Fully Pipelined Low-Cost and High-Quality Color Demosaicking VLSI Design for Real-Time Video Applications

Shih-Lun Chen, *Member, IEEE*, and Huan-Rui Chang

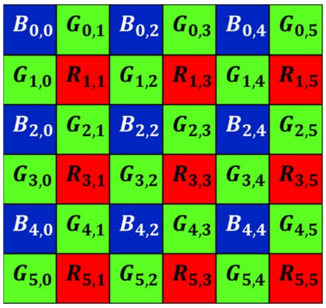
***Abstract*—This brief presents a fully pipelined color demosaicking design. To improve the quality of reconstructed images, a linear deviation compensation scheme was created to increase the correlation between the interpolated and neighboring pixels. Furthermore, immediately interpolated green color pixels are first to be used in hardware-oriented color demosaicking algorithms, which efficiently promoted the quality of the reconstructed image. A boundary detector and a boundary mirror machine were added to improve the quality of pixels located in boundaries. In addition, a hardware sharing technique was used to reduce the hardware costs of three interpolators. The very-large-scale integration architecture in this brief contains only 4.97 K gate counts, and the core area is 60 229 μm^2 synthesized by using 0.18-μm CMOS process. Compared with the previous low-complexity designs, this work has the benefits in terms of low cost, low power consumption, and high performance.**

***Index Terms*—Boundary detection, boundary mirror, color filter array (CFA), digital camera, linear deviation compensation, very-large-scale integration (VLSI).**

I. Introduction

Digital cameras are widely used in many fields. The most well known is portable consumer electronics such as smart phones, tablet PCs, digital videos, digital cameras, and notebooks. A color filter array (CFA) technique is an efficient and compact method to obtain a multispectral image both on CCD and CMOS image sensors.

The most widely used format of CFAs in modern electronic products is Bayer CFA [1], as shown in Fig. 1, in which each pixel contains one of red, green, and blue colors only. Since two-thirds of color information is missed after using CFA, it is necessary to interpolate missing values back to the CFA image to restructure a full color image.



$B_{0,0}$	$G_{0,1}$	$B_{0,2}$	$G_{0,3}$	$B_{0,4}$	$G_{0,5}$
$G_{1,0}$	$R_{1,1}$	$G_{1,2}$	$R_{1,3}$	$G_{1,4}$	$R_{1,5}$
$B_{2,0}$	$G_{2,1}$	$B_{2,2}$	$G_{2,3}$	$B_{2,4}$	$G_{2,5}$
$G_{3,0}$	$R_{3,1}$	$G_{3,2}$	$R_{3,3}$	$G_{3,4}$	$R_{3,5}$
$B_{4,0}$	$G_{4,1}$	$B_{4,2}$	$G_{4,3}$	$B_{4,4}$	$G_{4,5}$
$G_{5,0}$	$R_{5,1}$	$G_{5,2}$	$R_{5,3}$	$G_{5,4}$	$R_{5,5}$

Fig. 1. Block diagram of the proposed color demosaicking algorithm.

Recently, some high-performance and high-quality color demosaicking and interpolation algorithms have been proposed. An orientation-free edge strength filter is proposed in [2]. It utilized edge information to avoid averaging noncorrelated color differences and improved demosaicking performance successfully. A stochastic estimation approach to adaptive interpolation of CFA was proposed in [3]. A gradient edge detection masks and adaptive heterogeneity-projection method for demosaicking the CFA was proposed by Chung *et al.* [4]. In order to avoid artifacts such as zipper effect, blur, and color spots, a self-similarity color demosaicking algorithm [5] was exploited to interpolate missed colors, which can be treated more generally as graph-based regularization to the image. In addition, another novel graph-based regularization framework was presented in [6], in which a weight matrix was built to measure similarity and a static Laplacian was used to solve a variational problem. The artificial effect can be alleviated efficiently. Recently, a voting-based directional demosaicking algorithm has been proposed in [7]. It uses a voting-based edge direction detection method and a directional weighted method to improve the quality of reconstructed images. Although the color demosaicking and interpolation algorithms, as aforementioned, achieved high quality, it is hard to realize these algorithms by using the very-large-scale integration (VLSI) technique for real-time video applications.

Therefore, several hardware-oriented color demosaicking algorithms [8]–[12] were proposed for the VLSI implementation. Hsia *et al.* [8] proposed an edge direction weighting and local gain approach-based color demosaicking algorithm. In this design, the chip area cannot decrease obviously because the dividers and multipliers were used. Hence, a cost-efficient method was proposed in [9] by using only simple operations such as addition, subtraction, and shifting. A low-complexity algorithm based on the edge information and interchannel correlation methods was presented in [10], in which a resource sharing technique and a pipeline scheduling technique were used to reduce hardware cost and improve throughput. A cost-effective and pipelined architecture based on a modified color difference space method by using direction determination was

Manuscript received July 22, 2014; revised November 1, 2014; accepted January 13, 2015. Date of publication February 27, 2015; date of current version May 29, 2015. This work was supported in part by the National Science Council, Taiwan, under Grants NSC-102-2221-E-033-063, NSC-102-2622-E-033-009-CC2, NSC-102-2815-C-033-022-E, MOST-103-2221-E-033-070, MOST-103-2218-E-033-004, MOST-103-2622-E-033-001-CC2, and CYCU-EECS-10301, and in part by the National Chip Implementation Center, Taiwan. This brief was recommended by Associate Editor Y.-B. Kim.

The authors are with the Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan 320, Taiwan (e-mail: chrischen@cycu.edu.tw).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2015.2407773

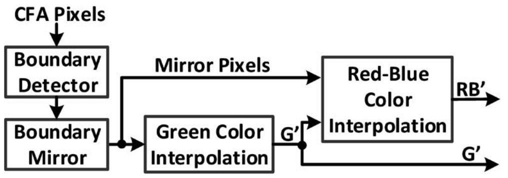


Fig. 2. Block diagram of the proposed color demosaicking algorithm.

proposed in [11]. Recently, an adaptive edge-enhanced method, including an anisotropic weighting model and a filter-based compensator, has been presented in [12].

In this brief, a novel cost-efficient and high-performance hardware-oriented color demosaicking algorithm is proposed for VLSI implementation. Since the demosaicking equations for different colors have some similar parts, they can be integrated into a shared VLSI architecture by using the hardware sharing technique. Furthermore, in order to meet the demand of real-time video applications, the pipeline scheduling technique is used to realize this design.

II. Proposed Color Demosaicking Algorithm

The proposed color demosaicking algorithm consists of a novel linear deviation compensation, immediately interpolated green color information, boundary detection, and boundary mirror models, as illustrated in Fig. 2. The boundary detector and mirror models were added to detect the boundary information and provide mirror pixels for green and red–blue color interpolations, which can efficiently improve the quality of interpolated pixels located in the boundary. In addition, the linear deviation compensation and immediately interpolated green color pixels G' are used to improve the quality of the interpolated red and blue color pixels. The details of green and red–blue color interpolation models are described in the following sections.

A. Green Color Interpolation

When the locations of target pixels in the original CFA image are red and blue colors, the green color pixels should be demosaicked by the green color interpolation model, as shown in (1). Since there is more information in horizontal than vertical directions, in order to avoid the difference between different rows, the result of green color pixels G' captures the CFA pixels in the horizontal direction only. Moreover, it is easy to use the same architecture to be implemented with (4) by the hardware sharing technique. The results of (1) will be stored into the second register bank as immediately interpolated green color pixels for red–blue color interpolation

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

B. Red–Blue Color Interpolation

In contrast with traditional bilinear guided demosaicking method, the immediately interpolated green color pixel G' in (1) is used to improve the quality of red–blue color interpolation. The immediately interpolated pixels are first to be used in the hardware-oriented color demosaicking algorithm. Moreover, a novel linear deviation compensation is also used to promote the performance of red–blue color interpolation. The difference between G' and the average of surrounding green color pixels in the original CFA image is used to compensate the interpolated red and blue pixels. Finally, the quality of the reconstructed red and blue color pixels can be significantly improved by using the immediately green color information and linear deviation compensation techniques. The details of the red–blue color interpolation method are described in the following.

Equation (2) shows the calculation of $RB'^{(BR)G}_{i,j}$ where the original CFA pixel is $B_{i,j}$ or $R_{i,j}$. $R'_{i,j}$ and $B'_{i,j}$ can be calculated by the average of four neighboring pixels (i.e., $RB_{i-1,j-1}$, $RB_{i-1,j+1}$, $RB_{i+1,j-1}$, and $RB_{i+1,j+1}$) and then compensated by green colors. The linear deviation compensation technique uses the difference between $G'_{i,j}$ and eight surrounding original and interpolated green color pixels, as presented in the latter part of (2). The quality of interpolated red and blue pixels can be efficiently improved by the proposed linear deviation compensation technique

$$\begin{aligned}RB'^{(BR)G}_{i,j}&\\ =\frac{1}{4}&(RB_{i-1,j-1}+RB_{i-1,j+1}+RB_{i+1,j-1}+RB_{i+1,j+1})\\ &+G'_{i,j}-\frac{1}{8}\big[(G'_{i-1,j-1}+G'_{i-1,j+1}+G'_{i+1,j-1}+G'_{i+1,j+1})\\ &+(G_{i-1,j}+G_{i,j-1}+G_{i,j+1}G_{i+1,j})\big].\end{aligned} \tag{2}$$

Equation (3) shows the calculation of $RB'^{(G)BR}_{i,j}$ where the original CFA pixel is $G_{i,j}$. Since the color of $RB'_{i,j}$ is the same as that of $RB_{i-1,j}$ and $RB_{i+1,j}$ in the vertical direction, the average of $RB_{i-1,j}$ and $RB_{i+1,j}$ is the best information to interpolate the pixel $RB'_{i,j}$. Moreover, the function of the linear deviation compensation technique used in (3) is the same as that used in (2). Hence, the hardware cost can be greatly reduced by sharing the linear deviation compensation with (2), which can improve the quality of the interpolated pixels both in (2) and (3) by the same linear deviation compensator efficiently

$$\begin{aligned}RB'^{(G)BR}_{i,j}&\\ =\frac{1}{2}&(RB_{i-1,j}+RB_{i+1,j})+G_{i,j}\\ &-\frac{1}{8}\big[(G'_{i-1,j}+G'_{i,j-1}+G'_{i,j+1}+G'_{i+1,j})\\ &+(G_{i-1,j-1}+G_{i-1,j+1}+G_{i+1,j-1}+G_{i+1,j+1})\big].\end{aligned} \tag{3}$$

$RB'^{(G)RB}_{i,j}$ is the result of $RB_{i,j}$ where the original CFA color is $G_{i,j}$, and the colors of $RB_{i,j-1}$ and $RB_{i,j+1}$ are the same as that of $RB'_{i,j}$. In order to achieve high quality and low complexity, the reconstruction process for $RB'^{(G)RB}_{i,j}$ is only capturing the CFA pixels in the horizontal direction. Hence, $R'_{i,j}$ and $B'_{i,j}$ can be obtained by using the CFA pixels in the horizontal direction, as shown in (4) as

$$\begin{aligned}RB'^{(G)RB}_{i,j}&=\frac{1}{2}(RB_{i,j-1}+RB_{i,j+1})+\frac{1}{2}G_{i,j}\\ &\quad-\frac{1}{4}(G_{i,j-2}+G_{i,j+2}).\end{aligned} \tag{4}$$

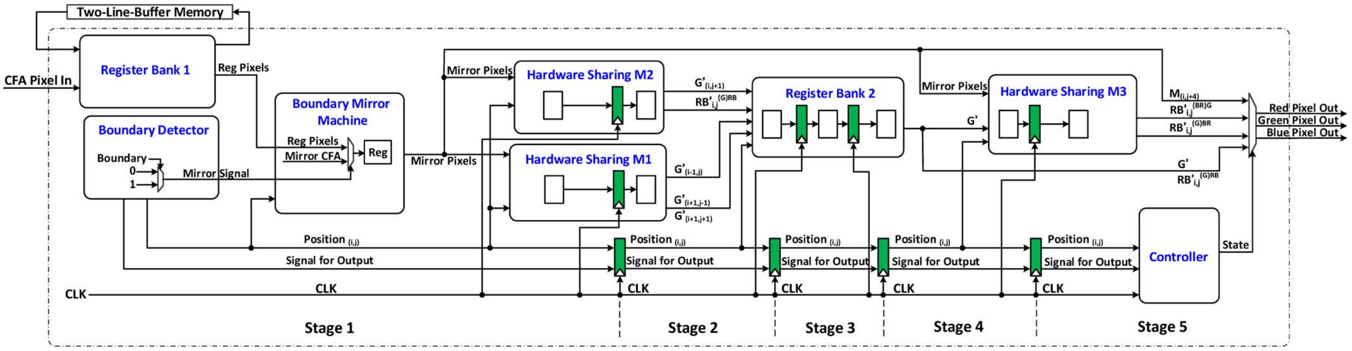


Fig. 3. Block diagram of the VLSI architecture for the proposed color demosaicking design.

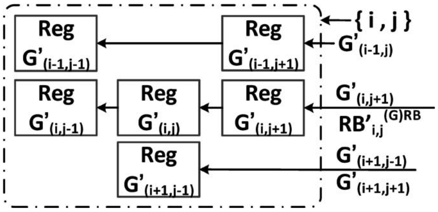


Fig. 4. Architecture of register bank 2.

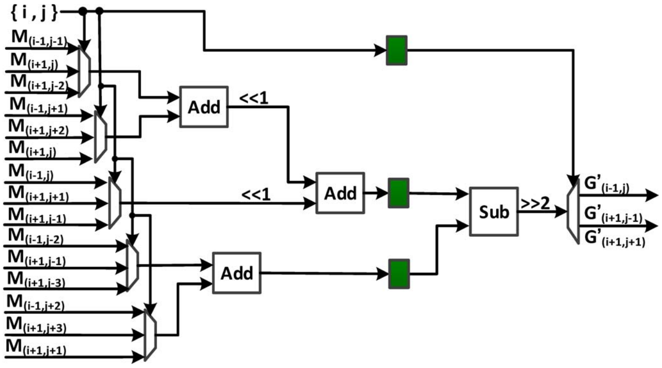


Fig. 5. Architecture of hardware sharing M1.

The eight surrounding values of G' information used in (2) and (3) can be obtained from (1). Since the interpolation equations have the characteristics of high similarity, the hardware sharing technique can be used to reduce hardware cost efficiently.

III. VLSI Architecture

Fig. 3 shows a five-stage pipelined architecture of the proposed color demosaicking algorithm. It consists of register bank 1; a boundary detector; a boundary mirror machine; three hardware sharing M1, M2, and M3; register bank 2; and a controller. All parameters in the proposed algorithm are designed as 2, 4, and 8. Hence, the hardware cost can be greatly reduced by using the shifters rather than dividers and multipliers. The following sections described the details of each block.

A. Register Banks

Register bank 1 is composed of 16 shift registers to provide the 16 original CFA pixels as inputs for the boundary mirror machine. By connecting with a two-line-buffer memory device, as shown in Fig. 3, the proposed color demosaicking design achieves pixel in and pixel out for real-time video applications. Register bank 2 consists of six registers, as shown in Fig. 4. Since the red–blue color interpolation uses the information of immediately interpolated green color pixels G', it is necessary to store the green color pixels of the upper, middle, and lower rows as inputs for hardware sharing M3 to perform the linear deviation compensation. Moreover, register bank 2 was also used to balance the arriving time of $RB'^{(G)RB}_{i,j}$ and $G'_{i,j+1}$ because the result of $RB'^{(G)RB}_{i,j}$ is one cycle later than $G'_{i,j+1}$.

B. Boundary Detector and Boundary Mirror Machine

The boundary detector is designed to provide a flag signal to the boundary mirror machine to determine whether the location of the target pixel is within the boundary region or not. If the location is within the boundary range, the boundary mirror machine will provide the missed CFA values in the boundary region by a mirror technique. Moreover, six extra shift registers were added to store the information of the mirror CFA pixels for hardware sharing M1 and M3.

C. Hardware Sharing M1

Fig. 5 shows the architecture of hardware sharing M1. It can be realized by (1) to obtain the results of $G'_{i-1,j}$, $G'_{i+1,j-1}$, and $G_{i+1,j+1'}$. Hardware sharing M1 consists of three adders, a subtractor, three shifters, and three registers. The three registers were added to develop the pipeline scheduling. In order to save the hardware cost, hardware sharing M1 is shared to interpolate the green color pixels of the upper and lower rows. It is used to obtain the result of $G'_{i-1,j}$ when the color of the target pixel $\mathrm{M}_{\mathrm{i,\,j}}$ in the CFA image is green. Otherwise, when the color of the target pixel $\mathrm{M}_{\mathrm{i,\,j}}$ in the CFA image is red and blue, it is used to calculate the result of $G'_{i+1,j-1}$ and $G'_{i+1,j+1}$, respectively. By adding six multiplexers to select different input signals, hardware sharing M1 can obtain $G'_{i-1,j}$, $G'_{i+1,j-1}$, and $G'_{i+1,j+1}$ successfully. The results of $G'_{i-1,j}$, $G_{i+1,j-1}$, and $G'_{i+1,j+1}$ will be stored in register bank 2 as inputs for hardware sharing M3. By using the hardware sharing technique, the computing resource of the green color interpolator achieves reduction of 50% adders, subtractors, and shifters.

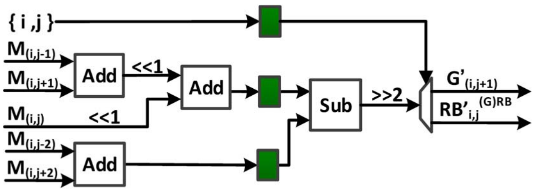


Fig. 6. Architecture of hardware sharing M2.

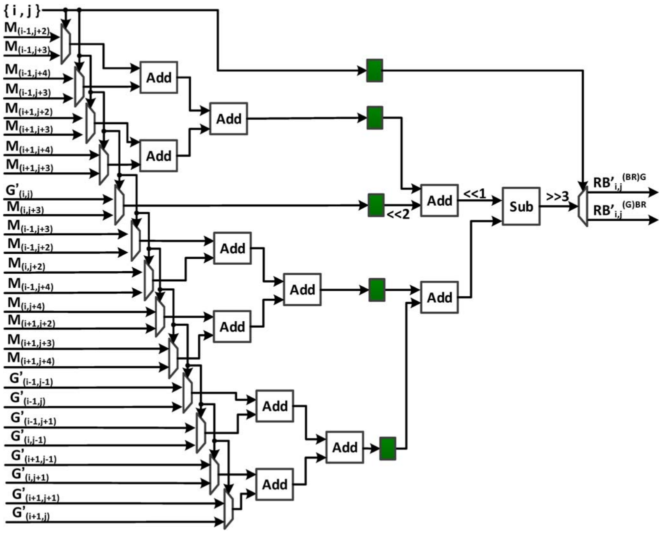


Fig. 7. Architecture of hardware sharing M3.

D. Hardware Sharing M2

Fig. 6 shows the architecture of hardware sharing M2. It is used to calculate the results of $G'_{\text{i,j+1}}$ and $RB'^{(G)RB}_{\text{i,j}}$, and it can be realized by (1) and (4). Hardware sharing M2 consists of three adders, a subtractor, three shifters, and three registers. The three registers were also designed to develop the pipeline scheduling. Moreover, since the results of $G'_{\text{i,j+1}}$ and $RB'^{(G)RB}_{\text{i,j}}$ are complementary, only a multiplexer was added to achieve hardware sharing for $G'_{\text{i,j+1}}$ and $RB'^{(G)RB}_{\text{i,j}}$. The results of $G'_{\text{i,j+1}}$ were stored in register bank 2 as inputs for hardware sharing M3. There are 50% adders, subtractors, and shifters that were reduced by using the hardware sharing technique.

E. Hardware Sharing M3

Fig. 7 shows the architecture of hardware sharing M3. It is designed to obtain the results of $RB'^{(BR)G}_{\text{i,j}}$ and $RB'^{(G)BR}_{\text{i,j}}$ and can be implemented by (2) and (3). It consists of 11 adders, a subtractor, 3 shifters, and 5 registers. The five registers were added to develop the pipeline scheduling. Hardware sharing M3 captures the original green color pixels in the CFA image from the boundary mirror machine and the interpolated green color pixels by hardware sharing M1 and M2 as inputs to calculate the results of $RB'^{(BR)G}_{\text{i,j}}$ and $RB'^{(G)BR}_{\text{i,j}}$. The difference between the interpolated green color pixels G' and original green color pixels in the CFA image can be considered as the linear deviation with the ideal value at that point. The linear deviation technique is used to compensate the results of $RB'^{(BR)G}_{\text{i,j}}$ and $RB'^{(G)BR}_{\text{i,j}}$, which greatly improves the quality of the reconstructed image by using the proposed linear deviation compensation technique. Since (2) and (3) are similar, the red–blue interpolator can be implemented by the hardware sharing technique to significantly save 45% adders, 50% subtractors, and 50% shifters.

TABLE I
COMPARISONS OF THE AVERAGE IMAGE QUALITY AND COMPUTING TIME FOR VARIOUS COLOR DEMOSAICKING ALGORITHMS WITH THIS WORK

	LHCI [8]	CDSP [9]	ACDS [10]	EECP [12]	This work
PSNR (dB)	31.10	32.63	33.21	34.81	34.87
CPSNR (dB)	30.710	32.206	32.799	34.377	34.584
SSIM	0.9718	0.9793	0.9798	0.9834	0.9829
S-CIELAB	8.0529	7.0040	6.5283	5.5701	5.5172
Executing Time (s)	4.0976	2.8444	7.7248	7.9299	2.6067

IV. SIMULATION RESULTS AND CHIP IMPLEMENTATION

In order to obtain the performance of the proposed color demosaicking algorithm, the MATLAB distribution was used to simulate the quality of reconstructed images by the hardware-oriented algorithms of low-power high-quality color interpolation (LHCI) [8], camera digital signal processor (CDSP) [9], area-efficient color demosaicking scheme (ACDS) [10], edge-enhanced color interpolation processor (EECP) [12], and this work. In addition, 24 photographs of Kodak data set, which are widely used in the image processing field, were selected as testing patterns. Each image in Kodak data set is at the resolution of 768 × 512 with 24 bits/pixel. Furthermore, four popular formulas, i.e., peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [13], color PSNR (CPSNR), and S-CIELAB $\triangle E^*$ metric [4], were used to evaluate the qualities of the interpolated images. The higher PSNR and CPSNR values denoted that the reconstructed image has more similarity to the original image. In addition, indexes called SSIM [13] and S-CIELAB $\triangle E^*$ metric [4] were used to quantify the reconstructed images if they are closer to the human eyes or not.

Table I lists the comparison of the four formula values and computing time for the previous designs with this work. The average CPSNR value in this work is 34.584 dB, which is better than 30.71, 32.21, 32.8, and 34.377 dB in the previous designs in [8], [9], [10], and [12], respectively. Compared with the previous designs [8]–[10], [12], this work improved the average values of PSNR, CPSNR, and S-CIELAB $\triangle E^*$ metric by over 0.06 dB, 0.207 dB, and 0.053, respectively. In addition, an experimental result of computing time is added for comparing time complexity, which is the average time to demosaick per image of Kodak database by a computer with Intel Core i7-2630QM when operated at 2.6 GHz. The results showed that this work had lower time complexity than the previous works. In addition, Fig. 8 shows a cropped region of the interpolated lighthouse image by the algorithms of LHCI [8], CDSP [9], ACDS [10], EECP [12], and this work for comparing the visual quality of high-frequency part and zipper effect of the interpolated image.

Table II lists the comparisons of computing resource for the previous designs with this work. The computing resource in this work contains only 17 additions, 3 subtractions, and 9 shifts.

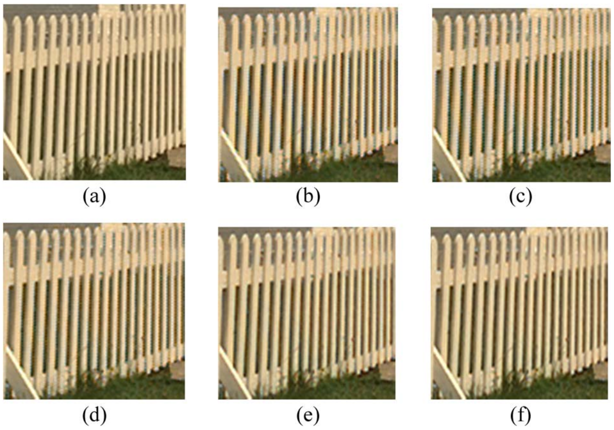


Fig. 8. Cropped region on a part of the interpolated *lighthouse* image for (a) original, (b) LHCI, (c) CDSP, (d) ACDS, (e) EECP, and (f) this work.

TABLE II
COMPARISONS OF COMPUTING RESOURCE FOR PREVIOUS DESIGNS WITH THIS WORK

	[8]	[9]	[10]	[12]	This Work	
					Without Hardware Sharing	With Hardware Sharing
Division	2	0	0	0	0	0
Multiplication	3	0	0	0	0	0
Absolute	1	0	2	6	0	0
Addition	6	20	17	30	32	17
Subtraction	1	4	5	10	6	3
Shift	0	10	13	15	18	9

By using the hardware sharing technique, 46.8% adders, 50% subtractors, and 50% shifters were successfully reduced in this work. This work can be realized by adders, subtractors, and shifters only without any dividers and multipliers. The silicon area of a divider or a multiplier is much greater than that of an adder, a subtractor, and a shifter. Moreover, compared with a previous design [12], this work achieved reduction of 43.3% adders, 70% subtractors, and 40% shifters and without using any absoluter. This work was implemented by using a hardware description language, and an EDA tool of Design Vision (Synopsys) was used to synthesize this design with the library of TSMC 0.18-μm CMOS process. The NAND-equivalent gate count in this work is only 4.97 K, and its power consumption is 4.76 mW when operates at 200 MHz. The core area is 60229 μm^2, in which the width and length are 243.85 and 246.99-μm, respectively.

Table III lists the comparisons of the previous low-complexity color demosaicking designs with this work. Compared with the previous designs, this work improved the average CPSNR by over 3.874, 2.378, 1.785, and 0.207 dB than the previous designs LHCI [8], CDSP [9], ACDS [10], and EECP [12], respectively. Moreover, this work also saved over 50.3%, 80.8%, 11.2%, and 4.4% gate counts than the previous designs LHCI [8], CDSP [9], ACDS [10], and EECP [12], respectively. The memory requirement in this work is only a two-line-buffer memory device; it is much less than 1 frame in [9]. Compared with the previous low-complexity designs, this work not only improved the quality of the interpolated images but also reduced the hardware cost and memory requirement. It provided an efficient color demosaicking VLSI design for real-time video applications.

TABLE III
COMPARISONS OF PREVIOUS LOW-COMPLEXITY COLOR DEMOSAICKING DESIGNS WITH THIS WORK

	LHCI [8]	CDSP [9]	ACDS [10]	EECP [12]	This Work
CPSNR (dB)	30.710	32.206	32.799	34.377	34.584
Process	0.35 μm	0.35 μm	0.18 μm	0.18 μm	0.18 μm
Gate Counts	10 K	26 K	5.6 K	5.2 K	4.97 K
Frequency (Hz)	40 M	50 M	200M	200 M	200 M
Power	200 mW	56 mW	150 mW	4.66 mW	4.76 mW
Core area (μm^2)	7 M	5 M	670 K	64.2 K	60 K
Memory	2 lines	1 frame	2 lines	2 lines	2 lines
Memory area (μm^2)	78.4 K	15.8 M	20.7 K	20.7 K	20.7 K
Throughput (pixel/s)	40 M	50 M	200 M	200 M	200 M
Quality	VGA	VGA	HD	HD	HD
Normalized Area	2.01	5.23	1.13	1.05	1

V. CONCLUSION

In this brief, a cost-efficient and high-performance color demosaicking VLSI design based on hardware sharing and pipeline scheduling techniques has been proposed for real-time video applications. A linear deviation compensation, immediately interpolated green color pixels, a boundary detector, and a boundary mirror machine are used to improve the quality of the reconstructed image.

REFERENCES

[1] B. E. Bayer, " *Color imaging array*," U.S. Patent 3 971 065, Jul. 1976.

[2] I. Pekkucuksen and Y. Altunbasak, "Edge strength filter based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

[3] H. A. Chang and H. H. Chen, "Stochastic color interpolation for digital cameras," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 8, pp. 964–973, Aug. 2007.

[4] K. L. Chung, W. J. Yang, W. M. Yan, and C. C. Wang, "Demosaicing of color filter array captured images using gradient edge detection masks and adaptive heterogeneity-projection," *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2356–2367, Dec. 2008.

[5] A. Buades, B. Coll, J.-M. Morel, and C. Sbert, "Self-similarity driven color demosaicking," *IEEE Trans. Image Process.*, vol. 18, no. 6, pp. 1192–1202, Jun. 2009.

[6] C. Hu, L. Cheng, and Y. M. Lu, "Graph-based regularization for color image demosaicking," in *Proc. IEEE ICIP*, Oct. 2012, pp. 2769–2772.

[7] X. Chen, G. Jeon, and J. Jeong, "Voting-based directional interpolation method and its application to still color image demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[8] S. C. Hsia, M. H. Chen, and P. S. Tsai, "VLSI implementation of low-power high-quality color interpolation processor for CCD camera," *IEEE Trans. VLSI Syst.*, vol. 14, no. 4, pp. 361–369, Apr. 2006.

[9] S. C. Hsia and P. S. Tsai, "VLSI implementation of camera digital signal processor for document projection system," in *Proc. IEEE ICSPS*, Jul. 2010, pp. 657–660.

[10] Y. H. Shiau, P. Y. Chen, and C. W. Chang, "An area-efficient color demosaicking scheme for VLSI architecture," *Int. J. Innov. Comput., Inf. Control*, vol. 7, no. 4, pp. 1739–1752, Apr. 2011.

[11] H. Chen and Y. Cheng, "VLSI implementation of color interpolation in color difference spaces," in *Proc. IEEE ISCAS*, May 2012, pp. 1680–1683.

[12] S. L. Chen and E. D. Ma, "VLSI implementation of an adaptive edge-enhanced color interpolation processor for real-time video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1991, pp. 1982–345, Nov. 2014.

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