A High Performance Alternating Projections Image Demosaicing Hardware

Hasan Azgin, Serkan Yaliman, Ilker Hamzaoglu
Faculty of Engineering and Natural Sciences, Sabanci University
34956 Tuzla, Istanbul, Turkey
{hasanazgin, serkanyaliman, hamzaoglu}@sabanciuniv.edu

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. Alternating Projections (AP) is one of the highest quality image demosaicing algorithms, and it has very high computational complexity. Therefore, in this paper, a high performance AP image demosaicing hardware is proposed. This is the first AP image demosaicing hardware in the literature. The proposed hardware is implemented using Verilog HDL. The Verilog RTL code is verified to work correctly in a Xilinx Virtex 6 FPGA. The FPGA implementation can process 31 full HD (1920x1080) images per second.**

Keywords— Image Demosaicing; Alternating Projections; Hardware Implementation; FPGA.

I. INTRODUCTION

Capturing three color channels (red (R), green (G), and blue (B)) per pixel increases the cost of digital cameras. Therefore, 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. There are several CFA patterns. Bayer pattern 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.

CFA interpolation, commonly 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. There are many image demosaicing algorithms with varying reconstructed image quality and computational complexity. Simple demosaicing algorithms such as bilinear interpolation produce low quality reconstructed images, often accompanied with zipper-effect artifacts around edges. Therefore, complex demosaicing algorithms such as Effective Color Interpolation (ECI) and Alternating Projections (AP) [2, 3, 4] are proposed. These algorithms use the correlation between color channels of the neighboring pixels. They produce higher quality reconstructed images at the expense of increased computational complexity.

TABLE I. COMPUTATIONAL COMPLEXITY COMPARISON

Method	Addition/ Subtraction	Absolute Value	Shift	Multiplica tion
Bilinear Interpolation	2.5 MN		MN	
ECI	10 MN		4 MN	
AP (3 Iterations)	391.5 MN	2 MN	3.5 MN	384 MN
AP (5 Iterations)	583.5 MN	2 MN	3.5 MN	576 MN

AP algorithm is one of the highest quality image demosaicing algorithms, and it has very high computational complexity. Computational complexity comparison for an MxN CFA (mosaic) image and PSNR (dB) comparison for the 24 images in the Kodak image suit [5] of several image demosaicing algorithms are shown in Tables I and II, respectively. Therefore, in this paper, a high performance AP image demosaicing hardware is proposed. This is the first AP image demosaicing hardware in the literature. It uses parallelism and pipelining to achieve real-time performance.

The proposed hardware is implemented using Verilog HDL. The Verilog RTL code is verified to work correctly in a Xilinx Virtex 6 FPGA. The FPGA implementation works at 83 MHz and it 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.

There are several lower quality image demosaicing hardware in the literature [6, 7, 8]. The proposed AP image demosaicing hardware produces much higher quality reconstructed images than them at the expense of increased computational complexity.

II. ALTERNATING PROJECTIONS ALGORITHM

AP algorithm uses the correlation between color (R, G, B) channels of the neighboring pixels for producing high quality reconstructed images. 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.

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.

III. 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 [2], 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 [2]. 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 used in the proposed AP hardware for the 24 images in the Kodak image suit [5] are shown in Table II. These results show that the modifications have negligible impact on the performance of the AP algorithm.

TABLE II. PSNR (dB) Performances of Demosaicing Algorithms

	Bilinear	ECI	AP (5 Iter)	AP HW (5 Iter)
Average Red	29.15	36.47	38.20	38.12
Average Green	33.03	39.05	41.70	41.62
Average Blue	29.49	36.61	38.70	38.66

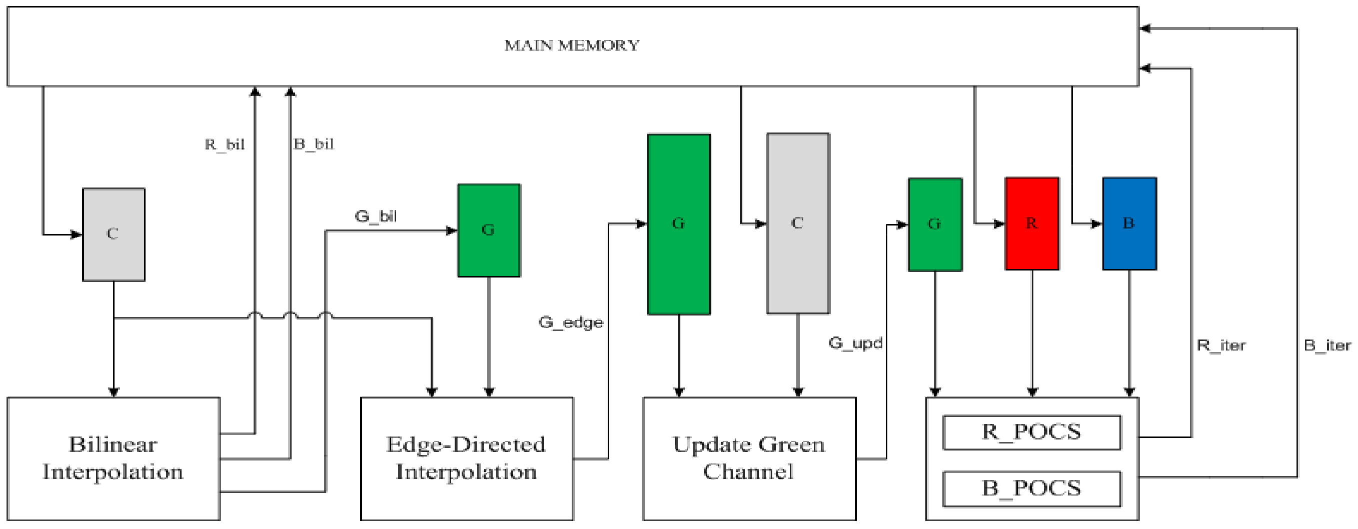

Fig. 1. Proposed Alternating Projections Demosaicing Hardware

TABLE III. 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

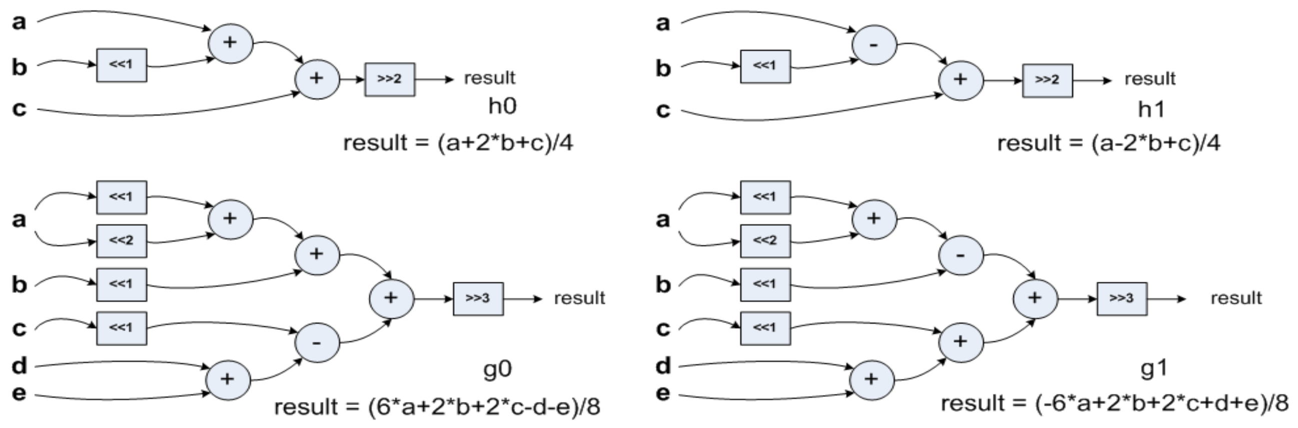

Fig. 3. Forward and Inverse Filter Hardware

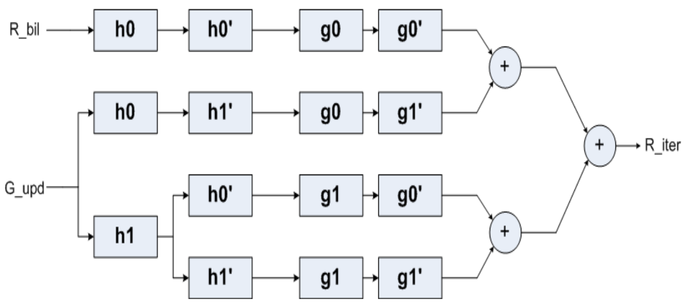

Fig. 2. POCS for Red Channel (R_POCS) Hardware

The proposed AP hardware is shown in Fig. 1. 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 Fig. 2. 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 Fig. 2. 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 Fig. 2. 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 Fig. 3. Edge-directed Interpolation hardware is shown in Fig. 4.

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 Fig. 1 are drawn to scale. In the FPGA implementation, Block RAMs (BRAMs) are used as on-chip memories. As shown in Table III for an

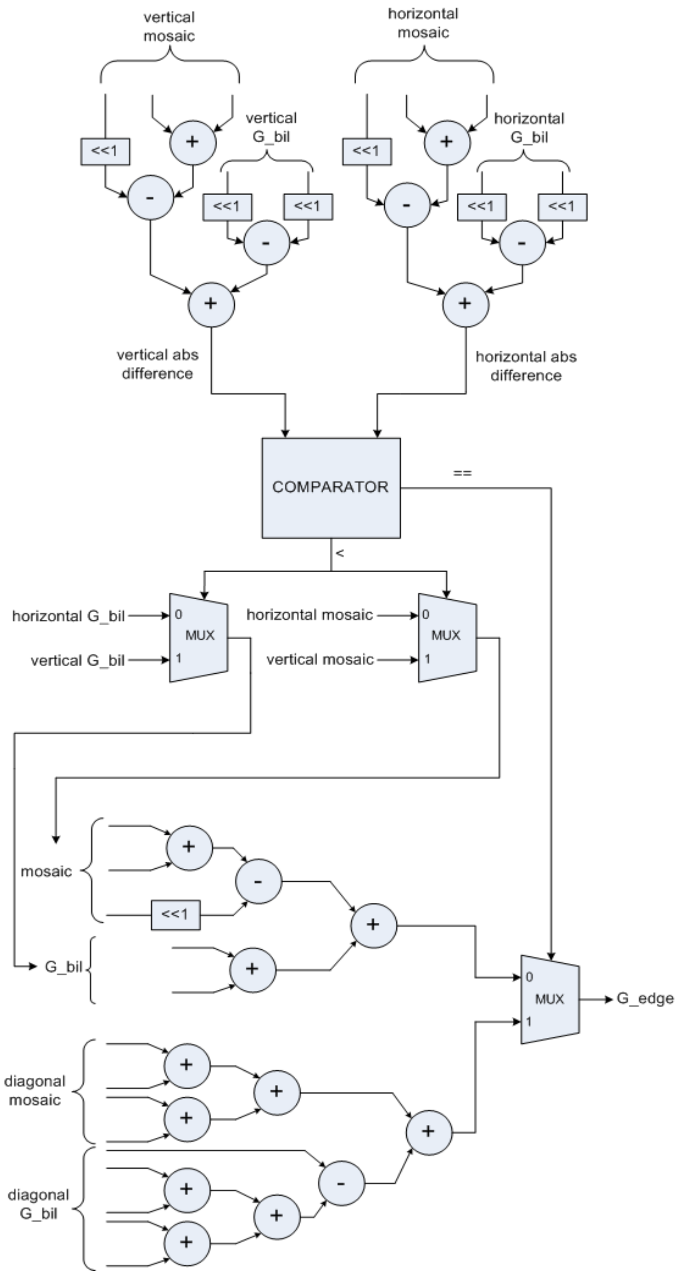

Fig. 4. Edge-Directed Interpolation Hardware

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.

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.

References

[1] B. Bayer, "Color Imaging Array," *U.S. Patent No. 3971065*, 1976.

[2] 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.

[3] 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.

[4] 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.

[5] Eastman Kodak Company, Kodak Lossless True Color Image Suite, http://r0k.us/graphics/kodak/.

[6] S.-C. Hsia, M.-H. Chen, P.-S. Tsai, "VLSI implementation of low-power high-quality color interpolation processor for CCD camera," *IEEE Transactions on Very Large Scale Integration Systems*, Apr. 2006.

[7] I. O. H. Fuentes, M. E. Bravo-Zanoguera, G. G. Yanez, "FPGA Implementation of the Bilinear Interpolation Algorithm for Image Demosaicking," *International Conf. on Electrical, Communications, and Computers*, Feb. 2009.

[8] A. Karloff, R. Muscedere, "A low-cost, real-time, hardware-based image demosaicking algorithm," *IEEE International Conf. on Electro / Information Technology*, June 2009.