An Efficient Implementation of the Adams-Hamilton's Demosaicing Algorithm in FPGAs

Jalal Khalifat, Ali Ebrahim, and Tughrul Arslan

{J.Khalifat,A.ebrahim,T.Arslan}@ed.ac.uk

Abstract. Demosaicing is the process of reconstructing a full color image from incomplete samples generated by typical image sensors. This paper discusses the Adams-Hamilton's demosaicing algorithm and presents a high-performance and cost-effective implementation of the algorithm in Field Programmable Gate Arrays (FPGAs). The paper also presents a proposed demosaicing hardware architecture which increases the number of pixels processed in a single clock cycle by using efficient pipelining. Images obtained from our FPGA implementation are compared to images obtained from standard software demosaicing functions. Our proposed hardware *architecture is shown to outperform previous hardware implementations of the algorithm. Our architecture is capable of processing up to 419 MPixels/s.*

Keywords: Adams-Hamilton, FPGAs, Demosaicing, PSNR , HD, Bayer CFA.

1 Introduction

Most of the commercial portable devices, which capture digital images such as digital cameras and mobile phones, use an array of filters on top of the image sensors. In the case of RGB images format, each pixel record is produced by a sensor overlaid with one type of color filter. This arrangement results in an incomplete image samples with two missing colors from each pixel. Using Color Filter Array (CFA), sensors produce a two-dimensional array of pixels each representing a single color: red, green or blue. Many types of color filter arrays are used in digital capturing devices. The most common type is the Bayer color filter array [1]. The Red-Green-Green-Blue (RGGB) arrangement of the Bayer color filter array is shown in Fig.1a). In this arrangement, the filters are 50% green, 25% red and 25% blue. Half of the sensors are coupled with filters for the green color, the color for which the human eye is more sensitive.

To construct a full RGB image, demosaicing algorithms are used to interpolate missing red, green and blue sub-pixels from the surrounding pixels. Different algorithms have different image qualities and vary in term of computation demand. Bilinear interpolation is the simplest algorithm that interpolates missing colors using symmetric bilinear interpolation from the nearest neighbors of the same color by finding the average color from two or four of the matched surrounding sub-pixels.

A huge number of Demosaicing algorithms have been patented. Some of them, such as the Freeman's algorithm [2], use a median filter added to the bilinear interpolation to

D. Goehringer et al. (Eds.): ARC 2014, LNCS 8405, pp. 205–212, 2014.

reduce the noise and artifacts produce by bilinear interpolation. Other algorithms exploit the spatial correlation principle such as Adams-Hamilton's algorithm demonstrated in [3], and the asymmetric interpolation scheme using color discontinuity equalization demonstrated in [4].

Most of the demosaicing algorithms are implemented and tested using software-based environments. This type of implementation causes a huge reduction in the performance as they execute instructions sequentially. On the other hand, hardware implementation could process the algorithm faster as they use parallel computing concept. Few research works discuss the implementation of demosaicing algorithm in hardware-based environments, but most of them use the bilinear interpolation [5][6] and [8], which adds zipper effects and artifacts to the images produced. Other implementations use customized algorithms such as the algorithm in [9]. In [5] the algorithm was verified using Altera's Cyclone II FPGA. In [6], the implementation uses the same algorithm with HD images. However, no details are given on how data are passed to/from memory. In [7], the author implemented the Freeman's algorithm using a dual-core architecture with coarse-grained dynamically reconfigurable processors that provide throughput of up to 241Mpixel/s.

This paper presents a novel hardware implementation of Adams-Hamilton's algorithm which utilizes efficient pipelining in Field Programmable Gate Arrays (FPGAs). The design of the different components of the data path is discussed and analyzed. This paper also presents a brief introduction of the algorithm and discusses its benefits compared to other algorithms. Moreover, implementation results are compared to other similar designs.

The rest of the paper is organized as follows. In Section 2, we explain the background of Adams-Hamilton's algorithm. In Section 3, we show the architecture of the system and design components. In Section 4, we show the experimental results. Finally, Section 5 concludes the paper.

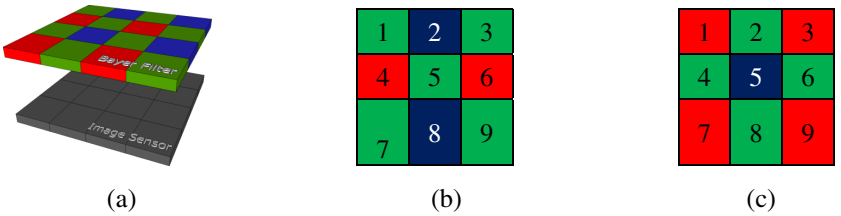
2 Adams-Hamilton's Demosaicing

Adams-Hamilton's algorithm is considered one of the edge-based algorithms that exploit the spatial correlation principle by interpolating along the edges and not across them. This technique reduces color artifacts and zipper effects to the regions with edges not like the other type of algorithms which disregard directional information. Moreover, averaging the pixels across an edge will decrease the sharpness at edges.

The algorithm is divided into two steps; firstly, the green color plane is interpolated, and then the red and blue planes are interpolated. G missing pixels can be interpolated vertically, horizontally or using the two directions based on specific classifiers to choose the interpolation direction as depicted in Fig. 1b) and Fig. 1c). In the case of G pixels in B positions, the same equations are used as R positions replaced with B.

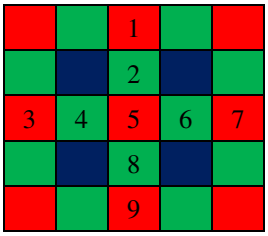
Once the G color plane is interpolated, the algorithm starts interpolating the red and blue colors. In this estimation, a window of 3x3 needed as depicted in Fig. 1b and1c. This step is categorized into three different cases: Case 1 is when the nearest neighbors to (R or B) are in the same column. Case 2 is when the nearest neighbors to

(R or B) are in the same row and case 3 is when the nearest neighbors to the (R or B) are at the corners. In cases 1 and 2, the missing (R or B)s are in G locations as shown in Fig. 1b). The classifiers used to estimate the missing colors are as follow: Equation 1 for nearest neighbor in the same column and Equation 2 in the same row.



(a) (b) (c)

Fig. 1. (a) Bayer Filters and sensors (a) pixel estimation case 1 and 2 (b) pixel estimation case 3



1. Calculate horizontal gradient
$\Delta H = |G4 - G6| + |R5 - R3 + R5 - R7|$
2. Calculate vertical gradient
$\Delta V = |G2 - G8| + |R5 - R1 + R5 - R9|$
3. If $\Delta H > \Delta V$,
$G5 = (G2 + G8)/2 + (R5 - R1 + R5 - R9)/4$
Else if $\Delta H < \Delta V$,
$G5 = (G4 + G6)/2 + (R5 - R3 + R5 - R7)/4$
Else
$G5 = (G2 + G8 + G4 + G6)/4$
$+ (R5 - R1 + R5 - R9 + R5 - R3 + R5 - R7)/8$

Fig. 2. G pixel estimation at pixel 5 using Adams and Hamilton's method

$$B5= (B2+B8)/2+ (-G2+2G5-G8)/2 \quad (1)$$

$$R5= (R4+R6)/2+ (-G4+2G5-G6)/2 \quad (2)$$

Case 3, when the missing R pixel part in B locations as the case depicted in Fig. 1c) or missing B pixel part in R locations. The nearest neighbors are at the corners of the 3x3 window. Classifiers composed of Laplacian second-order terms for the green data and gradients for the chroma data are used. These classifiers are sensing the high spatial frequency information present in the pixel neighborhood in the negative diagonal (DN) and positive diagonal (DP) directions as shown in Equations (3) and (4) for the case of missing R pixel part in B locations, and based of the classifiers, the direction of interpolation determined as shown in Equations 5, 6, 7.

$$DN = |R1 - R9| + |G5 - G1 + G5 - G9| \quad (3)$$

$$DP = |R3 - R7| + |G5 - G3 + G5 - G7| \quad (4)$$

If DN > DP,

$$R5 = (R3 + R7)/2 + (- G3 + 2G5 - G7)/2 \quad (5)$$

Else if DN < DP,

$$R5 = (R1 + R9)/2 + (-G1 + 2G5 - G9)/2 \quad (6)$$

Else

$$R5 = (R1 + R3 + R7 + R9)/4 + (-G1 - G3 + 4G5 - G7 - G9)/4 \quad (7)$$

3 Hardware Implementation

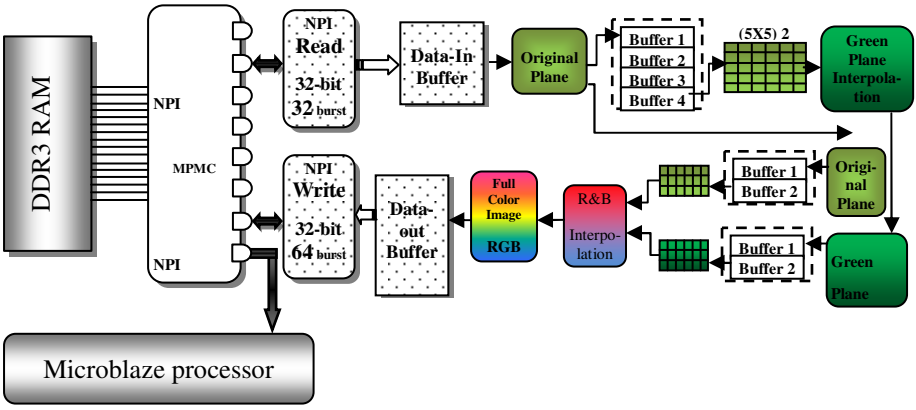


Fig. 3. System implementation

Our implementation of the Adams-Hamilton's demosaicing is illustrated in Fig. 3. As the images processed line by line from left to right, a set of data buffers employed for each line. The implemented system designed to process images of size 1920x1080 pixels where 1920 is the width. A shift window of sizes 5x5 and 3x3 are used in G interpolation and R&B colors interpolation stages respectively.

The system designed to process two pixels in one clock cycle using an efficient pipelining scheme, as the data input is designed to accept 2 bytes (pixels) and the processes are parallelized in the data path. This technique will double the performance but will increase the resources required. Each input pixel (byte) is converted to 3 bytes which limits the number of pixels processed in one clock cycle.

3.1 Data Buffering

The size of the FIFO is 1914 bytes as the image width is 1920 and the remaining six bytes are located in the shift registers where the window is located. Block memories (BRAMs) are used to design FIFOs for each line in the shift window. For the green interpolation stage, Fig. 4 shows the four FIFOs contain the image lines under process and the registers where the actual window is located. The red and blue interpolation uses the same structure but uses two FIFOs instead of four. P22 and P23 are the

locations of pixels processed at each clock cycle. In each clock cycle, a shift operation occurs by shifting the bytes in registers 1 and 2 to the next row buffer and bytes in registers 3 and 4 to registers 1 and 2 and finally the most significant two bytes in the same row buffer to registers 5 and 6. Inside the buffer, if the data of the image is still coming, the write address and read address are incremented when a shift operation occurs; otherwise, the read address is incremented only until the two addresses are equal, which means the end of the image data.

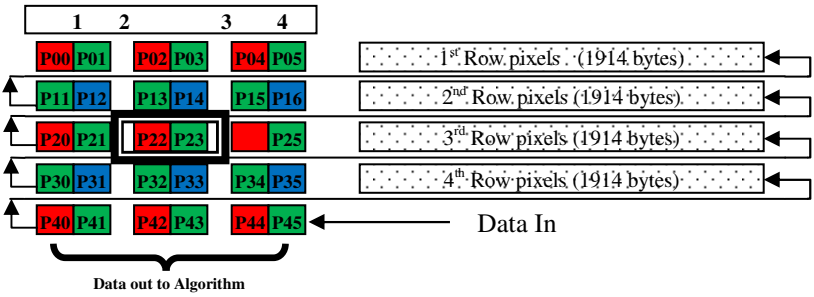


Fig. 4. Buffers of G interpolation stage

3.2 Image Interpolation

In each clock cycle, the content of registers 1 to 6 is sent to the hardware design of the equations mentioned in section 2, for both sets of equations, the G interpolation stage and the R & B interpolation stage. The design is divided into two parts:

1. Even rows: In these rows, the received six bytes ordered in the form RGRGRG. The algorithm interpolates the missing colors in the middle locations where the pixels RG are located. For the G interpolation stage, only the first Green pixel is interpolated as the second is contained in the original data. However, for the R & B interpolation stage, the B part is unknown in the first pixel and the B and R parts are unknown in the second pixel.
2. Odd rows: In these rows, the received six bytes are ordered in the form GBGBGB. The design in these rows acts as it acts in the even rows case, but interpolates different colors as the location of each color is changed. In G stage, the green color part is missing in the second pixels in the odd rows. On the other hand, the red and blue color parts are missing in the first pixel while the red color part is missing in the second pixel.

3.3 Memory Interface

The Xilinx Native Port Interface (NPI) [10] is used to connect the system to the DDR memory through the Xilinx Multi Port Memory Controller (MPMC) [10]. Two ports of the MPMC are dedicated, one for reading and the other for writing. Due to the difference in data size between the NPI part and the demosaicing system, two blocks in the middle named Data_In and Data_out are implemented to convert the size of data from the NPI form to the system form and vise versa. A Microblaze processor is used to control data transfer from memory side and number of burst required.

4 Experimental Results

4.1 Performance and Utilization

The above architecture was implemented on Xilinx ML605 boards with a Virtex-6 XCE6VLX240T FPGA chip. The maximum operation frequency is 209.6 MHz but the system was tested at a frequency of 200MHz due to the limitation in memory interface. When the maximum frequency is used, the throughput reaches 419.2Mpixel/s. Table 1 shows the system resource utilization. The BRAMs used in "Data in buffers" and "Data out" components are mainly to buffer the extra data not yet sent to the next stage, due to the difference in data size between the two consecutive components. To solve this problem a high priority given to the writing operation over the reading from memory to make sure that the buffers are not full.

Table 1. System resource's utilizstion

Resources	Data in B.	Ham Alg.	Data out B.	Mblaze & Mem.	Total	Percentage
Slices	101	1003(2.6%)	202	3163	4469	11%
RAMB36s	15	8 (2%)	36	29	88	21%

The latency of the design equals to the time needed for 2940 clock cycles as the data passes through two different set of FIFOs to process the data. The first FIFO needs around 1940 clock cycles and the second 960 clock cycles. The other logics need around 40 clock cycles. The time measured to process the whole image of size 1920x1080 is equal to 8.53 ms including the memory interface timing. This system can process around 117 frames per second. The design can perform better than the designs proposed in [6][7] in terms of execution time and average throughput as shown in Table. 2 for images of size 1920x1080.

Table 2. Demosaicing performance comparison

Resources	Multi-core DR Freeman [7]	FPGA bilinear implementation [6]	Bilinear Inter. Architecture for Vision Systems [11]	Our Implementation
Execution time (ms)	8.92	13.82	-	8.53
Throughput (Mpix./s)	232.2	150	250	419.2

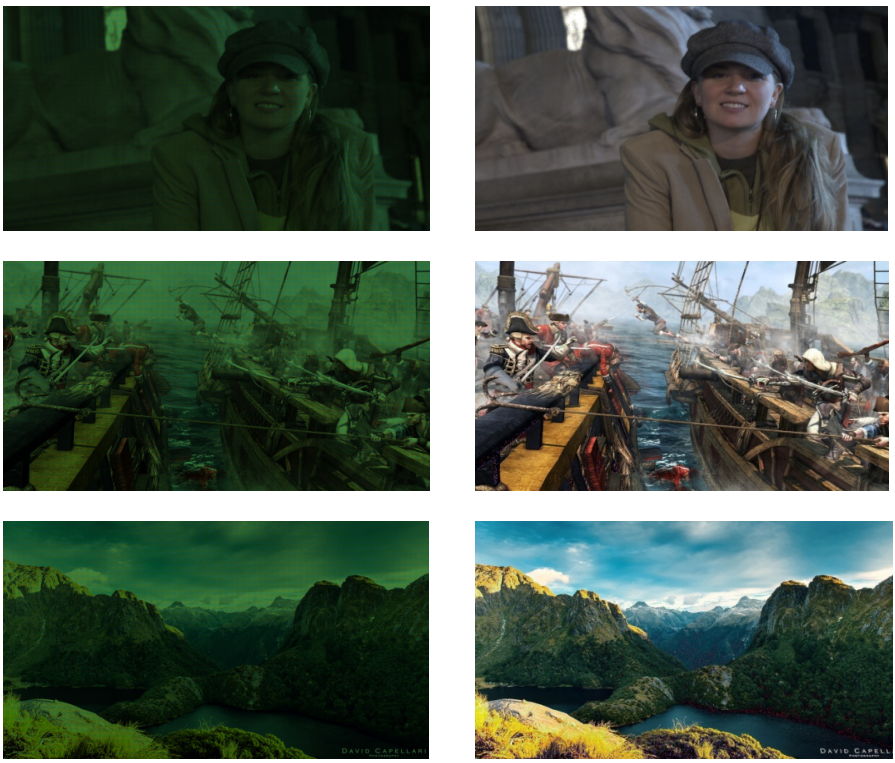
4.2 Image Analysis

Fig. 5b) shows the constructed images using the implemented system, which are originally in raw format as shown in Fig. 5a). To measure the quality of constructed images, a set of images are modified and mosaiced by removing two colors from each pixel to construct images as the one produced by the digital camera's sensors. The position of the colors determines the type Bayer pattern. RGGB Bayer patterned images are produced and then passed to the implemented system stages. The reconstructed images are compared with the original RGB images by taking the Peak Signal to Noise Ratio (PSNR) between the two images.

PSNR is used to measure how closely the constructed image fits the original one. In our case, a set of three images are mosaiced using Matlab. The three Bayer patterned images are passed to the implemented system and PSNR values are calculated for the constructed images to show the algorithm quality. Table 3 shows both the PSNR values for the images using the implemented system and for the same images reconstructed using the function domesaic from Matlab's image processing toolboxes. The results show that our system can achieve similar quality to the one achieved by the gradient-corrected linear interpolation algorithm used in the default demosaic function in Matlab.

Table 3. PSNR for different images vs Matlab

Image	PSNR of the implemented system images (dB)	PSNR of images using demosaic's Matlab function (dB)
Image1 (Woman)	34.214	34.562
Image2 (Battle)	41.931	36.361
Image3 (Green mountains)	31.254	32.458



(a) (b)

Fig. 5. (a) Raw Bayer images; (b) Constructed images

5 Conclusion

In this paper, we have presented an Adams-Hamilton demosaicing implementation based on FPGAs. The implementation is highly cost effective and real time approach to image demosaicing as the system can process images of size 1920x1080 within 9ms. The results have demonstrated that the architecture provides higher throughput than similar designs implemented in hardware. The system used number of data buffers to buffer the unprocessed data prior the processing units and to form processing windows of size 3x3 and 5x5. We have used the Xilinx NPI with MPMC cores to interface the system to the DDR3 RAM memory. The architecture achieved a clock speed of 210MHz on a Xilinx Virtex-6. We have also presented image analysis on number of images produced by the system and compare them with the original RGB images. The measured PSNR values range between 30dB and 42dB.

References

1. Bayer, B.E.: Color imaging array. U.S. Patent No. 3,971,065 (July 1976)
2. Freeman, W.T.: Median filter for reconstruction missing color samples. U.S. Patent No. 4,724,395 (1988)
3. Hamilton, J.F., Adams, J.E.: Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera. U.S. Patent, No. 5629734 (1997)
4. Nguyen, T.: System and method for asymmetrically demosaicing raw data images using color discontinuity equalization. U.S. Patent No. 0,167,602 A1 (2002)
5. Fuentes, I.O.H., Bravo-Zanoguera, M.E., Yanez, G.G.: FPGA Implementation of the Bilinar Interpolation Algorithm for Image Demosaicking. In: International Conference on Electrical, Communications, and Computers (CONIELECOM), pp. 25–28 (2009)
6. Jair, G.L., Miguel, A.A., Julio, W.V.: A Digital Real Time Image Demosaicking Implementation for High Definition Video Cameras. In: Electronics, Robotics and Automotive Mechanics Conference (CERMA), pp. 565–569 (2008)
7. Zhao, X., Yi, Y., Erdogan, A.T., Arslan, T.: Dual-core reconfigurable demosaicing engine for next generation of portable camera systems. In: Conference on Design and Architectures for Signal and Image Processing Conference (DASIP), pp. 289–294 (2010)
8. Rani, K.S., Hans, W.J.: FPGA implementation of bilinear interpolation algorithm for CFA demosaicing. In: International Conference on Communications and Signal Processing (ICCSP), pp. 857–863 (2013)
9. Karloff, A., Muscedere, R.: A low-cost, real-time, hardware-based image demosaicking algorithm. In: IEEE International Conference on Electro/Information Technology (EIT), pp. 146–150 (2009)
10. DS643: LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a), Xilinx Inc. (March 2011)
11. Fahmy, S.A.: Generalised Parallel Bilinear Interpolation Architecture for Vision Systems. In: International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 331–336 (2008)