An Efficient Edge-Based Technique for Color Filter Array Demosaicking

Chih-Yuan Lien, Fu-Jhong Yang, and Pei-Yin Chen, *Member, IEEE*

***Abstract*— An efficient edge-based technique for color filter array demosaicking is presented in this paper. The proposed algorithm uses the channel information from several correlated neighboring pixels to reconstruct the missing color channels of each pixel. We employ a simple edge detector to recognize the edge direction of each processing channel by using directional color differences, and an efficient color interpolator to reconstruct the missing color channels by observing the color correlation and edge information. The proposed technique can prevent image blur and demosaicking artefacts; moreover, it has a fixed local window size and requires no previous training and no iterations. Extensive experimental results demonstrate that the proposed technique preserves edge features and performs effectively in quantitative evaluations and visual quality. The proposed algorithm has a simple computation structure; therefore, it is appropriate for real-time hardware implementation and can be used in many real-time applications.**

***Index Terms*— Color filter array (CFA), colour interpolation, demosaicking, image reconstruction.**

I. Introduction

IMAGE sensors are widely used in current consumer electronics such as mobile phones, digital cameras, webcams and camcorders. Based on the RGB color model, a full-colour image is composed of three colour channels, namely the red (R), green (G), and blue (B) channels. Thus, three separate sensors are required for a DSC to measure the full-colour image completely. However, the sensor is usually the most expensive part of a DSC. For this reason, many cameras use only one image sensor with a colour filter array (CFA) pattern to capture digital images. Fig.1 shows the Bayer pattern [1], which is the standard CFA patterns. Because only one colour element is sampled at each pixel location, reconstructing a full-colour image requires the other two colour elements to be estimated. This is achieved through CFA interpolation, which is also known as "demosaicking."

Many algorithms that use the Bayer pattern have been proposed for this purpose [2]–[13]. One of the simplest methods is bilinear interpolation [2]. An effective colour interpolation (ECI) using signal correction was presented in [3]. In [4], Chang and Tan proposed an enhanced ECI algorithm. Pekkucuksen and Altunbasak proposed a multiscale gradient-based CFA interpolation method [5], which uses multiscale colour gradients to adaptively combine colour difference estimates from different directions. In [6], Chen et al. proposed a novel colour image demosaicking algorithm that uses voting-based edge direction detection and a directional weighted interpolation. A multidirectional weighted interpolation algorithm was proposed by Chen *et al.* [7]. The algorithm combines the eight-direction weighted interpolation procedure and an effective refinement procedure that uses gradient inverse weighted filtering. In [8], Wang and Jeon proposed a colour image demosaicking framework based on multi-directional weighted interpolation and guided filters. A CFA demosaicking process that involves using the subband synthesis method is presented by Sung and Tsao [9]. It uses the adaptive edge-sensing color interpolation method, the subband synthesis method, and the iterative subband synthesis of red and blue channels method to reduce false color artifacts and improve the quality of restored images. For real-time applications, several very-large-scale integration (VLSI) implementations for demosaicking have been proposed [10]–[13]. Fan et al. proposed an edge-oriented constant-hue scheme [10] which uses edge-oriented scheme and constant-hue interpolation to reduce costs and the complexity of the digital camera system. In [11], Shiau et al. proposed an area-efficient colour demosaicking scheme for VLSI architecture (ACDS) that uses edge information and inter-channel correlations. In [12], Chen and Ma presented an adaptive edge-enhanced colour interpolation (EECP) method, which uses an anisotropic weighting model, an edge detector, and the Laplacian and sharpening filters to reduce the memory requirement and improve image quality. The fully pipelined colour demosaicking design (FPCD) method proposed by Chen and Chang [13] uses linear deviation compensation, immediately interpolated green colour pixels, a boundary detector, and a boundary mirror machine to improve the quality of the reconstructed image.

Methods [3]–[10] all have higher image quality than methods [11]–[13], but encounter several problems such as high computational-complexity for their arithmetic operations, large local windows, and high levels of iteration. Methods [11]–[13] are more suitable for many real-time applications because of their simplicity and ease of implementation in the VLSI chip. However, our experimental results demonstrate that these

Manuscript received February 15, 2017; revised May 2, 2017; accepted May 6, 2017. Date of publication May 18, 2017; date of current version June 12, 2017. The associate editor coordinating the review of this paper and approving it for publication was Dr. Shoushun Chen. *(Corresponding author: Pei-Yin Chen.)*

C.-Y. Lien is with the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 80778, Taiwan (e-mail: cylien@kuas.edu.tw).

F.-J. Yang and P.-Y. Chen are with the Digital Integrated Circuit Design Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan (e-mail: kyo0623@hotmail.com; pychen@csie.ncku.edu.tw).

Digital Object Identifier 10.1109/JSEN.2017.2706086

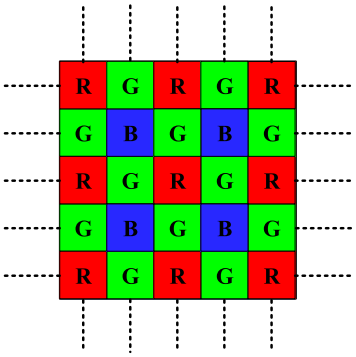


Fig. 1. Bayer pattern.

methods cannot offer satisfactory results in current consumer electronics.

Therefore, an arithmetically simple solution that has a fixed local window size and no iterations is developed in this article. Moreover, edge techniques were widely used in many digital image processing fields [4]–[19] to improve image quality or to avoid image blur. Hence, we propose an efficient edge-based demosaicking method (EEDM) suitable for low-cost VLSI implementation.

The rest of this article is organised as follows: Section II briefly introduces the proposed EEDM, Section III shows the simulation results, and Section IV presents the conclusion.

II. Proposed Algorithm

The CFA investigated in this article uses the Bayer pattern described in [2]–[13] and a mask size of 3×5. To achieve the goal of low cost, less memory and simple computations are necessary. Therefore, the limits are no more than 3 lines of buffer, low computational complexity, and no iteration. The proposed algorithm uses the channel information from several neighbouring pixels to reconstruct the missing colour channels of each pixel. Image may be blurred if noncorrelated neighboring channel values are averaged, therefore an edge technique was applied to improve image quality.

The proposed algorithm has two main components: a simple edge detector, which recognizes the edge direction of each processing channel by using directional colour differences, and an efficient colour interpolator, which reconstructs the missing colour channels by observing the colour correlation and edge information. The design concept of the EEDM is displayed in Fig. 2.

A. Simple Edge Detector

To locate the edge existing in the current mask, a simple edge technique was adopted that can be easily implemented using software and hardware. Assume that the current pixel to be processed is located at coordinate (i, j). To identify an edge and achieve low computational-complexity arithmetic operations, the horizontal and vertical edge strength were first determined; they are calculated by averaging several directional differences in the mask and denoted as e_h and e_v, respectively, and are defined

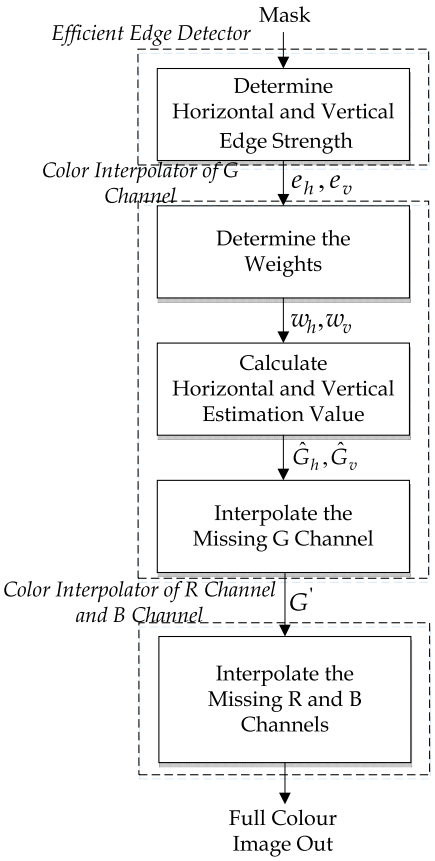


Fig. 2. Dataflow of EEDM.

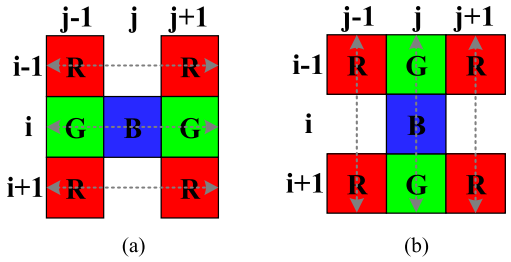


Fig. 3. Correlated neighboring pixels of e_h and e_v when $C = R$. (a) e_h. (b) e_v.

as follows:

$$e_h(i,j) = \frac{1}{4} \times \begin{pmatrix} |C(i-1,j-1) - C(i-1,j+1)| + \\ 2 \times |G(i,j-1) - G(i,j+1)| + \\ |C(i+1,j-1) - C(i+1,j+1)| \end{pmatrix}, \tag{1}$$

$$e_v(i,j) = \frac{1}{4} \times \begin{pmatrix} |C(i-1,j-1) - C(i+1,j-1)| + \\ 2 \times |G(i-1,j) - G(i+1,j)| + \\ |C(i-1,j+1) - C(i+1,j+1)| \end{pmatrix}. \tag{2}$$

$C(i-1, j-1)$, $C(i-1, j+1)$, $C(i+1, j-1)$, and $C(i+1, j+1)$ are the original neighbouring channel sample values; furthermore, $C \in \{R, B\}$. If $C = R$, the correlated neighbouring pixels and directions are shown in Fig. 3 and the equations

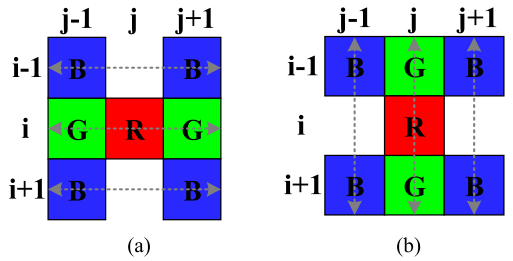


Fig. 4. Correlated neighboring pixels of e_h and e_v when $C = B$. (a) e_h. (b) e_v.

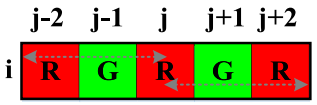


Fig. 5. Correlated neighboring pixels of $\hat{G}_h(i, j)$ when $C = R$.

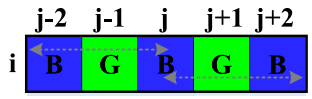


Fig. 6. Correlated neighboring pixels of $\hat{G}_h(i, j)$ when $C = B$.

are expressed as follows:

$$e_h(i,j) = \frac{1}{4} \times \begin{pmatrix} |R(i-1,j-1) - R(i-1,j+1)| + \\ 2 \times |G(i,j-1) - G(i,j+1)| + \\ |R(i+1,j-1) - R(i+1,j+1)| \end{pmatrix}, \tag{3}$$

$$e_v(i,j) = \frac{1}{4} \times \begin{pmatrix} |R(i-1,j-1) - R(i+1,j-1)| + \\ 2 \times |G(i-1,j) - G(i+1,j)| + \\ |R(i-1,j+1) - R(i+1,j+1)| \end{pmatrix}. \tag{4}$$

The smaller value has the stronger relation with the processing pixel and probably has an edge aligned in its direction. Conversely, the higher value has the slighter relation with the processing pixel. Fig.4 shows the correlated neighbouring pixels and directions of e_h and e_v at the case of "$C = B$". By observing the spatial correlation, the simple edge detector finds the directional edges and uses them to generate the estimated value of current pixel in the following component.

B. Efficient Color Interpolator

To improve the quality of interpolated images, an efficient colour interpolator was used to interpolate the missing channels of each pixel. It consists of two sections, described as follows:

1) Interpolating the Missing Green Channel: A weighted colour interpolation algorithm was employed to reconstruct the missing G colour channel of pixels that only have R or B colour channel and to improve the quality of the interpolated images. It uses the weighted average of the colour difference between the closest two neighbouring channels and the centre channel. This process is divided into two steps. Step 1 entails calculating the horizontal and vertical G values, denoted as $\hat{G}_h$ and $\hat{G}_v$, by using simple estimated G values ($\tilde{G}$) and the difference of the processing colour channel value (ΔC), where $C \in \{R, B\}$. The factor ΔC is based on the colour correlation between the processing sample and a neighbouring sample of the same channel. The directional estimates for the missing G values are:

$$\hat{G}_h(i,j) = \tilde{G}_h(i,j) + \Delta C_h(i,j), \tag{5}$$

$$\hat{G}_v(i,j) = \tilde{G}_v(i,j) + \Delta C_v(i,j). \tag{6}$$

Where the horizontal equations for $\tilde{G}$ and ΔC are given as follows:

$$\tilde{G}_h(i,j) = \frac{(G(i,j-1) + G(i,j+1))}{2}, \tag{7}$$

$$\Delta C_h(i,j) = \frac{1}{2} \times \begin{pmatrix} \frac{1}{2} \times (C(i,j) - C(i,j-2)) + \\ \frac{1}{2} \times (C(i,j) - C(i,j+2)) \end{pmatrix}. \tag{8}$$

When $C = R$, the equations are expressed as follows:

$$\hat{G}_h(i,j) = \tilde{G}_h(i,j) + \Delta R_h(i,j), \tag{9}$$

$$\hat{G}_v(i,j) = \tilde{G}_v(i,j) + \Delta R_v(i,j), \tag{10}$$

$$\Delta R_h(i,j) = \frac{1}{2} \times \begin{pmatrix} \frac{1}{2} \times (R(i,j) - R(i,j-2)) + \\ \frac{1}{2} \times (R(i,j) - R(i,j+2)) \end{pmatrix}. \tag{11}$$

Fig.5 and Fig.6 show the correlated neighbouring pixels of $\hat{G}_h(i, j)$ at the cases of "$C = R$" and "$C = B$", respectively. The equations for the vertical estimation are similar. In Step 2, e_h and e_v are used as the weighted factors for interpolating the missing G channel. When $e_h > e_v$, the possibility of a vertical edge being present is indicated; thus, $\hat{G}_v(i, j)$ should affect the interpolated G value more than $\hat{G}_h(i, j)$ does. Contrastingly, $e_h < e_v$ indicates the possibility of a horizontal edge being present. Therefore, the weights of the directional estimates $\hat{G}_h(i, j)$ and $\hat{G}_v(i, j)$ are denoted as w_h and w_v, respectively, and defined as follows:

$$w_h = \frac{\frac{1}{e_h(i,j)}}{\frac{1}{e_h(i,j)} + \frac{1}{e_v(i,j)}} = \frac{e_v(i,j)}{e_h(i,j) + e_v(i,j)}, \tag{12}$$

$$w_v = \frac{\frac{1}{e_v(i,j)}}{\frac{1}{e_h(i,j)} + \frac{1}{e_v(i,j)}} = \frac{e_h(i,j)}{e_h(i,j) + e_v(i,j)}. \tag{13}$$

Consequently, the interpolated G value of the processed channel is given as

$$G'(i,j) = w_h(i,j) \times \hat{G}_h(i,j) + w_v(i,j) \times \hat{G}_v(i,j). \tag{14}$$

2) Interpolating the Missing R and B Channels: The colour interpolation of the aforementioned G channel is not well-suited to reconstructing the missing R and B colour channel because only one weight can be calculated. Therefore, we use the G-R and G-B information for interpolation. For real-world images, the contrasts of G-R and G-B are relatively flat over small regions; this property is appropriate for colour

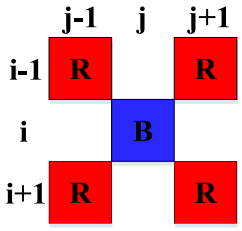


Fig. 7. Correlated neighboring pixels of d_{GR} when $C = R$.

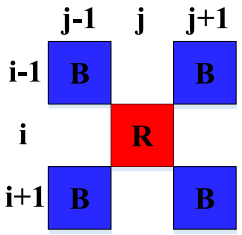


Fig. 8. Correlated neighboring pixels of d_{GB} when $C = B$.

interpolation [3]–[13]. To interpolate the missing R channel value, we calculate G-R at the up, down, left, and right positions. To avoid misdetection caused by choosing channel positions that contain no original R colour channel, only those channel positions containing original R colour channel are considered. Hence, we calculate the mean value of G-R and denote it as d_{GR}. The calculation is similar for the mean of G-B and is denoted as d_{GB}. We take d_{GC} to represent d_{GR} and d_{GB} and define it as follows:

$$d_{GC} = \frac{1}{n} \times \sum G'(i+k, j+l) - C(i+k, j+l), \quad (15)$$

where $C \in \{R, B\}$, $(k, l) \in \{(-1,-1), (-1,1), (1,-1), (1,1)\}$ and n is the number of pixel positions containing original C colour information.

The interpolated R or B value of the processed pixel is therefore represented as C' and denoted as

$$C'(i, j) = G'(i, j) - d_{GC}. \quad (16)$$

Where $C \in \{R, B\}$.

Fig.7 and Fig.8 show the correlated neighbouring pixels and positions of d_{GC} at the cases of "$C = R$" and "$C = B$", respectively. For example, the locations of target pixels in the original CFA image is blue, and $C = R$, the equations are expressed as follows:

$$d_{GR} = \frac{1}{4} \times \begin{pmatrix} G'(i-1, j-1) - R(i-1, j-1) + \\ G'(i-1, j+1) - R(i-1, j+1) + \\ G'(i+1, j-1) - R(i+1, j-1) + \\ G'(i+1, j+1) - R(i+1, j+1) \end{pmatrix}, \quad (17)$$

$$R'(i, j) = G'(i, j) - d_{GR}. \quad (18)$$

The equations for $B'(i, j)$ are very similar. Finally, upon completion of these two steps, we obtain a full colour image. The proposed algorithm has a simple computation structure and is suitable for real-time hardware implementation.

TABLE I

COMPARISON OF RESTORATION RESULTS IN CPSNR FOR VARIOUS DEMOSAICKING METHODS (KODAK DATA-SET)

CPSNR	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
Image01	26.59	29.66	30.41	30.60	**34.14**
Image02	33.19	35.88	36.40	36.54	**38.47**
Image03	34.62	37.96	38.09	38.81	**41.04**
Image04	33.93	36.75	37.13	37.80	**39.26**
Image05	26.84	30.70	31.42	31.89	**35.17**
Image06	27.89	31.08	31.22	32.09	**34.99**
Image07	33.74	37.28	38.02	38.08	**40.80**
Image08	23.76	26.99	28.22	27.47	**31.97**
Image09	32.64	35.69	36.93	36.43	**40.12**
Image10	32.68	36.11	36.63	37.09	**40.49**
Image11	29.29	32.46	33.01	33.38	**36.53**
Image12	33.75	36.66	37.19	37.53	**39.89**
Image13	24.06	27.40	27.29	28.62	**31.39**
Image14	29.31	32.64	32.86	33.46	**35.52**
Image15	33.24	35.92	36.32	36.93	**38.43**
Image16	31.43	34.58	34.64	35.49	**38.34**
Image17	32.29	35.51	36.02	36.49	**39.19**
Image18	28.33	31.60	31.75	32.69	**35.22**
Image19	28.20	31.41	32.94	31.83	**36.54**
Image20	31.75	35.19	35.86	35.99	**38.95**
Image21	28.65	31.90	32.15	32.91	**35.92**
Image22	30.59	33.53	34.18	34.28	**36.85**
Image23	35.38	38.67	39.13	39.45	**40.95**
Image24	26.97	30.14	30.22	31.33	**33.69**
Avg	30.38	33.57	34.08	34.46	**37.24**

TABLE II

COMPARISON OF RESTORATION RESULTS IN S-CIELAB FOR VARIOUS DEMOSAICKING METHODS (KODAK DATA-SET)

S-CIELAB	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
Image01	4.88	3.69	3.38	3.50	**2.21**
Image02	3.00	2.38	2.29	2.33	**1.80**
Image03	1.71	1.29	1.28	1.27	**1.01**
Image04	2.35	1.85	1.79	1.76	**1.37**
Image05	5.43	4.09	3.75	3.84	**2.52**
Image06	3.53	2.46	2.43	2.43	**1.62**
Image07	2.02	1.56	1.44	1.52	**1.14**
Image08	5.59	4.47	3.85	4.27	**2.40**
Image09	1.76	1.41	1.26	1.35	**0.94**
Image10	1.74	1.37	1.27	1.30	**0.93**
Image11	3.63	2.79	2.63	2.71	**1.85**
Image12	1.42	1.09	1.05	1.07	**0.82**
Image13	6.85	5.02	5.00	4.66	**3.16**
Image14	3.93	2.83	2.80	2.76	**1.96**
Image15	2.45	2.03	1.93	1.93	**1.57**
Image16	2.72	1.91	1.91	1.91	**1.32**
Image17	2.68	2.14	2.05	2.06	**1.54**
Image18	4.74	3.60	3.49	3.35	**2.43**
Image19	3.41	2.64	2.37	2.52	**1.56**
Image20	2.10	1.66	1.56	1.59	**1.12**
Image21	3.32	2.46	2.40	2.35	**1.62**
Image22	2.77	2.18	2.05	2.05	**1.55**
Image23	1.40	1.16	1.12	1.13	**1.02**
Image24	3.44	2.61	2.43	2.42	**1.67**
Avg	3.20	2.45	2.31	2.34	**1.63**

III. IMPLEMENTATION RESULTS AND COMPARISONS

To verify the characteristics and performances of various demosaicking algorithms, a variety of simulations were conducted on the 24-image 768 × 512 Kodak data-set [20], and 500-image 220 × 132 MSR data-set [21]. In the simulations, images were processed as a Bayer CFA to create the CFA test images; several approaches were then employed to reconstruct full-colour images. CPSNR, S-CIELAB and standard

TABLE III

COMPARISON OF RESTORATION RESULTS IN SD FOR VARIOUS DEMOSAICKING METHODS (KODAK DATA-SET)

SD	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
Image01	11.68	8.22	7.54	7.29	**4.98**
Image02	5.45	4.00	3.76	3.68	**2.93**
Image03	4.66	3.19	3.11	2.83	**2.23**
Image04	5.04	3.64	3.49	3.22	**2.68**
Image05	11.43	7.34	6.74	6.35	**4.43**
Image06	10.05	7.01	6.86	6.06	**4.52**
Image07	5.16	3.43	3.15	3.13	**2.30**
Image08	16.09	11.00	9.67	10.52	**6.39**
Image09	5.83	4.08	3.56	3.76	**2.50**
Image10	5.81	3.91	3.68	3.47	**2.39**
Image11	8.58	5.97	5.60	5.32	**3.77**
Image12	5.13	3.67	3.45	3.29	**2.56**
Image13	15.74	10.80	10.86	9.21	**6.86**
Image14	8.57	5.85	5.67	5.24	**4.19**
Image15	5.47	4.01	3.83	3.55	**2.96**
Image16	6.68	4.69	4.62	4.09	**3.07**
Image17	6.11	4.23	3.99	3.77	**2.79**
Image18	9.62	6.63	6.50	5.79	**4.41**
Image19	9.65	6.64	5.61	6.35	**3.77**
Image20	6.47	4.36	4.04	3.96	**2.86**
Image21	9.23	6.38	6.18	5.58	**4.06**
Image22	7.39	5.27	4.91	4.85	**3.65**
Image23	4.27	2.92	2.78	2.67	**2.25**
Image24	11.22	7.82	7.71	6.70	**5.22**
Avg	8.14	5.63	5.30	5.03	**3.66**

TABLE IV

COMPARISON OF RESTORATION RESULTS IN CPSNR FOR VARIOUS DEMOSAICKING METHODS (MSR DATA-SET)

CPSNR	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
1-100	32.99	35.78	36.14	36.12	**38.60**
101-200	29.28	32.19	32.72	32.56	**35.45**
201-300	32.07	34.73	35.23	35.15	**37.65**
301-400	30.39	33.20	33.59	33.63	**36.29**
401-500	32.17	34.86	35.21	35.32	**37.64**

deviation (SD) error measures were used to determine the difference between the original and reconstructed images. For the Bayer pattern, the results of the proposed EEDM method were compared to those of four demosaicking methods (BI [2], ACDS [11], EECP [12], and FPCD [13]) by means of objective testing (quantitative evaluation) and subjective testing (visual quality) using the parameters and thresholds recommended for these methods. The mask sizes of the comparison methods were 3×3 (BI [2]) and 3×5 (ACDS [11], EECP [12], FPCD [13]), respectively. Table I-III listed the CPSNR (dB), S-CIELAB and SD error measures of each 24 test images on Kodak data set, respectively. Tables IV-VI listed the restoration results in average CPSNR, S-CIELAB and SD error measures of each 100 test images on MSR data set, respectively. Furthermore, the box plot of restoration results in CPSNR for various demosaicking methods in MSR data-set are shown in Fig. 9.

The proposed algorithm outperformed other methods on the 24-image Kodak data-set and 500-image MSR data set in Table I-VI, respectively. It exhibited an average CPSNR 2.78 dB higher than that of the closest method (FPCD) in Table I. Therefore, it is easy to see that the quantitative qualities of the proposed method were always superior to those of other methods.

TABLE V

COMPARISON OF RESTORATION RESULTS IN S-CIELAB FOR VARIOUS DEMOSAICKING METHODS (MSR DATA-SET)

S-CIELAB	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
1-100	3.34	2.93	2.80	2.88	**2.24**
101-200	4.37	3.81	3.54	3.75	**2.70**
201-300	3.86	3.43	3.21	3.35	**2.61**
301-400	4.14	3.65	3.42	3.52	**2.67**
401-500	3.37	2.92	2.77	2.82	**2.25**

TABLE VI

COMPARISON OF RESTORATION RESULTS IN SD FOR VARIOUS DEMOSAICKING METHODS (MSR DATA-SET)

SD	BI[2]	ACDS[11]	EECP[12]	FPCD[13]	**Proposed**
1-100	6.62	4.80	4.64	4.58	**3.50**
101-200	9.00	6.47	6.13	6.20	**4.55**
201-300	7.18	5.21	4.95	4.98	**3.71**
301-400	8.15	5.90	5.69	5.58	**4.19**
401-500	6.84	4.99	4.83	4.72	**3.64**

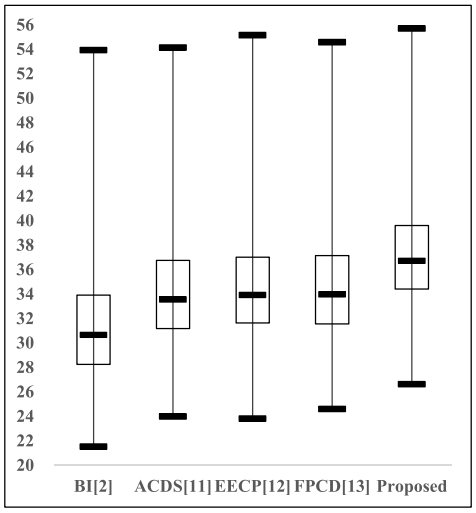
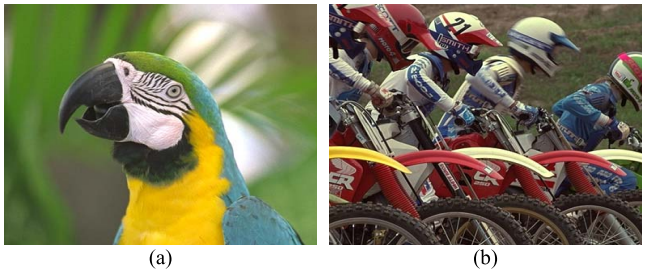


Fig. 9. Box plot of restoration results in CPSNR for various demosaicking methods (MSR data-set).



(a) (b)

Fig. 10. Two reference images for testing. (a) Image23, (b) Image05.

For subjective testing, the original and restored versions of Image23, and Image05 are shown in Figs. 10-13. A portion of each picture was enlarged to evaluate visual quality. The restored versions of EEDM are very similar with the original images. To verify the characteristics and quality of the demosaicked images, we take the difference of colour channel between the original image and the restoring image. It is defined as (the colour channel of demosaicked image)

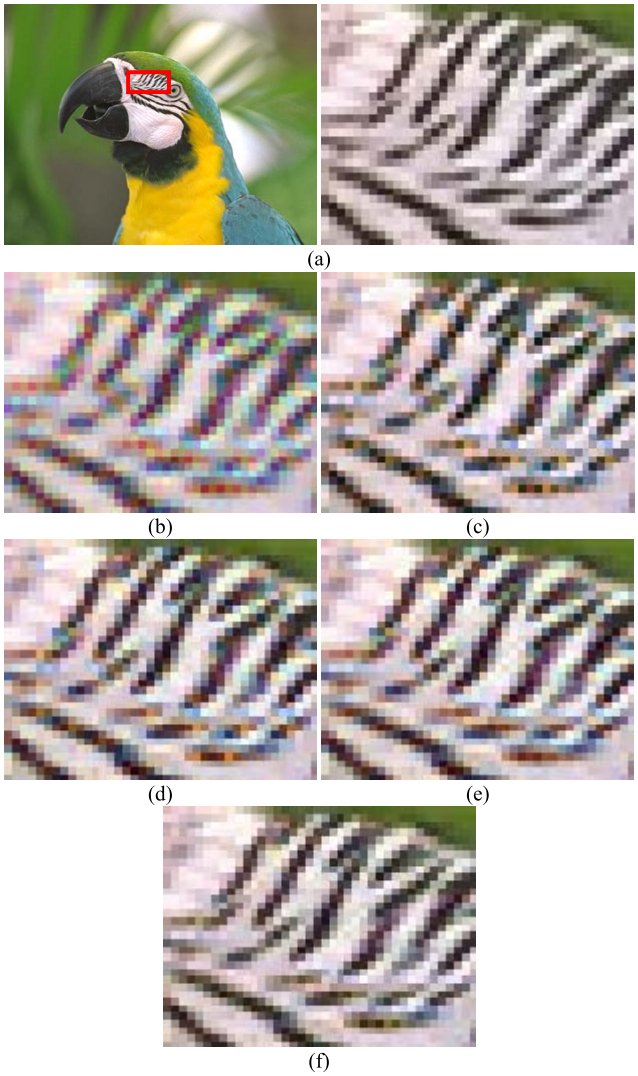


Fig. 11. Restoration results of different methods for restoring the corrupted image Image23. (a) Original image. (b) BI. (c) ACDS. (d) EECP. (e) FPCD. (f) Proposed.

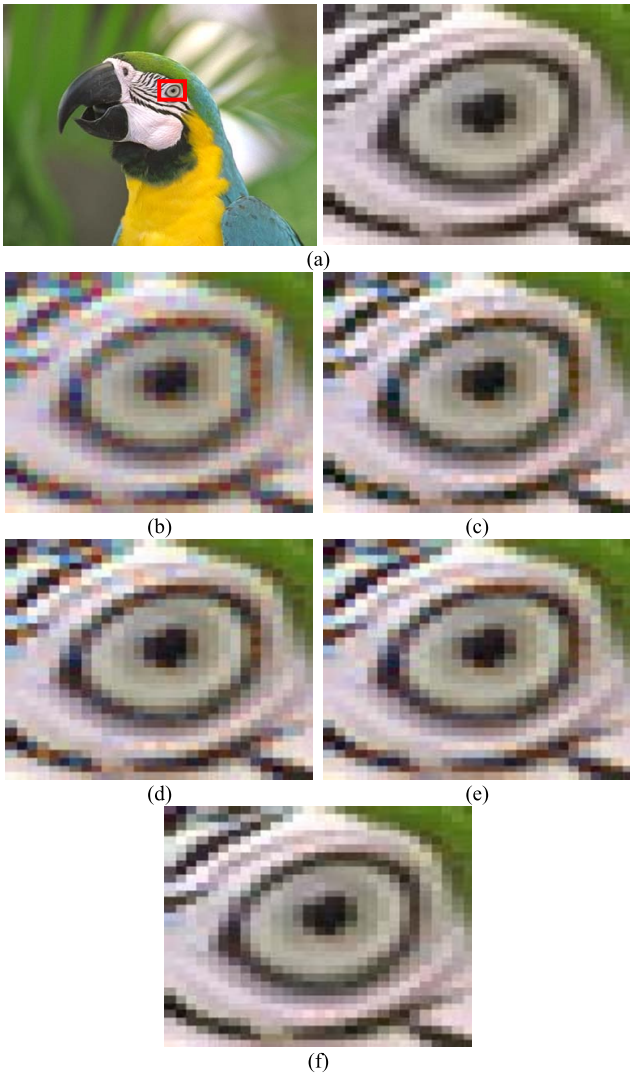


Fig. 12. Restoration results of different methods for restoring the corrupted image Image23. (a) Original image. (b) BI. (c) ACDS. (d) EECP. (e) FPCD. (f) Proposed.

TABLE VII

COMPARISON OF COMPUTATIONAL COMPLEXITY FOR DIFFERENT METHODS

Operation	EECI[4]	MGBI[5]	MDWI[7]	SSM[9]	**Proposed**
ADD/SUB	126 *MN*	84.5 *MN*	189 *MN*	75.5 *MN*	**21 *MN***
MUL/DIV	33 *MN*	17 *MN*	42.5 *MN*	20 *MN*	**1.5 *MN***
SHT	4 *MN*	14.5 *MN*	21.5 *MN*	32 *MN*	**10 *MN***
ABS	32 *MN*	1 *MN*	61.5 *MN*	4 *MN*	**3 *MN***
Mask	5*5	5*5	9*9	5*5	**3*5**
Iteration Times	1	1	1	1	**0**

minus (the same colour channel of orginal image) plus 128. If the demosaicked image and the orginal image are similar, the colour difference channels are quite flat. Fig. 14 shows the B channel's colour difference of Image05's cropped regions for comparison. The proposed method shows quite flat result than other methods. Obviously, the proposed method preserved edges very effectively and produced visually attractive pictures. Therefore, we can conclude that the proposed EEDM outperforms other methods [2], [11]–[13] in terms of quantitative evaluation and visual quality.

To explore the computational complexity, we listed the required operations for the proposed EEDM and other state-of-the-art methods [4], [5], [7], [9] in Table VII. The proposed EEDM requires much less computational complexity than other methods [4], [5], [7], [9] and the visual qualities of the resulting images are good and acceptable.

The experimental results demonstrate that the proposed EEDM performed excellently in both the objective and subjective tests. Moreover, EEDM has a very simple computation structure and two-line-memory buffer, it does not need such large frequency operation processing as convolution integration or large amounts of multiplication and division operations. Thus, it is suitable for low-cost VLSI implementation. For some consumer electronics, a faster execution time is required. Thus, we adopt the pipelined architecture. The proposed VLSI architectures of the proposed EEDM were implemented using Verilog HDL.

By employing pipeline scheduling to the optimized state flowchart (see Fig. 2), a 4-stage pipelined hardware

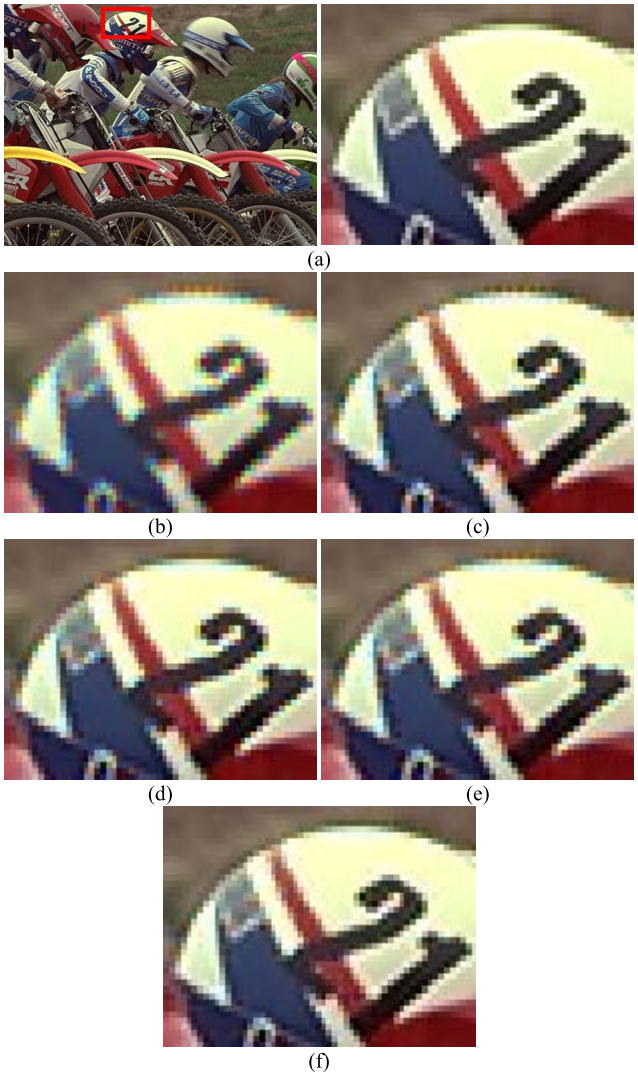


Fig. 13. Restoration results of different methods for restoring the corrupted image Image05. (a) Original image. (b) BI. (c) ACDS. (d) EECP. (e) FPCD. (f) Proposed.

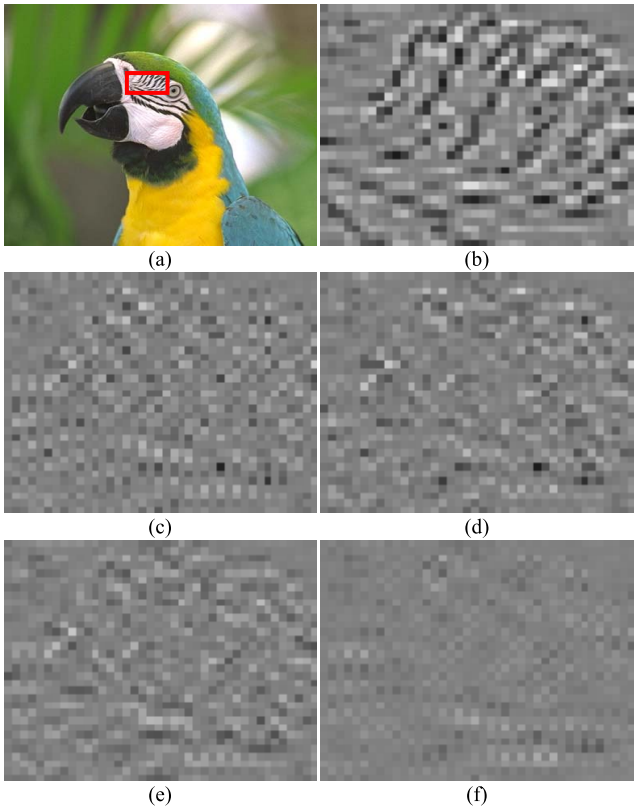


Fig. 14. The B channel's colour difference of Image05's cropped regions for comparison. (a) Original image. (b) BI. (c) ACDS. (d) EECP. (e) FPCD. (f) Proposed.

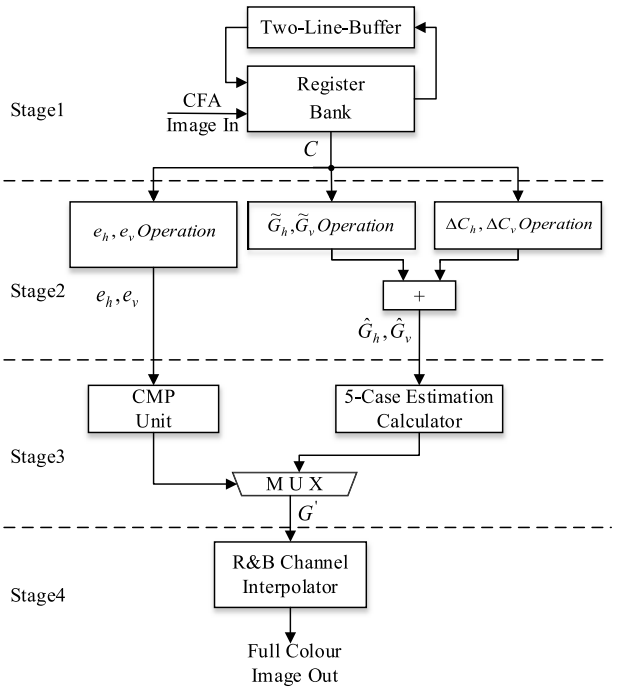


Fig. 15. Dataflow of proposed.

architecture was developed and shown in Fig. 15. In Stage 1, we realize a module of a register bank to store fifteen values of a 3×5 window of the CFA test image concurrently, and a module of two line buffers to store two rows of the processed colour components. Since the size of the CFA test images is 768×512, the required size for one line buffer is 768 bytes. However, the larger the image, the bigger the line buffer size is required. Once our design has completed the CFA demosaicking process of a pixel in the image, the reconstructed pixel values will be outputted and stored into the line buffer at every cycle for the following subsequent processing. The register bank, consisting of fifteen registers, is designed with a shift register style and used to store current 3×5 mask C. The fifteen pixel values stored in RB will be used simultaneously by the following e_h, e_v *Operation*, $\tilde{G}_h$, $\tilde{G}_v$ *Operation*, and ΔC_h, ΔC_v *Operation* modules in Stage 2. To realize the EEDM with achieving the goal of lower cost, the divider of weights of the directional estimates operation is replaced by CMP unit, 5-case estimation calculator, and a multiplexer in Stage 3. The CMP unit determines the type of edge from e_h and e_v, and the 5-case estimation calculator is used to calculate the five possible G' values. In the last stage, the R&B channel interpolator is used to get the remaining channels.

For an image of size M × N pixels, when the start signal is enabled, the hardware core will output the first pixel after (2M + 8) clock cycles. Then, it can process one pixel per clock cycle. Totally, it would take {M × (N + 2) + 8} cycles to process an image.

We used SYNOPSYS Design Vision to synthesise the proposed EEDM designs with the TSMC 0.18 μm cell library. The layouts for the designs were generated using a SYNOPSYS IC Compiler (for autoplacement and routing) and verified using MENTOR GRAPHIC Calibre (for the DRC and LVS checks). Nanosim was used for post-layout transistor-level simulation. Finally, SYNOPSYS PrimePower was employed to measure the total power consumption. Synthesis results demonstrate that the EEDM chip contains 8880 gate counts and the chip areas is 100.59k μm^2. The power consumptions of the architecture is 18.17 mW. It operated at 5 ns and achieved a processing rate of approximately 200 megapixels per second (MP/s) which is quick enough to process a video resolution of Full HD (1920 × 1080) at 30 fps in real time. The operating clock frequency of the EEDM is 200 MHz.

IV. Conclusion

An efficient edge-based reconstruction method for CFA patterns was proposed in this article. The proposed method applies arithmetic operations of low computational-complexity, a fixed-size local window, and no iterations. The experimental results indicate that our method performed excellently in both the objective and subjective tests. Moreover, the proposed algorithm has a very simple computation structure and requires only two lines for line buffering, therefore it can be used in many real-time applications.

References

[1] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, Jul. 20, 1976.

[2] R. Ramanath, W. E. Snyder, G. L. Bilbro, and W. A. Sander, III, "Demosaicking methods for Bayer colour arrays," *J. Electron. Imag.*, vol. 11, no. 3, pp. 306–315, Jul. 2002.

[3] S.-C. Pei and I.-K. Tam, "Effective color interpolation in CCD color filter arrays using signal correlation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 503–513, Jun. 2003.

[4] L. Chang and Y. P. Tan, "Effective use of spatial and spectral correlations for color filter array demosaicking," *IEEE Trans. Consum. Electron.*, vol. 50, no. 1, pp. 355–365, Feb. 2004.

[5] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[6] 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.

[7] X. Chen, G. Jeon, J. Jeong, and L. He, "Multidirectional weighted interpolation and refinement method for Bayer pattern CFA demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1271–1282, Aug. 2015.

[8] L. Wang and G. Jeon, "Bayer pattern CFA demosaicking based on multi-directional weighted interpolation and guided filter," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 2083–2087, Nov. 2015.

[9] D.-C. Sung and H.-W. Tsao, "Color filter array demosaicking by using subband synthesis scheme," *IEEE Sensors J.*, vol. 15, no. 11, pp. 6164–6172, Nov. 2015.

[10] Y.-C. Fan, Y.-F. Chiang, and Y.-T. Hsieh, "Constant-hue-based color filter array demosaicking sensor for digital still camera implementation," *IEEE Sensors J.*, vol. 13, no. 7, pp. 2586–2594, Jul. 2013.

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

[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. 11, pp. 1982–1991, Nov. 2014.

[13] S.-L. Chen and H.-R. Chang, "Fully pipelined low-cost and high-quality color demosaicking VLSI design for real-time video applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 6, pp. 588–592, Jun. 2015.

[14] Y. H. Shiau, H. Y. Yang, P. Y. Chen, and Y. Z. Chuang, "Hardware implementation of a fast and efficient haze removal method," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 8, pp. 1369–1374, Aug. 2013.

[15] C. Lee, W. Chao, S. Lee, J. Hone, A. Molnar, and S. H. Hong, "A low-power edge detection image sensor based on parallel digital pulse computation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 11, pp. 1043–1047, Nov. 2015.

[16] R. Wang, M. Pakleppa, and E. Trucco, "Low-rank prior in single patches for nonpointwise impulse noise removal," *IEEE Trans. Image Process.*, vol. 24, no. 5, pp. 1485–1496, May 2015.

[17] F. Vedadi and S. Shirani, "De-interlacing using nonlocal costs and Markov-chain-based estimation of interpolation methods," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1559–1572, Apr. 2013.

[18] Z. Song, D. Wang, Z. Huang, and Y. Pang, "Edge pattern based demosaicking algorithm of color filter array," *Trans. Tianjin Univ.*, vol. 19, no. 1, pp. 29–36, Feb. 2013.

[19] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Trans. Image Process.*, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[20] *Kodak Color Image Dataset*, accessed on Mar. 11, 2013. [Online]. Available: http://r0k.us/graphics/kodak/

[21] *MSR Demosaicing Data Set*, accessed 2014. [Online]. Available: http://research.microsoft.com/en-us/um/cambridge/projects/msrdemosaic/

Chih-Yuan Lien received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in computer science and information engineering from National Cheng Kung University, Taiwan, in 2009. He is currently an Assistant Professor with the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan. His research interests include image processing, VLSI chip design, and video coding system.

Fu-Jhong Yang received the B.S. degree in electronic communication engineering from National Kaohsiung Marine University, Kaohsiung, Taiwan, in 2011, and the M.S. degree in electronic engineering from the National Kaohsiung University of Applied Sciences, Kaohsiung, in 2013. He is currently pursuing the Ph.D. degree in computer science and information engineering at National Cheng Kung University, Taiwan. His research interests include image processing, VLSI chip design, and data compression.

Pei-Yin Chen (M'08) received the B.S. degree in electrical engineering from National Cheng Kung University, Taiwan, in 1986, the M.S. degree in electrical engineering from Penn State University, Pennsylvania, in 1990, and the Ph.D. degree in electrical engineering from National Cheng Kung University in 1999. He is currently a Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include VLSI chip design, video compression, fuzzy logic control, and gray prediction.