IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 2, Issue 3 (May. – Jun. 2013), PP 28-39
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

An Efficient Vlsi Architecture For Removal Of Impulse Noise In Image Using Edge Preseving Filter

Mr.R.Muralikrishnan[1], Mr.A.Suresh Babu[2]
[1]*(PG Scholar (EEE),Hindusthan college of Engineering and Technology, Coimbatore)*
[2]*(Asst.professor(EEE),Hindusthan College of Engineering and Technology, Coimbatore)*

Abstract: *Images are often corrupted by impulse noise in the procedures of image acquisition and transmission. In this paper, we propose An Efficient VLSI Architecture for Removal of Impulse Noise in Image using edge preserving filter, to achieve low-complexity VLSI architecture. We employ a decision-tree-based impulse noise detector to detect the noisy pixels, and an edge-preserving filter along with mathematical morphological filter to reconstruct the intensity values of noisy pixels. Furthermore, an adaptive technology is used to enhance the effects of removal of random valued impulse noise. Our extensive experimental results demonstrate that the proposed technique can obtain better performances in terms of both quantitative evaluation and visual quality than the previous lower-complexity methods. Moreover, the performance can be comparable to the higher-complexity methods. The VLSI architecture of our design yields a processing rate of about 200 MHz by using TSMC 0.18μm technology. The design requires only low computational complexity and two line memory buffers. Its hardware cost is low and suitable to be applied to many real-time applications.*

Keywords *- edge preserving filter, Image denoising, impulse noise, impulse detector, image acquisition*

I. INTRODUCTION

Image processing is widely used in many fields, such as medical imaging, scanning techniques, printing skills, license plate recognition, face recognition, and so on. In general, images are often corrupted by impulse noise in the procedures of image acquisition and transmission. The noise may seriously affect the performance of image processing techniques. Hence, an efficient denoising technique becomes a very important issue in image processing [1], [2]. According to the distribution of noisy pixel values, impulse noise can be classified into two categories: fixed-valued impulse noise and random-valued impulse noise. The former is also known as salt-and-pepper noise because the pixel value of a noisy pixel is either minimum or maximum value in grayscale images. The values of noisy pixels corrupted by random- valued impulse noise are uniformly distributed in the range of [0, 255] for gray-scale images. There have been many methods for removing salt-and-pepper noise, and some of them perform very well [3]-[7]. The random-valued impulse noise is more difficult to handle due to the random distribution of noisy pixel values. We only focus on removing the random-valued impulse noise from the corrupted image in this paper. Recently, many image denoising methods have been proposed to carry out impulse noise suppression [8]-[18]. Some of them employ the standard median filter [8] or its modifications [9], [10]. However, these approaches might blur the image since both noisy and noise-free pixels are modified. To avoid the damage on noise-free pixels, an efficient switching strategy has been proposed in the literature [11]- [13]. In general, the switching median filter consists of two steps: 1) impulse detection and 2) noise filtering. It locates the noisy pixels with an impulse detector, and then filters them rather than the whole pixels of an image to avoid causing the damage on noise-free pixels. In addition to median filer, there are other methods used to carry out impulse noise. In [14], Wenbin Luo proposed an alpha-trimmed mean based method (ATMBM). It used the alpha-trimmed mean in impulse detection and replaced the noisy pixel value by a linear combination of its original value and the median of its local window. A differential rank impulse detector (DRID) was presented in [15]. The impulse detector of DRID is based on a comparison of signal samples within a narrow rank window by both rank and absolute value. In [16], Yu, Zhao and Wang proposed a method using a statistic of rank ordered relative differences (RORD-WMF) to identify pixels which are likely to be corrupted by impulse noise. A directional weighted median (DWM) method proposed by Dong and Xu was presented in [17]. It is based on the differences between the current pixel and its neighbors aligned with four main directions. In [18], Petrović NI and Crnojević V proposed a method that employed genetic programming for impulse noise filter construction. The method is based on the switching scheme with cascaded detectors and corresponding estimators. Generally, the denoising methods can be classified into two categories: lower-complexity techniques [8]-[13] and higher-complexity techniques [14]-[18]. The complexity of denoising algorithms depends mainly on the local window size, memory buffer, and iteration times. The lower complexity techniques use a fixed-size local window, require a few line buffers and perform no iterations. Therefore, the computational complexity is low. However, the reconstructed image quality is not good enough. The higher complexity techniques yield visually pleasing images by using high computational complexity arithmetic operations, enlarging local window size adaptively or doing iterations. The higher-complexity approaches require long computational time as well as full frame buffer. Today in many practical real-time applications, the denoising process is included in end-user equipment, so there appears an increasing need of a good lower-complexity demolishing technique, which is simple and suitable for low-cost VLSI implementation. Low cost is a very important consideration in purchasing consumer electronic products. To achieve the goal of low cost, less memory and easier computations are indispensable. In this paper, we focus only on the lower-complexity denoising techniques because of its simplicity and easy implementation with the VLSI circuit. The decision tree is a simple but powerful form of multiple variable analysis [19]. It can break down a complex decision-making process into a collection of imple decisions, thus provide a solution which is often easier to interpret[20]. There have been several methods using decision tree to deal with salt-and-pepper noise [4], [5], [21]-[23] and some of them perform well. Based on above basic concepts, we present a novel adaptive decision-tree-based denoising method (DTBDM) and its VLSI architecture for removing random-valued impulse noise. To enhance the effects of removal of impulse noise, the results of reconstructed pixels are adaptively written back as a part of input data. The proposed design requires simple computations and two line memory buffers only, so its hardware cost is low. For a 512×512 8-bit gray-scale test image, only two line buffer (512×2×8 bits) is needed in our design. Most state-of-the-art methods need to buffer a full image (512×512×8 bits). In our design, 99.6% of storage is reduced. Furthermore, only simple arithmetic operations, such as addition and subtraction, are used in DTBDM. Especially, it can remove the noise from corrupted images efficiently and requires no previous training. Our extensive experimental results demonstrate that the proposed technique can obtain better performances in terms of both quantitative evaluation and visual quality than other lower-complexity denoising methods [8]-[13]. Moreover, the performance can be comparable to the higher-complexity methods [14]-[16].The seven-stage VLSI architecture for the proposed design was implemented and synthesized by using Verilog HDL and Synopsys Design Compiler, respectively. In our simulation, the circuit can achieve 200 MHz with only 21k gate counts by using TSMC 0.18μm technology. The rest of this paper is organized as follows. The proposed DTBDM is introduced briefly in Section 2. Section 3 describes the proposed VLSI architecture in detail. Section 4 illustrates the VLSI implementation and comparisons. The conclusion is provided in Section 5.

II. The Proposed Dtbdm

The noise considered in this paper is random-valued impulse noise with uniform distribution as practiced in [8]-[18]. Here, we adopt a 3×3 mask for image denoising. Assume the pixel to be denoised is located at coordinate (*i, j*) and denoted as *pi,j*, and its luminance value is named as *fi,j*, as shown in Fig. 1. According to the input equence of image denoising process, we can divide other eight pixel values into two sets: *W Top Half* and *W Bottom Half*. They are given as

$$W_{TopHalf} = \{a, b, c, d\} \quad (1)$$
$$W_{BottomHalf} = \{e, f, g, h\} \quad (2)$$

DTBDM consists of two components: decision-tree-based impulse detector and edge-preserving image filter. The detector determines whether *pi,j* is a noisy pixel by using the decision tree and the correlation between pixel *pi,j* and its neighboring pixels. If the result is positive, edge-preserving image filter based on direction-oriented filter generates the reconstructed value. Otherwise, the value will be kept unchanged. The design concept of the DTBDM is displayed in Fig. 2.

2.1 Decision-Tree-Based Impulse Detector

In order to determine whether *pi,j* is a noisy pixel, the correlations between *pi,j* and its neighboring pixels are considered [10],[11], [14], [16], [17], [23]-[30]. Surveying these methods, we can simply classify them into several ways observing the degree of isolation at current pixel [10], [11], [14], [16], [17],[23]-[25], determining whether the current pixel is on a fringe[16], [17], [26], [27] or comparing the similarity between current pixel and its neighboring pixels [16], [28]-[30]. Therefore, in our decision-tree-based impulse detector, we design three modules-isolation module, fringe module, and similarity module. Three concatenating decisions of these modules build a decision tree. The decision tree is a binary tree and can determine the status of *pi,j* by using the different equations in different modules. First, we use isolation module to decide whether the pixel value is in a smooth region. If the result is negative, we conclude that the current pixel belongs to noisy-free. Otherwise, if the result is positive, it means that the current pixel might be a noisy pixel or just situated on an edge. The fringe module is used to confirm the result. If the current pixel is situation an edge, the result of fringe module will be negative (noisy-free); otherwise, the result will be positive. If isolation module and fringe module cannot determine whether current pixel belongs to noisy free, the similarity module is used to decide the result. It compares the similarity between current pixel and its neighboring pixels. If the result is positive, *pi,j* is a noisy pixel; otherwise, it is noise free. The following sections describe the three modules in detail.

2.1.1 Isolation Module (IM)

The pixel values in a smooth region should be close or locally slightly varying, as shown in Fig. 3. The differences between its neighboring pixel values are small. If there are noisy values, edges or blocks in this region, the distribution of the values is different, as shown in Fig. 4. Therefore, we determine whether current pixel is an isolation point by observing the smoothness of its surrounding pixels. Fig. 5 shows an example of noisy image. The pixels with shadow suffering from noise have low similarity with the neighboring pixels and the so-called isolation point. The difference between it and its nerghboring pixel value is large. According to the above concepts, we first detect the maximum and minimum luminance values in *W Top Half*, named as *Top Half _max*, *Top Half _min*, and calculate the difference between them, named as *Top Half_ diff*. For *W Bottom Half*, we apply the same idea to obtain Bottom *Half _diff*. The two difference values are compared with a threshold *The IMa*to decide whether the surrounding region belongs to a smooth area.
The equations are as:

$$TopHalf\_diff = TopHalf\_max - TopHalf\_min. \qquad (3)$$

$$BottomHalf\_diff = BottomHalf\_max - BottomHalf\_min. \qquad (4)$$

$$Decision\ I = \begin{cases} true, & \text{if } (TopHalf\_diff \geq Th\_IM_a) \\ & \text{or} (BottomHalf\_difff \geq Th\_IM_a) \\ false, & \text{otherwise.} \end{cases} \qquad (5)$$

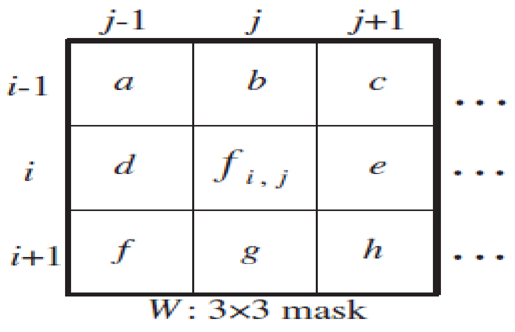
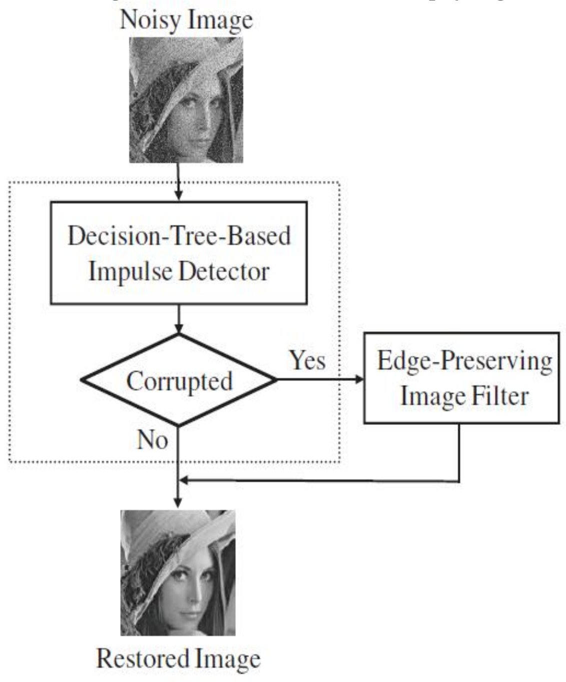
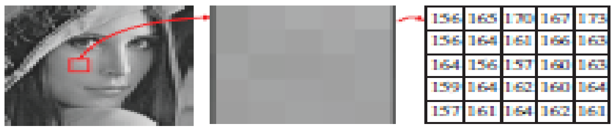


Fig. 1. A 3×3 mask centered on *pi,j*.Fig



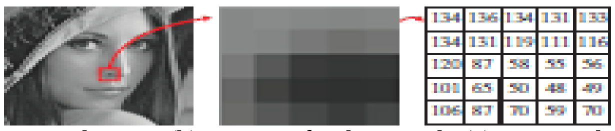
Noisy Image
Decision-Tree-Based Impulse Detector
Corrupted
Yes
No
Edge-Preserving Image Filter
Restored Image

Fig. 2. The dataflow of DTBDM



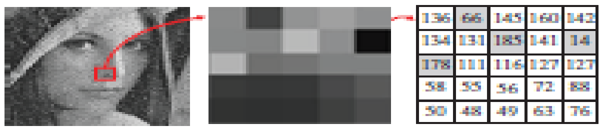
156	165	170	167	173
156	164	161	166	163
164	156	157	160	163
159	164	162	160	164
157	161	164	162	161

(a) original image (b) Region of red rectangle (c) Gray-scale value
Fig. 3. A smooth region in Lena



134	136	134	131	133
134	131	119	111	116
120	87	58	55	56
101	65	50	48	49
106	87	70	59	70

(a) original image (b) Region of red rectangle (c) Gray-scale value
Fig. 4. A non-smooth region in Lena



136	66	145	160	142
134	131	185	141	14
178	111	116	127	127
58	55	56	72	88
50	48	49	63	76

(a) original image (b) Region of red rectangle (c) Gray-scale value
Fig. 5. The difference between noisy and neighboring pixels in Lena

$$IM\_TopHalf = \begin{cases} true, & \text{if } (\left|f_{i,j} - TopHalf\_max\right| \geq Th\_IM_b) \\ & or(\left|f_{i,j} - TopHalf\_min\right| \geq Th\_IM_b) \\ false, & \text{otherwise.} \end{cases} \quad (6)$$

$$IM\_BottomHalf = \begin{cases} true, & \text{if } (\left|f_{i,j} - BottomHalf\_max\right| \geq Th\_IM_b) \\ & or(\left|f_{i,j} - BottomHalf\_min\right| \geq Th\_IM_b) \\ false, & \text{otherwise.} \end{cases} \quad (7)$$

$$Decision\ II = \begin{cases} true, & \text{if } (IM\_TopHalf = true) \\ & or(IM\_BottomHalf = true) \\ false. & \text{otherwise.} \end{cases} \quad (8)$$

2.1.2 Fringe Module (FM)

If *pi,j* has a great difference with neighboring pixels, it might be a noisy pixel (discussed in Section 2.1.1 IM) or just situated on an edge, as shown in Fig. 6. How to conclude that a pixel is noisy or situated on an edge is difficult. In order to deal with this case, we define four directions, from *E*1 to *E*4, as shown in Fig. 7. We take direction *E*1 for example. By calculating the absolute difference between *fi,j* and the other two pixel values along the same direction respectively, we can determine whether there is an edge or not. The equations are as:

$$FM\_E_1 = \begin{cases} false, & \text{if } (\left|a - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|h - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|a - h\right| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases} \quad (9)$$

$$FM\_E_2 = \begin{cases} false, & \text{if } (\left|c - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|f - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|c - f\right| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases} \quad (10)$$

$$FM\_E_3 = \begin{cases} false, & \text{if } (\left|b - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|g - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|b - g\right| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases} \quad (11)$$

$$FM\_E_4 = \begin{cases} false, & \text{if } (\left|d - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|e - f_{i,j}\right| \geq Th\_FM_a) \\ & or(\left|d - e\right| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases} \quad (12)$$

$$Decision\ III = \begin{cases} false, & \text{if } (FM\_E_1)\ or(FM\_E_2) \\ & or(FM\_E_3)\ or(FM\_E_4) \\ true, & \text{otherwise.} \end{cases} \quad (13)$$

2.1.3 Similarity Module (SM)

The last module is similarity module. The luminance values in mask *W* located in a noisy-free area might be close. The median is always located in the canter of the variational series, while the impulse is usually located near one of its ends. Hence, if there are extreme big or small values, that implies the possibility of noisy signals. According to this concept, we sort nine values in ascending order and obtain the 4th, 5th and 6th values which are close to the median in mask *W*. The 4th, 5th and 6th values are represented as 4*thinWi,j*, *Median In Wi,j* and 6*thinWi,j*.

$$\begin{aligned} Max_{i,j} &= 6_{th} in W_{i,j} + Th\_SM_a , \\ Min_{i,j} &= 4_{th} in W_{i,j} - Th\_SM_a \ . \end{aligned} \qquad (14)$$

$$N_{\max} = \begin{cases} Max_{i,j}, \text{ if } (Max_{i,j} \le MedianInW_{i,j} + Th\_SM_b) \\ MedianInW_{i,j} + Th\_SM_b, \quad \text{otherwise.} \end{cases} \qquad (15)$$

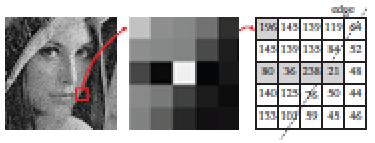
$$N_{\min} = \begin{cases} Min_{i,j}, \text{ if } (Min_{i,j} \ge MedianInW_{i,j} - Th\_SM_b) \\ MedianInW_{i} - Th\_SM_b, \quad \text{otherwise.} \end{cases} \qquad (16)$$

Maxi,j and *Mini,j* are used to determine the status of pixel *pi,j*. However, in order to make the decision more precisely, we do some modifications as Finally, if *fi,j* is not between *Nmax* and *Nmin*, we conclude that *pi,j* is a noise pixel. Edge-preserving image filter will be used to build the reconstructed value. Otherwise, the original value *fi,j* be the output. The equation is as

$$Decision\ IV = \begin{cases} true, \text{ if } (f_{i,j} \ge N_{\max}) \text{ or } (f_{i,j} \le N_{\min}) \\ false, \quad \text{otherwise.} \end{cases} \qquad (17)$$

:

Obviously, the threshold affects the quality of denoised images of the proposed method. A more appropriate threshold contributes to achieve a better detection result. However, it is not easy to derive an optimal threshold through analytic formulation. The fixed values of thresholds make our algorithm simple and suitable for hardware implementation. According to our extensive experimental results, the thresholds *Th_IMa*, *TH_IMb*, *Th_FMa*, *Th_FMb*, *Th_SMa*and *Th_SMb*are all predefined valuesand set as 20, 25, 40, 80, 15 and 60, respectively.



(a) original image (b) Region of red rectangle (c) Gray-scale value
Fig. 6. The edge region in Lena

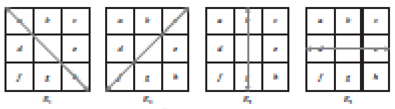


Fig. 7. Four directions in DTBDM.

2.2 Edge-Preserving Image Filter

To locate the edge existing in the current *W*, a simple edge preserving technique which can be realized easily with VLSI circuit is adopted. The dataflow and the pseudo code of our edge-preserving image filter are shown in Fig. 8 and 9, respectively .Here, we consider eight directional differences, from *D*1to *D*8, to reconstruct the noisy pixel value, as shown in Fig. 10and eq. (18). Only those composed of noise-free pixels are taken into account to avoid possible misdetection. Directions passing through the suspected pixels are discarded to reduce miss detection. Therefore, we use *Maxi j* and *Mini,j*, defined in similarity module (SM), to determine whether the values of *d*, *e*, *f*, *g* and *h*are likely corrupted respectively. If the pixel is likely being corrupted by noise, we don't consider the direction including the suspected pixel. In the second block, if *d*, *e*, *f*, *g* and *h* are all suspected to be noisy pixels, and no edge can be processed, so*fi, j*ˆ (the estimated value of *pi,j*) is equal to the weighted average of luminance values of three previously denoised pixels and calculated as (*a* +*b* *2+*c)*/4 . In other conditions, the edge filter calculates the directional differences of the chosen directionsand locates the smallest one (*D*min) among them in thethird block. The equations are as follows. In the last block of Fig. 8, the smallest directional difference implies that it has the strongest spatial relation with *pi,j*, and probably there exists an edge in its direction. Hence, the mean of luminance values of the pixels which possess the smallest directional difference is treated as *fi, j* ˆ . After *fi, j* ˆ is determined, a tuning skill is used to filter the bias edge. If

fi, *j* ˆ obtain the correct edge, it will situate at the median of *b*, *d*, *e* and *g* because of the spatial relation and the characteristic of edge preserving. Otherwise, the values of *fi*, *j* will be replaced by the median of four neighboring pixels (*b*, *d*, *e* and *g*). We can express
fi, *j* as *fi*, *j*= Median (*f*ˆ*i*, *j* ,*b*,*d*,*e*, *g*).

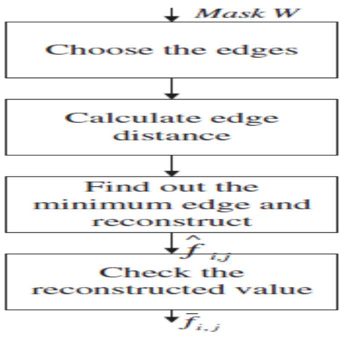


Fig. 8. Dataflow of edge-preserving image filter.

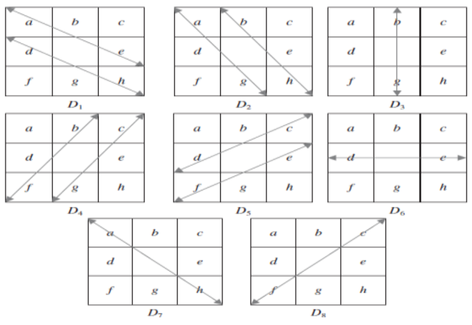


Fig. 9. Eight directional differences of DTBDM

III. Vlsi Implementation Of Dtbdm

DTBDM has low computational complexity and requires only two line buffers instead of full images, so its cost of VLSI implementation is low. For better timing performance, we adopt the pipelined architecture to produce an output at every clock cycle. In our implementation, the SRAM used to store the image luminance values is generated with the 0.18μm TSMC/Artisan memory compiler [31], and each of them is 512×8 bits. According to the simulation results obtained from Design Ware of SYNOPSYS [32], we find that the access time for SRAM is about 5 ns. Hence, we adopt the 7-stage pipelined architecture for DTBDM. Fig. 11 shows block diagram of the VLSI architecture for DTBDM. The architecture adopts an adaptive technology and consists of five main blocks: line buffer, register bank, decision tree- based impulse detector, edge-preserving image filter and controller. Each of them is described briefly in the following subsections.

3.1 Adaptive Technology

The proposed method employs an adaptive technology to improve the quality of reconstructed image. Fig. 11 shows the architecture of the proposed adaptive algorithm. The reconstructed pixels are adaptively written or stored into the line buffers. The current pixel to be denoised is located at coordinate (*i*,,*j*) . The adaptive points located at coordinate (*i* -1, *j* -1) , (*i* -1, *j*) and (*i* -1, *j* -1) are already denoised at the previous denoising process. Since we adopt a pipelined hardware architecture in this work, the denoised value located at coordinate (*i*, *j* □1) is still in the pipeline and not available, as shown in Fig. 12. Here, 6 original points and 3 adaptive points are combined as 9 input pixels for the 3×3 mask to reconstruct the resulting pixel *fi*, *j* . Although only 3 adaptive points (less than half) of 9 input points are available, the reconstructed value is significantly affected by the adaptive points, since each adaptive point is written back to influence the next point continuously

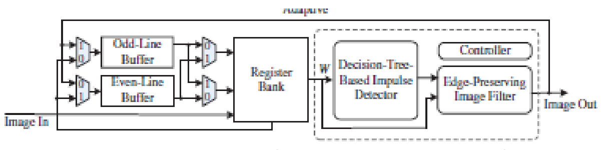


Fig. 10. Block diagram of VLSI architecture of DTBDM

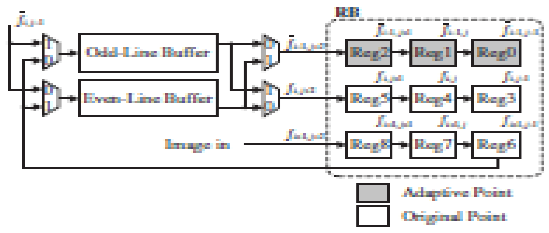
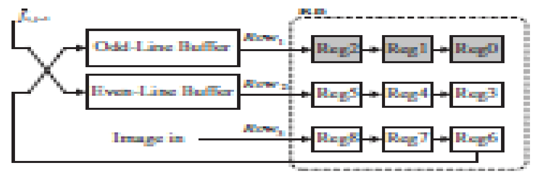
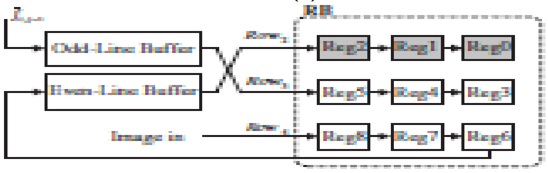


Fig. 11. Architecture of register bank (RB) in DTBDM.



(a)



(b)

Fig. 12. Two examples of the interconnections between two line buffers and RB

3.2 Line Buffer (Ping-Pong Buffer)

DTBDM adopts a 3×3 mask, so three scanning lines are needed. If *pi*, *j*are processed, three pixels from *rowi*-1, *rowi* and *rowi*+1, are needed to perform the denoising process. Here, we use the concept of ping-pong arrangement. With the help of four crossover multiplexers (see Fig. 12), we realize three scanning lines with two line buffers. Odd-Line Buffer and Even- Line Buffer are designed to store the pixels at odd and even rows respectively, as shown in Fig. 12.To reduce cost and power consumption, the line buffer is implemented with a dual-port SRAM (one port for reading out data and the other for writing back data concurrently) instead of a series of shifter registers. If the size of an image is *Iw*×*Ih*, thesize required for one line buffer is *Iw*-3 bytes in which 3 represents the number of pixels stored in the register bank.

3.3 Register Bank

The register bank (RB), consisting of 9 registers, is used to store the 3×3 pixel values of the current mask *W*. Figure 12 shows its architecture where each 3 registers are connected serially in a chain to provide three pixel values of a row in *W*, and the Reg4 keeps the luminance value (*fi*,*j*) of the current pixel to be denoised. Obviously, the denoising process for *pi*,*j*doesn't start until *fi*+1,*j*+1 enterers from the input device. The nine values stored in RB are then used simultaneously by subsequent data detector and noise filter for denoising. Once the denoising process for *pi*,*j*is completed, the reconstructed pixel value *fi*, *j* generated by the edge-preserving filter is outputted and written into the line buffer storing *rowi*to replace *fi*,*j*. When the denoising process shifts from *pi*,*j*to *pi*,*j*+1, only 3 new values (*fi*-1,*j*+2, *fi*,*j*+2, *fi*+1,*j*+2) are needed to be read into RB (Reg2, Reg5 and Reg8 respectively) and other 6 pixel values are shifted to each one's proper register. At the same time, the previous input value from the input device, *fi*+1,*j*+1, is written back to the line buffer storing *rowi*-1 for subsequent denoising process. The selection signals of the four multiplexers are all set to 1 or 0 for denoising the odd or the even rows respectively. Two examples are shown in Fig. 13 to illustrate the interconnections between the two line buffers and RB. Assume that we denoise*row*2, and set all four selection signals to 0, those samples of *row*1 and *row*2 are stored in Odd-Line Buffer and Even-Line Buffer, respectively. The samples of *row*3 are inputted from the input device, as shown in Fig. 13(a). The previous value in Reg6 and the denoised results generated by the edge-preserving filter are written back to Odd-Line Buffer and Even-Line Buffer, respectively. After the denoising process of *row*2 has been completed, Odd-Line Buffer is now full with the whole samples of *row*3, while those denoised samples of *row*2 are all stored in Even-Line Buffer. To denoise*row*3, we set all selections signals to 1. Thus the previous value in Reg6 and the denoised results generated by the edge preserving filter are written back to Even-Line Buffer and Odd-Line Buffer respectively, as shown in Fig. 13(b). After the denoising process of *row*3 has been completed, Even-Line Buffer is now full with the whole samples of *row*4, while those denoised samples of *row*3 are stored in Odd-Line Buffer.

3.4 Decision-Tree-Based Impulse Detector

The decision-tree-based impulse detector is composed of three modules (isolation module, fringe module and similarity module). Each of them is described in the following subsections.

3.4.1 Isolation Module (IM)

Fig. 14 shows the architecture of IM (we take *W Top Half* for example). The comparator CMPL is used to output the larger value from the two input values while the comparator CMPS is used to output the smaller value from the two input values. The first two-level comparators are used to find *Top Half _max* and *Top Half _min*. The SUB unit is used to output the difference which is subtracted the lower input (*Top Half _min*) from the upper one(*Top Half _max*), and the |SUB| unit is used to output the absolute value of difference of two inputs. The GC is the greater comparator that will output logic 1 if the upper input value is grater than the lower one. The OR gate is employed to generate the binary result for *IM_Top Half*. Finally, if the result of *Decision II* is positive, *pi,j* might be a noisy pixel or situate on an edge. The next module (FM) will be used to confirm the result.

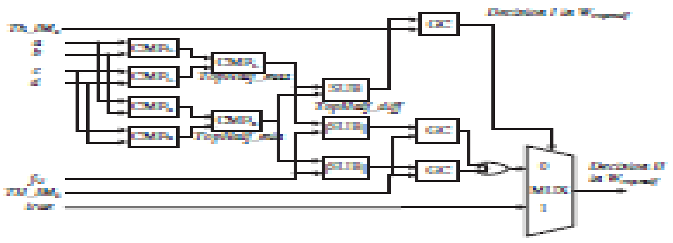


Fig. 13. Architecture of IM (*WTopHalf*)

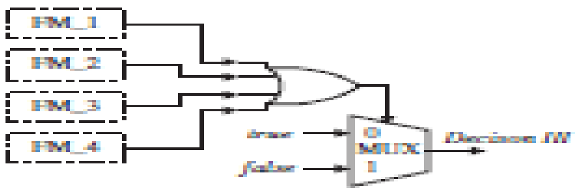


Fig. 14. Architecture of FM

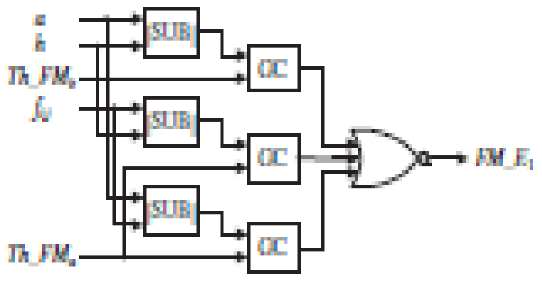


Fig. 15. Architecture of FM_1 module

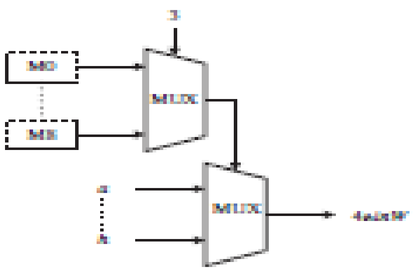


Fig. 16. Architecture of sorting (4thinW)

3.4.2 Fringe Module (FM)

Fig. 15 shows the architecture of FM. The FM is composed of four small modules, from FM_1 to FM_4, and each of them is used to determine its direction, as mentioned in subsection 2.1.2. Fig. 16 is a detailed implementation of M_1. Since *E*1 is the direction from *a* to *h* (Fig. 7), the relation between *a*, *h* and *fi,j* must be referenced. The three |SUB| units are used to determine the absolute differences between them. The GC is described in the above section and the NOR gate is used to generate the result of *FM_E*1. If the result is positive, we consider that *fi,j*is on the edge *E*1 and regard it as noise-free.

3.4.3 Similarity Module (SM)

If IM and FM can't determine whether *fi,j* belongs to a noise free value or not, SM is used to confirm the result. Fig. 17 shows our architecture that is designed to accelerate the sorting speed to obtain the 4th value in mask *W*. The detailed implementation of module M0 is shown in Fig. 18. If *a* is greater than *b*, *C*01 is set to 1; otherwise, *C*01 is set to 0. The eight GC units are used to determine the values from *C*01 to *C*08. After comparing, a combined unit is used to combine the results of each comparator to obtain a number between 0 and 8. The number indicates the order of value in mask *W*. If *a* is the smallest value in mask W, the output of the M0 module is 0; if *a* is the biggest value in mask W, the output is 8. The architectures of other modules (M1 to M8) are almost the same as M0, with only little difference. By means of this implementation, we can find out the order of values efficiently by using simple comparators and combined units. Comparing with traditional sorting algorithms, this method not only speeds up the sorting time, but also reduces the space which used to store the value to be exchanged. Fig. 19 shows the architecture of SM after we obtain the 4th, 5th and6th values in mask *W*. The SUB and ADD units are used to calculate the value of *Max i,j* and *Min i,j* as mentioned in Section 2.1.3. The Two GC and MUX units are used to determine the *N max* and *N min*. The TC unit is a triple input comparator which can output the logic 1 if the lowest input is not between the upper two inputs.

3.5 Edge-Preserving Image Filter

The Edge-Preserving Image Filter is composed of two modules, min ED generator and average generator (AG). Fig. 20 shows the architecture of the min ED generator which is used to determine the edge that has the smallest difference. Eight directional differences are calculated with twelve |SUB|, four ADD and four shifter units. Then the smallest one is determined by using the Min Tree unit. Min Tree is made up of a series of comparators. After that, the mean of luminance values of the pixels which process the smallest directional difference (*D min*) can be obtained from the average generator, as shown in Fig. 21. As mentioned in Section 2, if *pi,j*-1, *pi,j*+1, *pi*+1,*j*-1, *pi*+1,*j* and *pi*+1,*j*+1 are all suspected to be noisy pixels, the final MUX will output (a+b×2+c)/4. Otherwise, the MUX will output the mean of the pixel values which process *D min*. Some directional differences are determined according to four pixel values, so its reconstructive values also need four pixel values. Two-level ADD and shifter units are used to complete our calculation. As for the chip implementation, the gate counts or silicon area of one multiplier or division is much larger than one shifter. In our design, all multipliers or divisions will be replaced by shifter units in order to lower the hardware cost. After above computations, we sort *b*, *d*, *e* and *g* in order. The reconstructed value *fi, j* ˆ obtained from edge-preserving filter will be compared with the 2th and 3th values, named as *Sort-Four*2, *SortFour*3, and the final value *fi, j* is obtained from the equation as

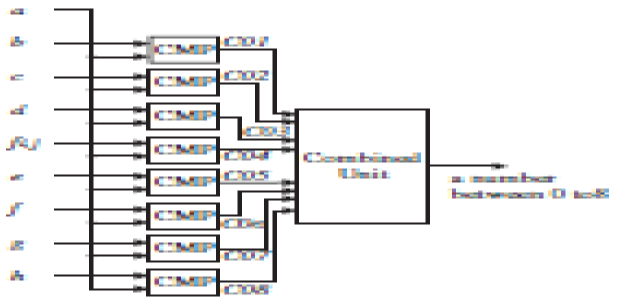


Fig. 17. Architecture of M0 module

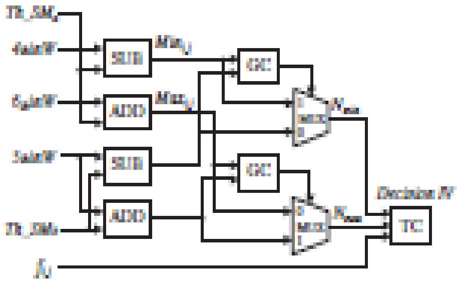


Fig. 18. Architecture of SM

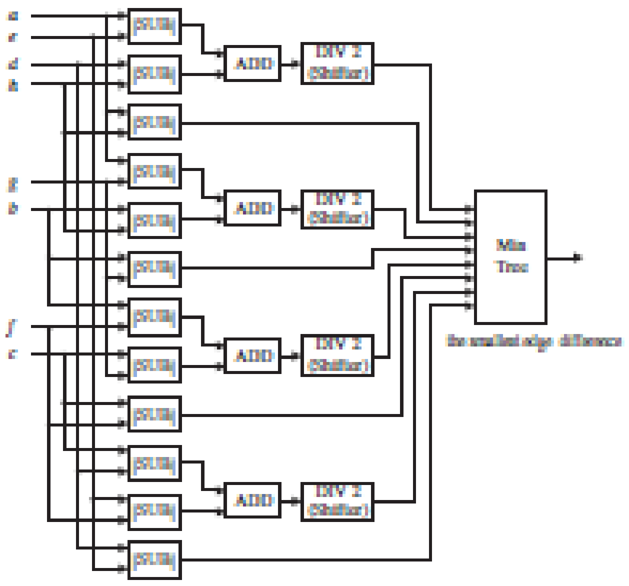


Fig. 19. Architecture of minED generator

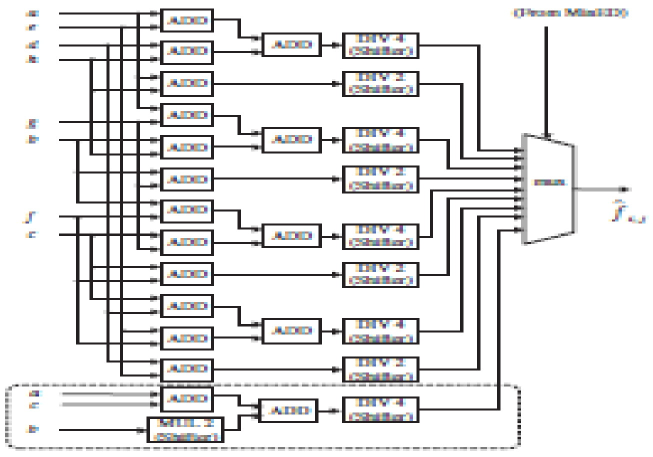


Fig. 20. Architecture of average generator (AG)

3.6 Controller

Controller sends signals to control pipelining and timing statuses of the proposed circuits. It also sends control signals to schedule reading and writing statuses of the data that are stored in register bank or in line-buffers. The realization of the controller is based on the concept of finite state machine (FSM). By the controller design, the proposed circuit can automatically receive stream-in data of original images and produce stream-out results of reconstructed images.

IV. Implementation Results And Comparisons

To verify the characteristics and the quality of denoised images of various denoising algorithms, a variety of simulations are carried out on the six well-known 512×512 8-bit gray-scale test images: Lena, Boat, Couple, Peppers, Airplane, and Goldhill. For a single test image, the corrupted versions of it are generated in MATLAB environment with random-valued impulse noise at various noise densities from 5% to 20% with increments of 5%. Then we employ different approaches to detect impulse noise and restore the corrupted image. Thus, we can easily compare the restored images with the source image for various denoising methods. Totally, twelve denoising methods (Median Filter [8],ACWM [12], MSM(3:T) [13], MSM(5:T) [13], MSM(7:T) [13],ATMBM [14], DRID [15], RORD-WMF [16], RVNP [33],AMF [34], NAVF [35], LCNR [36]) and our method (DTBDM)are compared in terms of objective testing (quantitative evaluation)and subjective testing (visual quality) where the parameters or thresholds of these methods are set as suggested. We employ the peak signal-to-noise ratio (PSNR) to illustrate the quantitative quality of the reconstructed images for various methods. Table 1, 2, 3, and 4 list the restoration results in PSNR(dB) of test images corrupted by 5%, 10%, 15% and 20% impulses, respectively. The first ten belong to lower-complexity methods and perform no iteration.

The others belong to higher complexity methods and have more complicated operations and iterations. Comparing with those experimental results, the quantitative qualities of DTBDM are always better than those lower complexity methods in low noise ratio and almost the same with other higher-complexity methods. Table 5 lists the mask size, line buffer and iteration times of each methods. Since most methods are software implementation and some have complex operations, it is hard to list the detailed operations, such as the numbers of adder, subtraction, multiplication, and division. Generally, the cost of VLSI implementation depends mainly on the required memory and computational complexity. The proposed design requires only few computations. Hence, we use line buffer and iteration times to prove that our design requires lower cost. In [14], the author claims that their method achieves better filtering quality with lower complexity than other methods[9]-[11]. Therefore, we can conclude that our DTBD Mout performs other methods [8]-[16], [33]-[36] in terms of quantitative evaluation. Table 6 shows the probabilities of impulse detection, including the false alarms and misses. The false alarm means that there is no impulse but detector says there is one. Since the quantitative qualities of our method are much better than those lower-complexity methods, we only list the comparisons between our method and other higher-complexity methods. Although ATMBM and DRID can reduce the numbers of misses by doing iteration, they also increase the numbers of false alarm, as shown in table 6. The total numbers (misses + false alarms) in this work are much lower than those higher-complexity methods To explore the visual quality, we show the reconstructed images of different denoising methods in restoring 20% corrupted image “Lena” and “Peppers” in

Fig. 22 and Fig. 23. Since the quantitative quality of [12][14][15][16][36] and our method are better than others, we take them for visual comparisons only. The images restored by ACWM, ATMBM, and DRID have some obvious noise. The reconstructed image of our method can preserve more details in edges (see stalk of pepper) as shown in Fig. 22. Clearly, the proposed DTBDM produces visually pleasing images. The VLSI architecture of our design was implemented by using Verilog HDL. We used SYNOPSYS Design Vision to synthesize the design with TSMC’s 0.18μm cell library. The layout for the design was generated with SYNOPSYS Astro (for auto placement and routing), and verified by MENTOR GRAPHIC Calibre (for DRC and LVS checks). The synthesis results show that the DTBDM chip contains 21k gate counts. It works with a clock period of 5 ns and operates at a clock rate of 200 M Hz. The power consumption is 17.12 m W with 1.8-V supply voltage .Furthermore, DTBDM is also implemented on the Alter a Stratix II EP2S60F1020C5 FPGA board for verification. The operating clock frequency of DTBDM is 163.83 MHz (with1.29k logic elements). Since our design requires only simple operations (fixed bit-width adding, subtracting, or comparing), the denoised results of DTBDM with software program and with the VLSI circuit are identical. To evaluate the circuit, we compared DTBDM with four recent denoising chips [33]-[36]. They were realized with different VLSI implementations, so it is very difficult to compare our chip with them directly. Hence, we have implemented our design by using four different technologies respectively. Table 7 shows the detailed comparisons for various implementations in terms of the total number of logic elements, line buffer, and frequency. As demonstrated, DTBDM requires lower hardware cost and works faster than RVNP [33], AMF [34] and NAVF [35]. Although the proposed method requires more logic elements than [36], it can achieve higher frequency and produce better image quality when denoising an image corrupted by random-valued impulse noise.

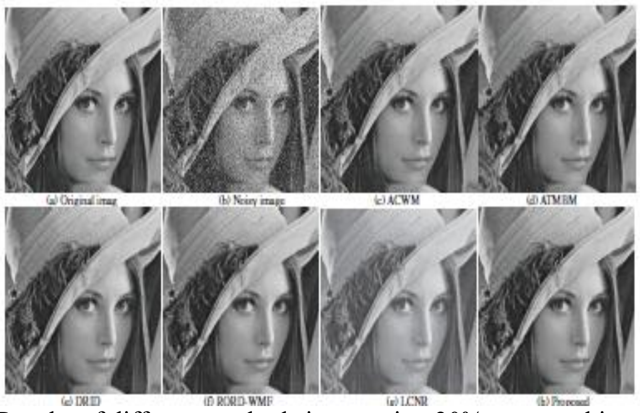


Fig. 21. Results of different methods in restoring 20% corrupted image “Lena”.

V. Conclusions

A low-cost VLSI architecture for efficient removal of random- valued impulse noise is proposed in this paper. The approach uses the decision-tree-based detector to detect the noisy pixel and employs an effective design to locate the edge. With adaptive skill, the quality of the reconstructed images is notable improved. Our extensive experimental results demonstrate that the performance of our proposed technique is better than the previous lower-complexity methods and is comparable to the higher-complexity methods in terms of both quantitative evaluation and visual quality. The VLSI architecture of our design yields a processing rate of about 200 MHz by using TSMC 0.18μm technology. It requires only low computational complexity and two line memory buffers. Therefore, it is very suitable to be applied to many real-time applications

Acknowledgement

Authors would like to thank all the researchers who have contributed in this field of research. The comments of anonymous reviewers to improve the quality of this paper are also acknowledged.

Reference

[1] R. Giral, L. Martinez-Salamero, and S. Singer, "Interleaved convertersoperation based on CMC," *IEEE Trans. Power Electron.*, vol. 14, no. 4,pp. 643–652, Jul. 1999.

[2] H. Kosai, S. McNeal, B. Jordan, J. Scofield, B. Ray, and Z. Turgut,"Coupled inductor characterization for a high performance interleaved boost converter," *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 4812–4815,Oct. 2009.

[3] C. A. Gallo, F. L. Tofoli, and J. A. C. Pinto, "A passive lossless snubberapplied to the AC–DC interleaved boost converter," *IEEE Trans. PowerElectron.*, vol. 25, no. 3, pp. 775–785, Mar. 2010.

[4] Y. Jang and M. M. Jovanovic, "Interleaved boost converter with intrinsicvoltage-doubler characteristic for universal-line PFC front end," *IEEETrans. Power Electron.*, vol. 22, no. 4, pp. 1394–1401, Jul. 2007.

[5] F. Musavi, W. Eberle, and W. G. Dunford, "A high-performance single-phasebridgeless interleaved PFC converter for plug-in hybrid electricvehicle battery chargers," *IEEE Trans. Ind. Appl.*, vol. 47, no. 4, pp. 1833–1843, Jul./Aug. 2011.

[6] C. A. Gallo, F. L. Tofoli, and J. A. C. Pinto, "Two-stage isolated switch modepower supply with high efficiency and high input power factor,"*IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3754–3766, Nov. 2010.

[7] M. O'Loughlin, "UCC28070 300-W interleaved PFC pre-regulator designreview," TI Appl. Rep. SLUA479B, Aug. 2008, revised Jul. 2010.

[8] C.-P. Ku, D. Chen, C.-S. Huang, and C.-Y. Liu, "A novel SFVM-M3control scheme for interleaved CCM/DCM boundary-mode boost converterin PFC applications," *IEEE Trans. Power Electron.*, vol. 26, no. 8,pp. 2295–2303, Aug. 2011.2303, Aug. 2011.

[9] Y. T. Chen, S. M. Shiu, and R. H. Liang, "Analysis and design of a zero-voltage-switching and zero-current-switching interleaved boost converter,"*IEEE Trans. Power Electron.*, vol. 27, no. 1, pp. 161–173, 2011.

[10] G. Yao, A. Chen, and X. He, "Soft switching circuit for interleaved boost converters," *IEEE Trans. Power Electron.*, vol. 22, no. 1, pp. 80–86, Jan.2007.

[11] B.-R. Lin, H.-K. Chiang, C.-Y. Tung, and C.-Y. Cheng, "Implementation of an interleaved ZVS boost-type converter," in *Proc. IEEE Int. Symp.Ind. Electron.*, Jul. 5–8, 2009, pp. 819–824.

[12] N. Jain, P. Jain, and G. Joos, "A zero voltage transition boost converter employinga soft switching auxiliary circuit with reduced conduction losses,"*IEEE Trans. Power Electron.*, vol. 19, no. 1, pp. 130–139, Jan. 2004.

[13] R. Streit and D. Tollik, "A high efficiency telecom rectifier using a novelsoft-switching boost-based input current shaper," in *Proc.Conf. Rec. IEEEINTELEC*, 1991, pp. 720–726.

[14] K. M. Smith and K. M. Smedley, "A comparison of voltage-mode softswitchingmethods for PWM converters," *IEEE Trans. Power Electron.*,vol. 12, no. 2, pp. 376–386, Mar. 1997.

[15] C.-J. Tseng and C.-L. Chen, "Novel ZVT-PWM converter with activesnubbers," *IEEE Trans. Power Electron.*, vol. 13, no. 5, pp. 861–869,Sep. 1998.

[16] G. Moschopoulos, P. Jain, G. Joos, and Y.-F Liu, "Zero voltage switchedPWMboost converter with an energy feed forward auxiliary circuit," *IEEETrans. Power Electron.*, vol. 14, no. 4, pp. 653–662, Jul. 1999.

[17] T.-W. Kim, H.-S. Kim, and H.-W. Ahn, "An improved ZVT PWM boostconverter," in *Proc. Conf. Rec. IEEE Power Electron. Spec. Conf.*, 2000,pp. 615–619.