Advanced Bayer Demosaicing on FPGAs

Donald Bailey
School of Engineering and Advanced Technology,
Massey University
Palmerston North, New Zealand
D.G.Bailey@massey.ac.nz

Sharmil Randhawa, Jim S. Jimmy Li
School of Computer Science, Engineering and Mathematics,
Flinders University
Adelaide, Australia
sherry.randhawa@flinders.edu.au, jimmy.li@flinders.edu.au

***Abstract*—Demosaicing is the process of interpolating the output from a single chip colour filter array sensor to form a full colour image. In hardware, the simplest algorithms: zero order hold and bilinear interpolation, are commonly used because of their simplicity and low resource requirements. State of the art algorithms are difficult to implement in hardware because of their complex access patterns. This paper explores the streamed implementation of a higher order interpolation filter, with a weighted median classifier. Although this comes at a cost of a factor of 10 increase in hardware resources, and a reduction in maximum pixel clock frequency by 30%, this state of the art algorithm gives considerably improved images of 11.2 dB in peak signal to noise ratio with a considerable reduction in interpolation artifacts. For real-time applications where image quality is critical, an implementation of such an advanced demosaicing algorithm on FPGA is essential.**

Keywords—colour filter array; Bayer pattern; demosaicing; stream processing; higher order interpolation

I. Introduction

Most modern colour image sensors use a single chip with a colour filter array to capture the intensity value of a single primary colour for each pixel. The most common filter is the Bayer pattern [1] within which 50% of the pixels are green, 25% are red, and 25% are blue, as shown in Fig. 1. Demosaicing is the process to interpolate the other missing colour pixel values [2] with common artifacts such as blurring, colour bleeding, and zipper effect.

The simplest approach is zero order hold, but this results in significant colour bleeding. The next simplest approach is bilinear interpolation which will tend to blur sharp edges, and results in some colour bleeding due to the averaging process. It may also result in the zipper effect along sharp edges as shown in Fig. 7. A higher order filter will produce improved results, reducing both colour bleeding and the zipper effect, but it is still prone to blurring, particularly of fine details.

One state of the art algorithm, the higher order interpolation with classification of Li and Randhawa [3] will be explored in this paper. It uses several techniques to reduce the artifacts and improve the reconstruction accuracy. Rather than interpolate a single value, four estimates are derived for each missing component by interpolating from four different directions in order to avoid interpolation across an edge. A classifier is then used to produce an output from the four estimates.

FPGAs are increasingly prevalent for embedded image processing because the parallel nature of hardware implementation makes it possible to perform regular processing on a large volume of data without the very high clock speeds that would be required of a software platform [4]. When processing colour images, it is useful to integrate the Bayer interpolation within the image processing pipeline implemented on the FPGA.

This paper explores the implementation of a state of the art demosaicing algorithm on an FPGA. The remainder of this paper is structured as follows. Section II reviews the Li and Randhawa higher order interpolation and classification algorithm [3], followed by an outline of its realisation on FPGA. Several variations of the algorithm were considered to reduce resource requirements with an analysis on the impact of accuracy. Section III compares these variations and section IV gives the discussion and conclusions of the paper.

II. Proposed Bayer Filter Realisations on FPGAs

There have been very few publications on the implementation of Bayer interpolation on FPGAs [5-8]. Much of the literature on hardware implementation simply refers to using bilinear interpolation for demosaicing, with relatively few papers describing any implementation details [9-14]. An FPGA implementation to reduce blurring for contrast enhancement is described in [4], and is then optimised to reduce the hardware requirements. Gunturk's alternating projection algorithm [15] was realised on an FPGA [16]. However, it is not quite as accurate as the algorithm presented in this paper [3].

One constraint for stream processing is that the pixels must be processed in the order that they are streamed from the sensor. This requires pixels to be cached so that the multiple pixel values required for the filter function are available when required. Such caching is implemented using row buffers constructed from on-chip RAM blocks. The other constraint is that a throughput of processing one pixel per clock cycle must be maintained. For complex processing, a pipelined architecture is used to enable the processing for each pixel to be distributed over many clock cycles resulting in increased latency.

The algorithm consists of three stages [3]. The missing green pixels are first estimated due to being more densely sampled. The red samples are then interpolated for the blue pixels, and the blue samples for the red input pixels by interpolating along the diagonals. Finally, the remaining red and blue samples are interpolated horizontally and vertically. For each stage, four estimates are derived for each pixel, from four different directions, and a classifier is used to select one of these as the output value. The architecture is shown in Fig. 2, and described in the following subsections.

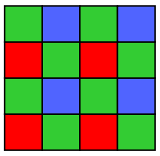
Fig. 1. Bayer Pattern

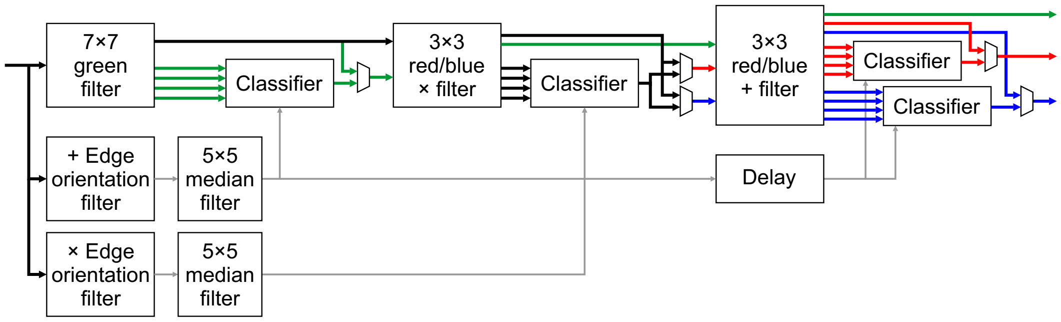

Fig. 2. Block diagram of the realisation of the higher order interpolation with classification.

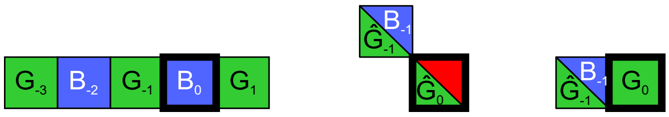

Fig. 3. Interpolation contexts for the three stages (only one direction shown).

A. *Green pixels*

Referring to Fig. 3, a higher order interpolation (HOI) is used to calculate the missing green values based on the hue assumption [3]. Consider the pixel labelled B0: The green value on the blue rows may be estimated from the left as follows:

$$\hat{G}_0^L = \tfrac{3}{4}G_{-1} + \tfrac{1}{8}\left(G_1 + G_{-3}\right) + \tfrac{1}{2}\left(B_0 - B_{-2}\right) \tag{1}$$

with the resulting sum clipped to prevent overflow. Other estimates are similarly made from the top, right and bottom, requiring a 7×7 window to obtain all four estimates (G^L, G^T, G^R, G^B). A similar process is applied for estimating the green pixels on the red rows.

Fig. 4 shows a more detailed view of the architecture of the first stage filter. The 7×7 window is formed using 6 row buffers to cache previous rows. Image borders are managed using 2-phase duplication, duplicating the nearest pixel of the correct phase from alternating rows and columns within the image. This is accomplished using multiplexers (not shown in Fig. 4).

Each HOI block performs higher order interpolation from the four directions using (1). The delayed image output for the following stage is taken 2 pixels after the centre of the window to account for the latency of the HOI calculation, and the classifier. In parallel with this, an edge orientation map is formed to assist with the selection by the classifier. The edge orientation filter estimates whether the local gradient is predominantly horizontal or vertical:

$$E_+ = \begin{cases} 1 & \left|I_{+x} - I_{-x}\right| > \left|I_{+y} - I_{-y}\right| \\ 0 & otherwise \end{cases} \tag{2}$$

where *I* is the input component. The edge orientation map is then filtered to remove outliers. A 5×5 separable median filter is used in this paper, instead of a 7×7 standard median filter [3], to reduce the number of row buffers. The E_+ output is formed using the 7×7 window, as shown in Fig. 4. The sub-window offset by 2 rows and 2 columns accounts for the difference in latency from the median filter. Since the edge map is binary, the median filter can be efficiently implemented by counting pixels as illustrated in Fig. 5. Image borders are again managed using pixel duplication in forming the window.

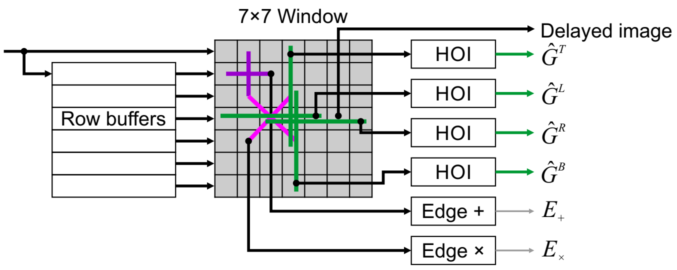

Fig. 4. Architecture of the 7×7 stage 1 filter.

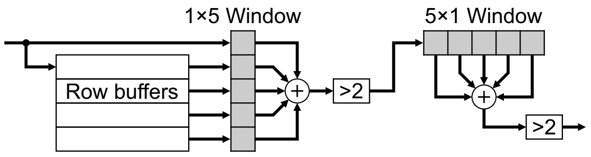

Fig. 5. Architecture of the 5×5 separable median filter.

A weighted median classifier is used to select an output from the four estimates. The estimates parallel to the edge (using the filtered edge orientation map) are given a weight of 2, while those across the edge are given a weight of 1.

$$\hat{G} = \begin{cases} \text{median}\left(\hat{G}^L, \hat{G}^T, \hat{G}^T, \hat{G}^R, \hat{G}^B, \hat{G}^B\right) & \text{when } E = 1 \\ \text{median}\left(\hat{G}^L, \hat{G}^L, \hat{G}^T, \hat{G}^R, \hat{G}^R, \hat{G}^B\right) & \text{when } E = 0 \end{cases} \tag{3}$$

The architecture of the classifier is shown in Fig. 6. A simple bubble sort network is used to sort the four estimates. Rather than duplicate the elements before sorting to determine the median, each of the inputs is augmented with the median filtered edge orientation. This reduces the number of compare and swap blocks from 15 (for sorting 6 values) to 6 (for sorting 4 values). The weights are then used to select the appropriate output.

B. *Red and blue pixels*

The second stage uses a simple first order interpolation to estimate the blue samples for red input and red samples for blue inputs. This requires interpolating diagonally. Referring to Fig. 3, an estimate of the blue pixel value for a red raw pixel from the top left direction is

$$\hat{B}_0^{TL} = B_{-1} + \left(B_0 - B_{-1}\right) \approx B_{-1} + \left(\hat{G}_0 - \hat{G}_{-1}\right) \tag{4}$$

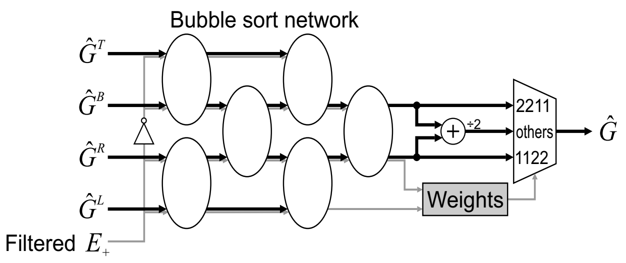

Fig. 6. Architecture of the weighted median classifier.

The other three diagonal directions are similarly interpolated (as are the red pixel values at the blue locations). The raw and green input streams are delayed for synchronisation for the final filter stage. A diagonal edge orientation map is formed in a similar manner to that in (2) for the green pixels. To reduce the logic, the diagonal edge map is calculated using pixels from the original 7×7 window in stage 1, producing $E_\times$ as shown in Fig. 4. The pixels used are offset to account for the latencies of the intervening parallel stages.

The final stage is to interpolate the red and blue channels horizontally and vertically to give outputs for the green pixels. Estimating, the blue pixels from the left gives:

$$\hat{B}_0^L = B_{-1} + \left(B_0 - B_{-1}\right) \approx B_{-1} + \left(G_0 - \hat{G}_{-1}\right) \tag{5}$$

and similarly for the red pixels. The filtered edge orientation map from (2) is delayed to synchronise it with the four estimates for classification.

C. *Variations considered*

Three variations on the above demosaicing algorithm are considered.

1) First order green interpolation: Replacing the second order interpolation of (1) with a first order interpolation reduces the initial stage to a 5×5 window:

$$\hat{G}_0^L = G_{-1} + \tfrac{1}{2}\left(B_0 - B_{-2}\right) \tag{6}$$

2) Not filtering the edge map: The algorithm may be simplified by not filtering the edge map. The edge orientation calculated by (2) is used directly by the classifier.

3) Mean classifier: The weighted median in (3) is quite complex to implement, requiring quite a lot of logic to implement the sorting. Replacing this with the mean of the two estimates along the edge would simplify this:

$$\hat{G} = \begin{cases} \tfrac{1}{2}\left(\hat{G}^T + \hat{G}^B\right) & \text{when } E = 1 \\ \tfrac{1}{2}\left(\hat{G}^L + \hat{G}^R\right) & \text{when } E = 0 \end{cases} \tag{7}$$

III. Results and Performance

The algorithm was implemented on an Altera DE2-115 board, which is based on a low cost Cyclone IV FPGA. Images with 800×600 resolution were captured from a D5M camera and displayed onto a monitor. A pixel size of 8 bits (0 to 255) was used throughout the design. The display had a 40 MHz pixel clock rate, which was the timing constraint used for synthesis. The design was synthesised using Quartus Web edition, version 14.1 using VHDL to represent the design.

The lighthouse image from the Kodak image set is commonly used for objectively evaluating the performance of demosaicing algorithms. ModelSim-Altera was used to simulate the behaviour on the image and to calculate the mean square error (MSE) and peak signal to noise ratio (PSNR). The reconstructed images were also visually assessed for quality.

The synthesis results, and reconstruction accuracy for the proposed method are summarised in Table I with samples of the image showing the picket fence compared in Fig. 7.

TABLE I. Comparison of different algorithms and variations (all were implemented by the authors)

Method	**LUTs**	**Flip-flops**	**M9K RAMs**	**f_{max} (MHz)**	**MSE**	**PSNR (dB)**
Nearest neighbour	159	144	2	96.7	223.2	24.6
Bilinear	221	148	2	97.3	97.20	28.3
Proposed method	**2522**	**1448**	**19**	**67.8**	**7.22**	**39.5**
First order for green	2338	1376	18	64.8	8.35	38.9
No median for edge map	2359	1244	16	61.4	7.92	39.1
Mean classifier	2032	1448	19	82.1	10.00	38.1
All 3 variations	1536	1072	14	79.8	15.20	36.3

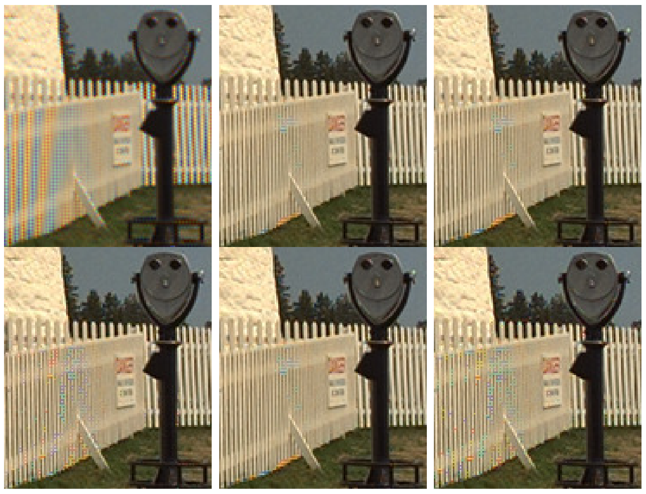

Fig. 7. Samples for visual quality assessment. Top row: bilinear interpolation; proposed method; first order for green. Bottom row: unfiltered edge map; mean classifier; all 3 variations.

As expected, zero order hold uses the fewest resources, but has the lowest quality. Bilinear interpolation only uses a few more resources with some improvement in quality but still contains significant colour artifacts. Our proposed method gives significantly better quality at the cost of around 10 times increase in hardware resources due to larger filters, multiple stages, and more complex selection method. The maximum clock frequency is also 30% lower, as a result of the propagation delay through the classifier. Since this was faster than the target clock frequency, this was not considered a problem. If necessary, the clock frequency could be increased by adding an additional layer of pipelining within the classifier.

Of the variations, using a first order interpolation for green gives a small increase in reconstruction error but visually is almost unnoticeable. Whilst removing the edge map filtering has only a minor effect numerically, visual artifacts have been considerably increased as shown in Fig. 7, particularly in terms of random colour bleeding on the picket fence. To include the recommended median filter in [3], this would require more row buffers to extend the 7×7 window for the first stage to account for the increased latency of a larger median filter. Switching to the mean classifier has the biggest reduction in logic resources, with only a modest decrease in numerical quality, while showing little visual quality differences between the mean classifier and weighted median classifier.

The resources required by the proposed filter are broken down into that required by the constituent component blocks in Table II. Window formation includes the multiplexers for managing image borders, and the set of registers required to make all of the pixels within the window available in parallel. The stage 2 and 3 filters require wider row buffers (16 bits and 24 bits per pixel respectively) to provide all of the data from the previous stage. This accounts for the increased resources, even though the filters are smaller. The median filter required relatively few resources. However, the weighted median classifier was relatively expensive, given that four are required for the design.

TABLE II. RESOURCE BREAKDOWN FOR THE ADVANCED BAYER INTERPOLATION INPLEMENTATION

	Component	LUTs	Flip-flops	M9K RAMs
	7×7 window	407	378	6
	Filter calculations (incl E_+ and $E_\times$)	318	33	0
	Control signals and timing	68	107	0
Stage 1 filter (green and edges)		**793**	**518**	**6**
	3×3 window	247	236	4
	Filter calculations	124	48	0
	Control signals and timing	56	41	0
Stage 2 filter (red/blue ×)		**427**	**325**	**4**
	3×3 window	303	292	6
	Filter calculations	188	88	0
	Control signals and timing	37	29	0
Stage 3 filter (red/blue +)		**528**	**409**	**6**
	5×5 window	47	41	1
	Filter calculations	6	5	0
Median filter (2 required)		**53**	**46**	**1**
Weighted median classifier (4 reqd)		**147**	**8**	**0**
	Interconnection logic	80	72	1
Total		**2522**	**1448**	**19**

IV. DISCUSSION AND CONCLUSIONS

Reconstruction quality can be improved significantly over simple bilinear interpolation, with a reduction in error by more than 11.2 dB for the test image used. However this does come at a significant cost in terms of resources. The individual component filters are relatively simple to implement, with particular care required to match the latencies of the parallel branches. The weighted median classifier alone increased the logic by 20%. For a small reduction in quality, the simpler mean classifier could be considered.

Further work on the optimisation of the algorithm could include multiplexing two of the classifier blocks over the 4 instances to reduce resources since each of them is used only 50% of the time. As the design met the target VGA clock rate, little effort was made in optimising the timing. Future work is required to improve the timing for higher speed designs through increased pipelining. Our proposed FPGA implementation of a higher order state-of-the-art demosaicing algorithm has been successfully realised, enabling the image quality to be significantly improved for required applications.

REFERENCES

[1] B. E. Bayer, "Color imaging array," *United States of America* patent 3971065, 1976.

[2] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: a systematic survey," in *Visual Communications and Image Processing 2008*, San Jose, California, USA, 2008, p. 15 pages.

[3] J. S. J. Li and S. Randhawa, "Color filter array demosaicking using high-order interpolation techniques with a weighted median filter for sharp color edge preservation," *IEEE Transactions on Image Processing,* vol. 18, no. 9, pp. 1946-1957, 2009.

[4] D. G. Bailey, *Design for embedded image processing on FPGAs*. Singapore: John Wiley and Sons (Asia) Pte. Ltd., 2011.

[5] J. Liu, Y. Pan, W. Ding, and R. Huan, "Cost effective hardware based demosaicking algorithm for embedded system," in *2013 International Conference on Wireless Communications & Signal Processing (WCSP)*, Hangzhou, China, 2013, pp. 1-6.

[6] Q. Zhang and L. Zhao, "Implementation of Bayer image interpolation based on FPGA," in *6th International Conference on New Trends in Information Science and Service Science and Data Mining (ISSDM)*, Taipei, Taiwan, 2012, pp. 45-49.

[7] A. Karloff and R. Muscedere, "A low-cost, real-time, hardware-based image demosaicking algorithm," in *IEEE International Conference on Electro/Information Technology (EIT'09)*, Windsor, Ontario, Canada, 2009, pp. 146-150.

[8] J. Khalifat, A. Ebrahim, and T. Arslan, "An efficient implementation of the Adams-Hamilton's demosaicing algorithm in FPGAs," in *10th International Symposium on Reconfigurable Computing: Architectures, Tools, and Applications (ARC 2014)*, Vilamoura, Portugal, 2014, pp. 205-212.

[9] I. O. H. Fuentes, M. E. Bravo-Zanoguera, and G. G. Yanez, "FPGA implementation of the bilinear interpolation algorithm for image demosaicking," in *International Conference on Electrical, Communications, and Computers (CONIELECOMP 2009)*, Cholula, Puebla, Mexico, 2009, pp. 25-28.

[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

[11] J. Garcia-Lamont, M. Aleman-Arce, and J. Waissman-Vilanova, "A digital real time image demosaicking implementation for high definition video cameras," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA '08)*, Morelos, Mexico, 2008, pp. 565-569.

[12] D. Baumgartner, P. Rossler, and W. Kubinger, "Performance benchmark of DSP and FPGA implementations of low-level vision algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, Minneapolis, Minnesota, USA, 2007, pp. 1-8.

[13] K. S. Rani and W. J. Hans, "FPGA implementation of bilinear interpolation algorithm for CFA demosaicing," in *2013 International Conference on Communications and Signal Processing (ICCSP)*, Melmaruvathur, India, 2013, pp. 857-863.

[14] J. M. Perez, P. Sanchez, and M. Martinez, "Low-cost Bayer to RGB bilinear interpolation with hardware-aware median filter," in *16th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2009)*, Yasmine Hammamet, Tunisia, 2009, pp. 916-919.

[15] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing,* vol. 11, no. 9, pp. 997-1013, 2002.

[16] H. Azgin, S. Yaliman, and I. Hamzaoglu, "A high performance alternating projections image demosaicing hardware," in *24th International Conference on Field Programmable Logic and Applications (FPL)*, Munich, Germany, 2014, p. 4 pages.