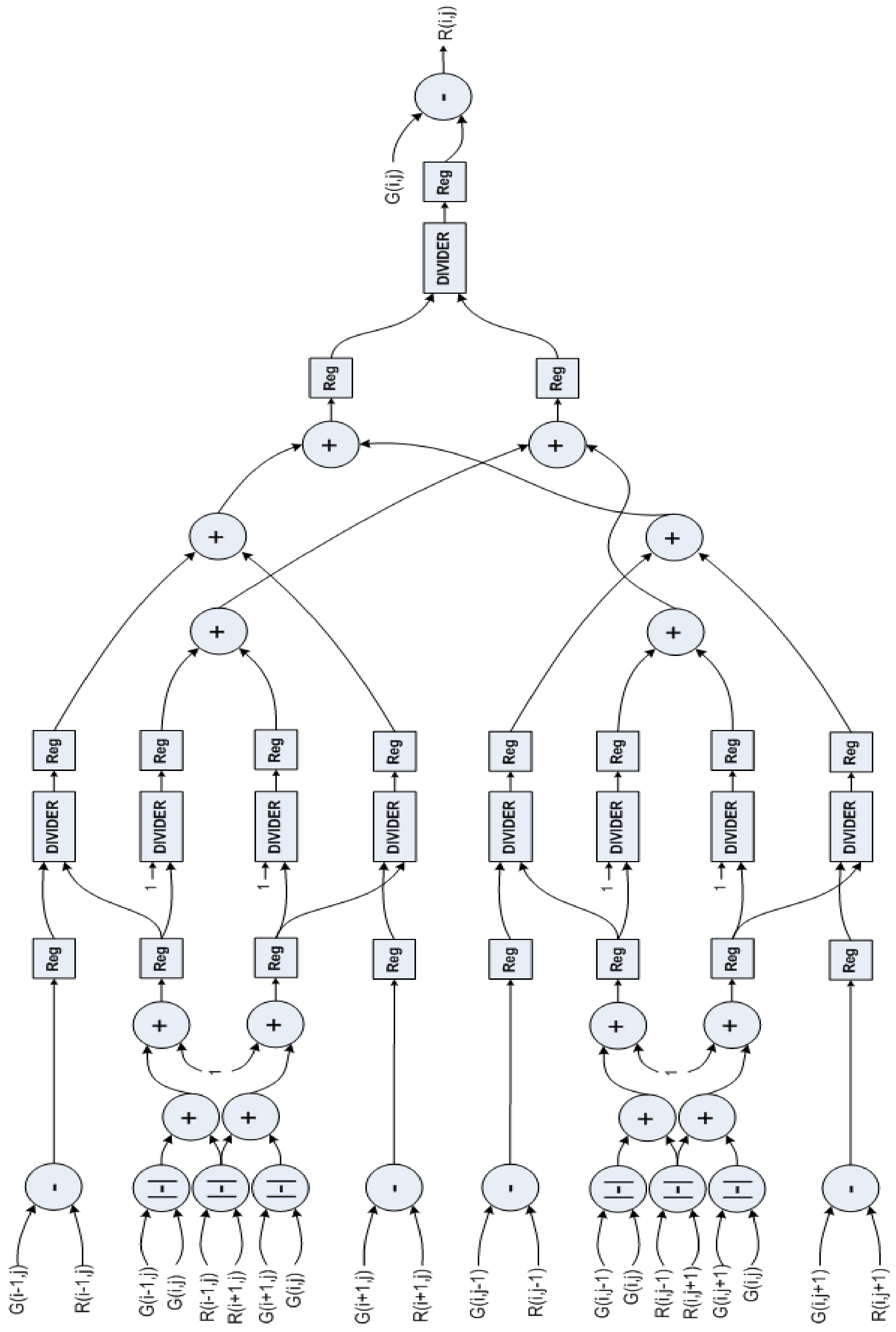


Figure 3.9 Reconstruct / Refine Red at Green Pixel hardware

Reconstruct Green module requires the pixels in its 5x5 neighborhood (two neighbors in each of four directions) for reconstructing a pixel. It starts working after first three rows of CFA (mosaic) input image are stored into C on-chip memory.

Reconstruct Red at Blue Pixel module requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. Reconstruct Blue at Red Pixel module also requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. Both modules start working after first two rows of the output of Reconstruct Green module are stored into G1 on-chip memory. Reconstruct Red at Blue Pixel module interpolates only half of red channels at even rows, and stores the results into even rows of R1/B1 on-chip memory. Similarly, Reconstruct Blue at Red Pixel module interpolates only half of blue channels at odd rows, and stores the results into odd rows of R1/B1 on-chip memory.

Reconstruct Red at Green Pixel module requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. Reconstruct Blue at Green Pixel module also requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. Both modules start working after first two rows of the outputs of the previous modules are stored into R1/B1 on-chip memory.

Refine Green module requires the pixels in its 3x3 neighborhood (one neighbor in each of four directions) for reconstructing a pixel. It starts working after first two rows of the outputs of the previous modules are stored into R2 and B2 on-chip memories.

Refine Red at Blue Pixel, Refine Red at Green Pixel, Refine Blue at Red Pixel and Refine Blue at Green Pixel modules require the pixels in their 3x3 neighborhoods (one neighbor in each of four directions) for reconstructing a pixel. They all start working after first two rows of the output of Refine Green module are stored into G_last on-chip memory.

As shown in Figures 3.6 - 3.9, a divider module is used in the proposed EECI demosaicing hardware. The divider module takes two fixed point variables as input, divides one variable with the other, and outputs a fixed point variable. Pseudo-code of the division operation without fractional part calculation is shown below. The division operation is done using addition, subtraction, comparison and shift operations. There is a condition for the divider module to work correctly. Divisor and quotient should have the same number of bits, and dividend should have twice as many bits as divisor. The number of bits in fractional part of the output is determined by a shift in dividend. The number of bit shifts in dividend results in the same number of fractional bits in the output. Dividend inputs to divider modules are shifted 10 bits to left in the EECI demosaicing hardware. Therefore, the fractional parts of divider outputs are 10 bits.

```
bit_diff    = num_bits(dividend) - num_bits(divisor);
dvs         = divisor << (bit_diff-1);
dvd         = (dividend – dvs) << 1;
result      = 0;
for n = 1:bit_diff
    if(dvd >= 0)
        result  = result << 1 + 1;
        dvd     = dvd - dvs;
    else
        result  = result << 1;
        dvd     = dvd + dvs;
    end
    dvd     = dvd << 1;
end
```

The sizes of the on-chip memories in Figure 3.5 are drawn to scale. C on-chip memory stores 8x768x8 bits (8 rows of CFA (mosaic) input image). G1 on-chip memory stores 8x768x8 bits (8 rows of output of Reconstruct Green module). R1/B1 on-chip memory stores 4x384x8 bits (2 rows of output of Reconstruct Red at Blue Pixel module and 2 rows of output of Reconstruct Blue at Red Pixel module). R2 on-chip memory stores 6x768x8 bits (6 rows of output of Reconstruct Red at Green Pixel module). B2 on-chip memory stores 6x768x8 bits (6 rows of output of Reconstruct Blue at Green Pixel module). G_final on-chip memory stores 4x768x8 bits (4 rows of output of Refine Green module).

In the FPGA implementation, Block RAMs (BRAMs) are used as on-chip memories. 2x8=16 BRAMs are used to implement C and G1 on-chip memories. Two of these 16 BRAMs are used to implement a rotating BRAM structure for storing the next row. 4 BRAMs are used to implement R1/B1 on-chip memory. One of these 4 BRAMs is used to implement the rotating BRAM structure. 2x6=12 BRAMs are used to implement R2 and B2 on-chip memories. Two of these 12 BRAMs are used to implement the rotating BRAM structure. 4 BRAMs are used to implement G_final on-chip memory. One of these 4 BRAMs is used to implement the rotating BRAM structure. Therefore, 36 BRAMs are used in the proposed EECI hardware.

The proposed EECI hardware is implemented using Verilog HDL. The Verilog RTL code is synthesized and mapped to a Xilinx XC6VLX240T FF1156 FPGA with speed grade 1 using Xilinx ISE 13.4. The FPGA implementation works at 100 MHz and it uses 49456 LUTs, 32449 DFFs and 36 BRAMs. The FPGA implementation is verified to work correctly in a Xilinx Virtex 6 FPGA. Its results matched the results of MATLAB implementation of the same EECI algorithm. The FPGA implementation processes a 768x512 CFA (mosaic) input image in 2.005 ms. Therefore, it can process 498 768x512 images per second or 94 full HD (1920x1080) images per second.

In order to determine the speedup achieved by the proposed EECI hardware, a C++ software implementation of the same EECI algorithm is developed in Microsoft Visual Studio. The results of this software implementation matched the results of the FPGA implementation. The execution time of the software implementation is measured at 2.4 GHz on an Intel Core i7-3630QM processor with 16 GB DRAM and 64-bit Windows 8 operating system. Reading and writing input and output text files are not included in measuring the execution time of the software implementation. The software implementation of EECI algorithm takes 210 ms. Therefore, the FPGA implementation is 105 times faster than the software implementation.

As shown in Table 3.3, reconstructed image quality of demosaicing hardware in the literature are worse than that of the proposed EECI hardware. The demosaicing hardware proposed in [11]-[13] implement bilinear interpolation algorithm, but their PSNR results are not given. Therefore, PSNR results of bilinear interpolation algorithm are used to compare reconstructed image quality of the demosaicing hardware proposed in [11]-[13] and the proposed EECI hardware.

The demosaicing hardware proposed in [14] implements bilinear interpolation algorithm together with a median filter. PSNR result for only one image is given as 35.15, and it is not specified as PSNR of any color (red, green or blue) channel. Therefore, this PSNR value is given in the ‘Not Specified’ column in Table 3.3. The demosaicing hardware proposed in [15] implements Adams-Hamilton algorithm. Reconstructed image quality is evaluated with three images named “Woman”, “Battle” and “Green Mountains” which are not in Kodak Image Suite. PSNR results for these three images are given as 34.21, 41.93 and 31.25, but they are not specified as PSNR of any color (red, green or blue) channel. Therefore, average of these three values is given in the ‘Not Specified’ column in Table 3.3.

In Table 3.3, PSNR values given for the demosaicing hardware proposed in [17] and [18] are averages of PSNR values for the images used in these papers which are not in Kodak Image Suite. PSNR values given for Bilinear Interpolation, AP, EECI and demosaicing hardware proposed in [20] are averages of PSNR values for 24 images in Kodak Image Suite.

AP and EECI demosaicing hardware proposed in this thesis are compared with ECI, modified EECI and hybrid ECI & AP demosaicing hardware proposed in [21] in terms of reconstructed image quality, hardware area and hardware performance in processed images per second (ips). The comparison results are given in Table 3.4. PSNR values given in the table are averages of PSNR values for 24 images in Kodak Image Suit. The proposed AP and EECI demosaicing hardware have considerably better reconstructed image quality than the three image demosaicing hardware proposed in [21]. However, the proposed AP and EECI demosaicing hardware have larger area and lower performance than them. This is because AP and EECI image demosaicing algorithms have higher computational complexity than the image demosaicing algorithms implemented in [21].

Table 3.3 AVERAGE PSNR VALUES OF IMAGE DEMOSAICING HARDWARE

| **DEMOSAICING HARDWARE** | **AVERAGE RED** | **AVERAGE GREEN** | **AVERAGE BLUE** | **NOT SPECIFIED** |
|---|---|---|---|---|
| **BILINEAR INTERPOLATION [11][12][13]** | 29.12 | 33.04 | 29.49 | |
| **BILINEAR INTERPOLATION + MEDIAN FILTER [14]** | | | | 35.15 |
| **ADAMS-HAMILTON [15]** | | | | 35.80 |
| **[17]** | 28.37 | 30.80 | 29.57 | |
| **[18]** | 34.14 | 40.91 | 33.85 | |
| **PROPOSED AP** | 38.12 | 41.62 | 38.66 | |
| **[20]** | 38.60 | 42.27 | 39.01 | |
| **PROPOSED EECI** | 38.88 | 42.52 | 38.48 | |

Table 3.4 COMPARISON OF PROPOSED DEMOSAICING HARDWARE WITH DEMOSAICING HARDWARE PROPOSED IN [21]

| | **ECI [21]** | **MODIFIED EECI [21]** | **HYBRID ECI & AP [21]** | **PROPOSED AP** | **PROPOSED EECI** |
|---|---|---|---|---|---|
| **RED (PSNR)** | 36.14 | 37.62 | 36.14 | 38.12 | 38.88 |
| **GREEN (PSNR)** | 39.05 | 39.69 | 40.93 | 41.62 | 42.52 |
| **BLUE (PSNR)** | 36.93 | 37.19 | 36.93 | 38.66 | 38.48 |
| **DFF** | 655 | 3046 | 6388 | 22573 | 32449 |
| **LUT** | 800 | 17446 | 8142 | 53584 | 49456 |
| **BRAM** | 10 | 40 | 20 | 76 | 36 |
| **IPS** | 160 FULL HD | 118 FULL HD | 119 FULL HD | 31 FULL HD | 94 FULL HD |

# Chapter 4

# CONCLUSION AND FUTURE WORK

In this thesis, first, we proposed a high performance AP image demosaicing hardware. This is the first AP image demosaicing hardware in the literature. It uses parallelism and pipelining to achieve real-time performance. The proposed hardware architecture is implemented using Verilog HDL on a Xilinx Virtex 6 FPGA. The proposed FPGA implementation can process 31 full HD (1920x1080) images per second. It is 136 times faster than a C++ software implementation of AP algorithm running at 2.4 GHz on an Intel Core i7 processor with 16 GB DRAM.

Then, we proposed a high performance EECI image demosaicing hardware. This is the first EECI image demosaicing hardware in the literature. It uses parallelism and pipelining to achieve real-time performance. The proposed hardware architecture is implemented using Verilog HDL on a Xilinx Virtex 6 FPGA. The proposed FPGA implementation can process 94 full HD (1920x1080) images per second. It is 105 times faster than a C++ software implementation of EECI algorithm running at 2.4 GHz on an Intel Core i7 processor with 16 GB DRAM.

As future work, the power and energy consumptions of the proposed AP and EECI demosaicing hardware can be estimated. Their power and energy consumptions can be reduced. A high performance high quality hybrid AP and EECI image demosaicing hardware can be designed and implemented. This hybrid AP and EECI image demosaicing hardware can either have better reconstructed image quality or smaller hardware area than the proposed AP and EECI image demosacing hardware.

# BIBLIOGRAPHY

[1] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, "Demosaicking: Color Filter Array Interpolation," *IEEE Signal Processing Magazine, Vol. 22, No.1*, pp. 44-54, Jan. 2005.

[2] B. E. Bayer, "Color Imaging Array," *U.S. Patent No. 3 971 065*, Jul. 1976.

[3] S. Yamanaka, "Solid State Camera," *U.S. Patent 4 054 906*, Oct. 1977.

[4] R. Lukac and K. N. Plataniotis, "Color Filter Arrays: Design and Performance Analysis," *IEEE Transactions on Consumer Electronics, Vol. 51, No. 4*, pp. 1260–1267, Nov. 2005.

[5] M. Parmar and S. J. Reeves, "A Perceptually Based Design Methodology For Color Filter Arrays," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Vol. 3,* pp.473–476, May 2004.

[6] K. Hirakawa and P. J. Wolfe, "Spatio-spectral Color Filter Array Design For Optimal Image Recovery," *IEEE Transactions on Image Processing, Vol. 17, No.10*, pp. 1876–1890, Oct. 2008.

[7] S. Pei, I. Tam, "Effective Color Interpolation in CCD Color Filter Arrays Using Signal Correlation," *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 6,* pp. 503-513, June 2003.

[8] L. Chang, Y. Tan, "Effective Use of Spatial and Spectral Correlations for Color Filter Array Demosaicking," *IEEE Transactions on Consumer Electronics, Vol. 50, No. 1,* pp. 355-365, Feb. 2004.

[9] B. K. Gunturk, Y.Altunbasak, R. M. Mersereau, "Color Plane Interpolation Using Alternating Projections," *IEEE Transactions on Image Processing, Vol. 11, No. 9*, pp. 997-1013, Sep. 2002.

[10] Eastman Kodak Company, *Kodak Lossless True Color Image Suite*, <http://r0k.us/graphics/kodak/>, Retrieved Aug. 2011.

[11] K. S. Rani, W. J. Hans, "FPGA Implementation of Bilinear Interpolation Algorithm for CFA Demosaicing," *International Conference on Communications and Signal Processing (ICCSP)*, pp. 857-863, April 2013.

[12] I. O. H. Fuentes, M. E. Bravo-Zanoguera, G. G. Yanez, “FPGA Implementation of the Bilinear Interpolation Algorithm for Image Demosaicking,” *International Conference on Electrical, Communications, and Computers*, pp. 25-28, Feb. 2009.

[13] G. L. Jair, A. A. Miguel, W. V. Julia, “A Digital Real Time Image Demosaicking Implementation for High Definition Video Cameras,” *Electronics, Robotics and Automotive Mechanics Conference*, pp. 565-569, Oct. 2008.

[14] J. M . Perez, P. Sanchez, M. Martinez, “Low-Cost Bayer to RGB Bilinear Interpolation with Hardware-Aware Median Filter,” 16th *IEEE International Conference on Electronics, Circuits, and Systems*, pp. 916-919, Dec. 2009.

[15] J. Khalifat, A. Ebrahim, T. Arslan, “An Efficient Implementation of the Adams-Hamilton’s Demosaicing Algorithm in FPGAs,” *Reconfigurable Computing: Architectures, Tools, and Appplications, Lecture Notes in Computer Science, Vol. 8405,* pp. 205-212, April 2014.

[16] H. Azgin, S. Yaliman, I. Hamzaoglu, “A High Performance Alternating Projections Image Demosaicing Hardware,” *24th International Conference on Field Programmable Logic and Applications (FPL)*, Sept. 2014.

[17] S. C. Hsia, M. H. Chen, P. S. Tsai, “VLSI Implementation od Low-Power High-Quality Color Interpolation Processor for CCD Camera,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 4,* pp. 361-369, April 2006.

[18] A. Karloff, R. Muscedere, “A Low-Cost, Real-Time, Hardware-Based Image Demosaicking Algorithm,” *IEEE International Conference on Electro/Information Technology,* pp. 146-150, June 2009.

[19] J. Liu, Y. Pan, W. Ding, R. Huan, “Cost Effective Hardware Based Demosaicking Algorithm for Embedded System,” *International Conference on Wireless Communications & Signal Processing (WCSP),* pp. 1-6, Oct. 2013.

[20] T. H. Chen, S. Y. Chien, “Cost Effective Color Filter Array Demosaicking with Chrominance Variance Weighted Interpolation,” *IEEE International Symposium on Circuits and Systems,* pp. 1277-1280, May 2007.

[21] S. Yalıman, “High Performance Image Demosaicing Hardware Designs”, MS Thesis, Sabancı University, Jan. 2014.