

Edge-Preserving Algorithm for Block Artifact Reduction and Its Pipelined Architecture

Truong Quang Vinh and Young-Chul Kim

This paper presents a new edge-protection algorithm and its very large scale integration (VLSI) architecture for block artifact reduction. Unlike previous approaches using block classification, our algorithm utilizes pixel classification to categorize each pixel into one of two classes, namely smooth region and edge region, which are described by the edge-protection maps. Based on these maps, a two-step adaptive filter which includes offset filtering and edge-preserving filtering is used to remove block artifacts. A pipelined VLSI architecture of the proposed deblocking algorithm for HD video processing is also presented in this paper. A memory-reduced architecture for a block buffer is used to optimize memory usage. The architecture of the proposed deblocking filter is verified on FPGA Cyclone II and implemented using the ANAM 0.25 μm CMOS cell library. Our experimental results show that our proposed algorithm effectively reduces block artifacts while preserving the details. The PSNR performance of our algorithm using pixel classification is better than that of previous algorithms using block classification.

Keywords: Deblocking filter, offset filter, edge-preserving filter, post-processing, very large scale integration (VLSI) architecture.

Manuscript received May 20, 2009; accepted Jan. 25, 2010.

This work was supported by the Ministry of Education, Science Technology (MEST) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Regional Innovation and also partly supported by the Ministry of Knowledge Economy (MKE), Rep of Korea, under the ITRC support Program supervised by the NIPA (NIPA-2010-C1090-1011-0008).

Truong Quang Vinh (phone: +82 62 530 0369, email: tqvinh@soc.chonnam.ac.kr) and Young-Chul Kim (email: yckim@chonnam.ac.kr) are with the Department of Electronics and Computer Engineering, Chonnam National University, Gwangju, Rep. of Korea.
doi:10.4218/etrij.10.0109.0290

I. Introduction

Currently, video post-processing is mostly utilized for high-end display products to improve the quality of images. Due to the limitation of bandwidth or media storage, the video materials are compressed by using various international video-compression standards. Traditional transform-based video coders such as MPEG-1/2 [1], MPEG-4 [2], and recent video compression schemes such as H.264/AVC [3] are based on the block-based discrete cosine transform (BDCT). BDCT is a well-known and efficient transform method to achieve sparse representation of image blocks, and is thus successfully applied in compression applications. However, at low-bit-rate compression, BDCT-based coding techniques result in many noticeable artifacts including grid noise, staircase noise, and corner outlier noise. Therefore, the deblocking filter for block artifact reduction is an essential part in video post-processing systems.

In order to eliminate block artifacts, many deblocking methods have been proposed in the last decade. These can be divided into two categories: spatial domain and frequency domain. Spatial-domain methods can be based on spatial adaptive filters [4]-[7], statistical models [8], or projection onto convex sets (POCS) theory [9], [10]. Frequency-domain filtering methods employ discrete cosine transform (DCT) [11], [12] or discrete wavelet transform (DWT) [13], [14] to convert an image to frequency domain, analyze frequency coefficients of blocks and adjacent blocks, and then apply filtering to minimize the blocking effects.

From a hardware point of view, not many published algorithms are applicable for hardware implementation. Some proposed algorithms require heavy computation that makes the design cost very high [8], [11]-[14]. Other algorithms using

loop routines, such as POCS-based algorithms [9], [10], are only suitable for software because these loop routines require high delay which is not appropriate for real-time processing. Thus, most of deblocking filters use spatial-domain filtering methods which have low computational complexity.

Recently, classification-based deblocking method in the spatial domain has emerged as a promising approach due to low computational complexity, fast execution, and real-time ability. Jongho Kim and others [4] explicitly classify each deblocking block (DB) which has an image made of 8×8 squares, into two metrics: horizontal activity and vertical activity. A hard threshold is applied to the two metrics to give four categories: uniform DB, horizontal DB, vertical DB, and complex DB. For each type of DB, different filtering methods are used to reduce all kinds of block artifacts. Ling Shao and others [5] proposed a deblocking scheme using local entropy analysis. Each block is classified into detailed regions, flat regions, and intermediate regions based on entropy values. In the Amir Z. Averbuch's method [6], blocks are categorized into 2 classes: uniform block and nonuniform block. Deblocking of the uniform blocks is accomplished by applying three iterations of the weight adaptation by grading (WABG) scheme. Then, a fourth iteration is used in order to deblock the nonuniform blocks or detailed blocks. Since block artifacts are more visible in smooth regions, the authors' purpose is to apply strong smoothing to uniform regions and little or no smoothing on detailed regions. Therefore, the detail in the image can be preserved while block artifacts are reduced.

In all these proposed methods, the authors attempt to categorize blocks into classes. Then, different filtering techniques for each region are applied based on the classification characteristics of the blocks. The strong smoothing is applied on smooth regions, and weak smoothing is applied to textured regions. Therefore, the sharpness and detail of the image is preserved while the block artifact is reduced. However, the classification with fixed size blocks does not always give robust distinction between different regions. The drawback of these approaches is that fixed-size classified blocks may contain some smooth regions and edge regions with various shapes. For example, a block is classified into a smooth region when the intensity activity of the block is less than a predefined threshold. However, this block actually contains some edge pixels. When the deblocking filter applies strong smoothing to the whole of this block, these edge pixels can be destroyed. As a result, the deblocking filter may smooth inappropriate regions because of the fixed-size block classification. Figure 1 demonstrates the case when a block is classified into a smooth region, while containing some edge pixels.

In this paper, we propose a new pixel classification-based

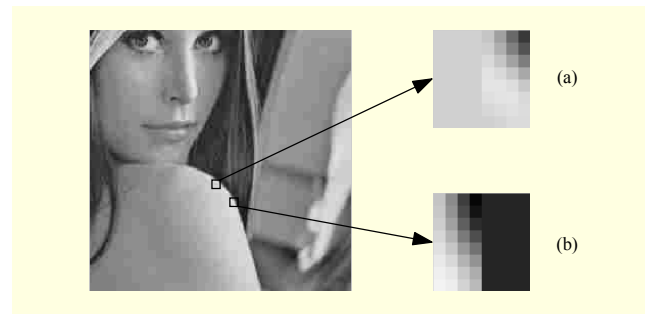


Fig. 1. (a) The block is classified into the smooth region, but it contains some edge pixels. (b) The block is classified into the edge region, but it contains some smooth pixels.

approach for block artifact reduction and also present an efficient architecture for its hardware implementation. In our proposed algorithm, we first categorize each pixel by an edge detection process instead of classifying each fixed-size block as in other techniques, and then generate three binary edge-protection maps for horizontal orientations, vertical orientations, and edge strength. Based on the horizontal and vertical maps, we apply an offset filter to reduce grid noise in a smooth region. In the next step, we employ an edge-preserving filter with an edge profile map to remove staircase noise and corner outlier noise. We also propose an efficient pipelined architecture for the algorithm. The hardware implementing architecture is designed practically for HD video real-time post-processing systems by using pipelined architecture and area-optimized block buffers to achieve the highest throughput as well as reduce memory usage. Our architecture is verified on Cyclone II FPGA and synthesized by using the ANAM 0.25 μm CMOS cell library.

The rest of this paper is organized as follows. In section II, we describe the proposed edge-protection deblocking algorithm. Section III presents VLSI architecture with detailed block diagrams of the deblocking filter. Section IV shows the experiment results including algorithm simulation and architecture verification, and the performance comparison with other approaches. Finally, we conclude in section V.

II. Proposed Edge-Protection Deblocking Algorithm

The aim of the proposed approach is to preserve the detail of images by using an edge-protection map while filtering blocky noise. The edge-protection map is generated by pixel classification. Two adaptive filters are then applied to process pixels with different weighted factors according to the edge-protection map.

1. Pixel Classification

Let's consider an 8×8 block as shown in Fig. 2. This block is

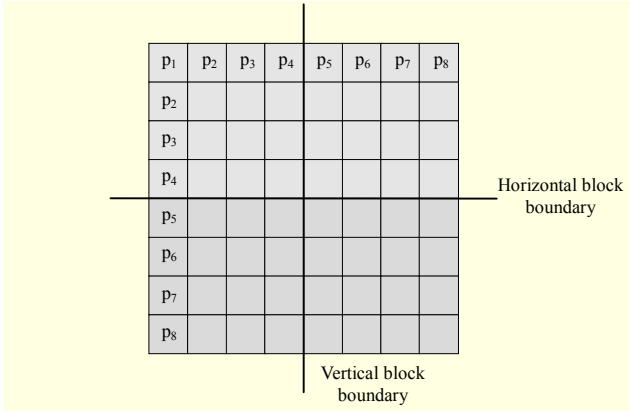


Fig. 2. Pixel location in a block.

not classified into a smooth or edge region as in some previous research. We classify every pixel in this block by an edge detection process. Various convolution kernels such as Sobel or Prewitt can be used for this operation. In this paper, for the purpose of real-time hardware implementation, we choose the Prewitt operator which needs low computation resources to estimate the magnitude and orientation of edges.

After applying the Prewitt operator for a 3×3 sliding window, we obtain the orientation magnitude G_x and G_y of the horizontal and vertical direction for each pixel. The magnitude, or *edge strength*, is then calculated using the formula

$$G = |G_x| + |G_y| \quad (1)$$

Three binary edge-protection maps are then created from the x -direction gradient, y -direction gradient, and edge strength. We apply the threshold T and T_d to generate three binary edge maps, E_x , E_y , and E_z , for the block. In the experiment, the threshold T and T_d are set to 20 and 10, respectively;

$$\begin{aligned} E_{x_{i,j}} &= \begin{cases} 0, & |G_x| < T_d, \\ 1, & |G_x| \geq T_d, \end{cases} \\ E_{y_{i,j}} &= \begin{cases} 0, & |G_y| < T_d, \\ 1, & |G_y| \geq T_d, \end{cases} \\ E_{z_{i,j}} &= \begin{cases} 0, & G < T, \\ 1, & G \geq T. \end{cases} \end{aligned} \quad (2)$$

These maps indicate that a pixel belongs to a smooth region or an edge region if its binary value in these maps is zero or one, respectively. Therefore, the adaptive filter can use these maps to protect the detail of images.

2. Offset Filter for Grid Noise

After obtaining binary edge-protection maps for the block, we apply 1-D offset filtering for all pixels in the block to reduce blocky noise. The offset filter processes horizontally first, and

then, vertically. The procedure of 1-D offset filtering is

$$\begin{aligned} offset_i &= p_{i,4} - p_{i,5}, \\ p_{i,1} &= p_{i,1} - offset_i / 16 \times \overline{Ex_{i,1}}, \\ p_{i,2} &= p_{i,2} - offset_i / 8 \times \overline{Ex_{i,2}}, \\ p_{i,3} &= p_{i,3} - offset_i / 4 \times \overline{Ex_{i,3}}, \\ p_{i,4} &= p_{i,4} - offset_i / 2 \times \overline{Ex_{i,4}} - offset_i / 4 \times Ex_{i,4}, \\ p_{i,5} &= p_{i,5} + offset_i / 2 \times \overline{Ex_{i,5}} + offset_i / 4 \times Ex_{i,5}, \\ p_{i,6} &= p_{i,6} + offset_i / 4 \times \overline{Ex_{i,6}}, \\ p_{i,7} &= p_{i,7} + offset_i / 8 \times \overline{Ex_{i,7}}, \\ p_{i,8} &= p_{i,8} + offset_i / 16 \times \overline{Ex_{i,8}}, \end{aligned} \quad (3)$$

where p_{ij} with $j, i=1:8$ is the intensity value of the pixel at the position (i, j) in the 8×8 block as shown in Fig. 1, and $\overline{Ex_{i,j}}$ is the inversion of $Ex_{i,j}$, that is, if $Ex_{i,j}$ is zero, $\overline{Ex_{i,j}}$ is one.

In (3), we define “offset” as the amount of discontinuity at the block boundary. This value is calculated from the difference of the two nearest pixels at the block boundary. As with some previous methods [4], [5], we also use shifted offset values (that is offset/2, offset/4, and offset/8) to eliminate discontinuity at the block boundary. The pixels on the left ($p_{i,1}, p_{i,2}, p_{i,3}, p_{i,4}$) and pixels on the right ($p_{i,5}, p_{i,6}, p_{i,7}, p_{i,8}$) are updated by the way to balance the gap at the block boundary. Since block artifacts are more visible on a smooth region, we employ the oriented edge map to make the filter process only pixels in smooth regions except for two pixels at the block boundary $p_{i,4}$ and $p_{i,5}$. If these pixels are in the edge region, they are updated with the coefficient offset/4. The edge map E_x is applied for horizontal filtering, and E_y is applied for vertical filtering.

After this step, the blocky noise in the smooth region can be reduced, but staircase noise and corner outlier noise still remain in edge regions. A post-processing filter should be used to remove these noises.

3. Edge-Preserving Filter for Staircase Noise and Corner Outlier Noise

We propose an edge-preserving filter to remove staircase noise and corner outlier noise. Our approach is based on the fundamental idea that the bilateral filter is able to smooth along the edge and thus enhance the detail of images [15]. Consider a 3×3 spatial sliding window

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}, \quad (4)$$

where $x_i, i=1, \dots, 9$ is an intensity value, and x_5 is a to-be-filtered

pixel. The intensity distances from the center pixel x_5 are defined by

$$d_i = |x_i - x_5|, \quad i = 1, \dots, 9. \quad (5)$$

The coefficients for the convolution mask of the filter are extracted as follows:

$$c_i = (255 - d_i)^q. \quad (6)$$

The coefficient c_i receives larger values for smaller values of intensity distance d_i . That is, pixels which are very different in intensity from the central pixel are weighted less than pixels that are close intensity from the central pixel. This concludes to the following convolution mask:

$$C = \frac{1}{\sum_{i=1}^9 c_i} \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}. \quad (7)$$

Similar to a bilateral filter whose coefficients are weighted based on intensity similarity with Gaussian distribution, the proposed edge-preserving filter uses coefficients depending on intensity distances with an exponent function. Therefore, the convolution mask allows creating a smooth effect along an edge and thus can remove staircase noise and corner outlier noise. The factor q is to control the amount of edge smoothing and was chosen for the experiment. In our experiment, q is set to 8. The denominator $\sum c_i$ is a normalization factor to maintain zero-gain for the output image. The output of the edge-preserving filter is computed by applying spatial filtering with the coefficient mask C and the sliding window X .

The edge-preserved filtering is only applied for pixels in edge regions in order to remove staircase noise and corner outlier noise. Therefore, the binary edge map E_z is used in the filtering procedure as shown in (8).

$$p_{i,j} = p_{i,j} \times \overline{E_{z_{i,j}}} + edge_filter(X) \times E_{z_{i,j}}. \quad (8)$$

In summary, our proposed algorithm for block artifact reduction is described in the following procedure:

For each 8×8 block :

- i) establish three binary maps, E_x , E_y , and E_z , for edge protection,
- ii) apply the horizontal and vertical offset filters to pixels of the block according to procedure (3),
- iii) apply the edge-preserving filter to pixels of the block according to (8).

III. Efficient Architecture for Block Artifact Reduction

The proposed architecture is designed to perform real-time processing, so a pipeline technique is employed. Parallel

processing is used for the offset filter and edge-preserving filter. An 8-bit fixed-point numerical representation for pixel intensity is chosen. HDTV 1080p with 30 Hz frame rate processing is a target constraint, so the maximum resolution of the input image can be $1,920 \times 1,080$ pixels. Therefore, the required throughput is $1,920 \times 1,080 \times 30 \approx 75.10^6$ pixels/s. The design core processes 1 pixel per 2 clock cycles. Hence, the required operation clock is 150 MHz.

1. Top Block Design

The deblocking core is comprised of five basic functional units: a block buffer, a pixel classifier, an offset filter, an edge-preserving filter, and a flow controller. The top design architecture is shown in Fig. 3.

The proposed deblocking filter requires input data pixel by pixel. The block buffer converts raster scan input pixels into blocks of 8×8 pixels for the offset filter, and blocks of 3×3 pixels for the pixel classifier. After being applied to the offset filter, the 8×8 pixel blocks are converted to 3×3 pixel blocks for the edge-preserving filter. The control unit implemented by a finite state machine is responsible to control data flow. The delay unit realized by FIFO memory is to keep synchronization for pipeline stages.

2. Memory-Reduced Block Buffer

In the proposed architecture, the pixel classifier accepts 3×3 pixel blocks as the input, the offset filter receives 8×8 pixel blocks, and the edge-preserving filter requires 3×3 pixel input blocks. Therefore, we need two types of block buffer for 3×3 pixel blocks and 8×8 pixel blocks. For raster scan input format as shown in Fig. 4(a), conventional buffers normally used for image processing hardware [16] utilize 8 line-buffers to create 8×8 pixel blocks as shown in Fig. 5(a). Therefore, large FIFO memory buffers are required if the input image has HD resolution.

In order to reduce memory usage for block buffers, we propose a new block buffer using modified raster format as shown in Fig. 4(b). The new block buffer receives pixels in vertical order and generates 8×8 pixel arrays for the offset filter as in Fig. 5(b).

The new buffer architecture only requires 8-byte FIFO memory to store temporal pixels. The total memory for the new block buffer is 120 ($7 \times 16 + 8$) bytes. Therefore, we can save 99% memory usage of the block buffer if the input data are 1080p image frames, with the cost of extra hardware for address generator to access image memory in vertical raster scan order, while the conventional block buffer requires 13,448 ($7 \times 1920 + 8$) bytes.

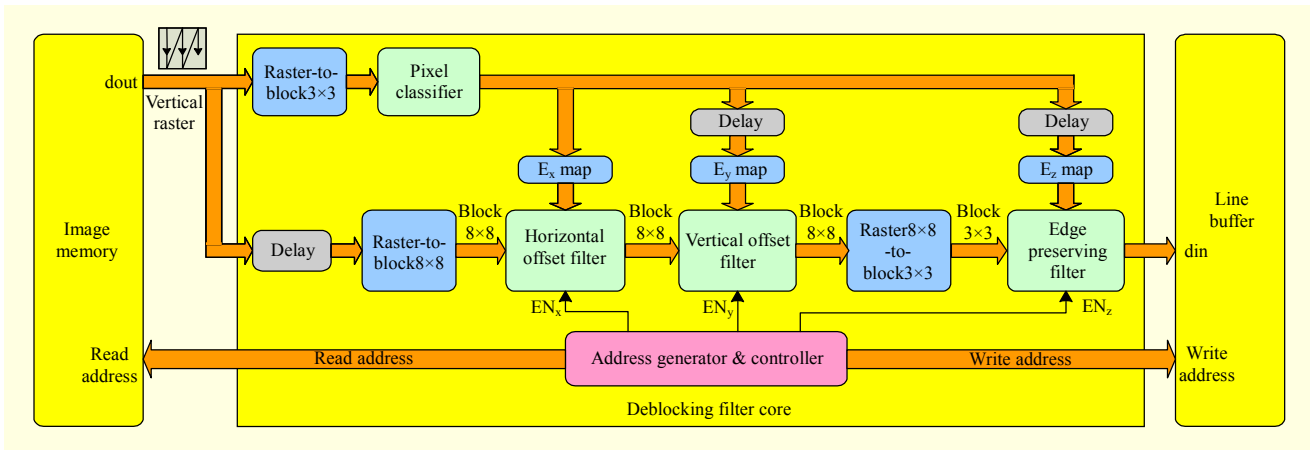


Fig. 3. Block diagram of the deblocking filter.

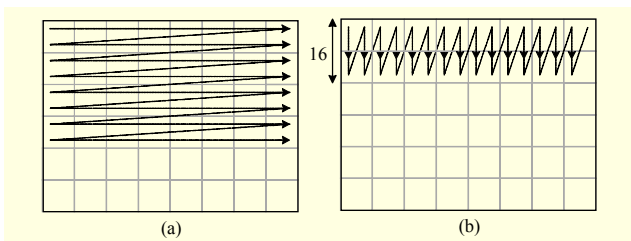


Fig. 4. (a) Conventional raster scan order and (b) modified raster scan format.

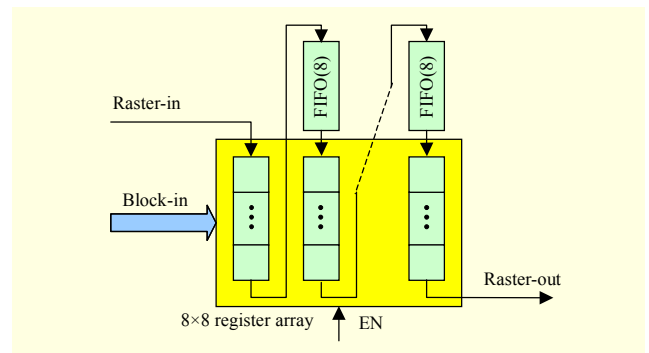


Fig. 6. Architecture of the 8x8 block-to-raster unit.

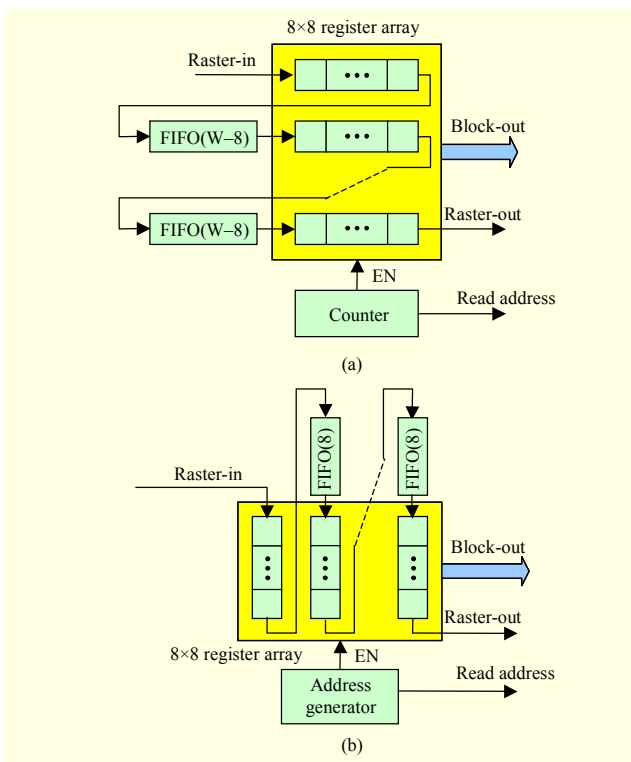


Fig. 5. (a) Conventional 8x8 block buffer using horizontal raster scan format (W is the width of images) and (b) memory-reduced 8x8 block buffer using vertical raster scan format.

The raster-to-3x3block unit is similar to the 8x8 block buffer. It includes a 3x3 register array and FIFO memory with the length 13 (16-3). The 8x8block-to-3x3block unit is designed by combining the 8x8block-to-raster and the raster-to-3x3block units. The 8x8block-to-raster unit is shown in Fig. 6.

3. Pixel Classification Unit

The pixel classification unit comprises a Prewitt operator and thresholding unit. We apply 3-stage pipeline architecture for pixel classification unit (Fig. 7). The pixel classification unit is able to generate three binary values, E_x , E_y , and E_z , for each pixel in every clock cycle.

4. Offset Filter

The design of the offset filter unit is based on a shifting parameter to reduce the hardware complexity. The offset filter receives two types of input data including a pixel vector and an edge map vector. Each clock cycle, the offset filter processes 8 pixels in vector format. Hence, it takes 8 clock cycles for the offset filter to complete an 8x8 pixel block.

Two identical offset filtering units are used to allow simultaneous on-the-fly filtering for vertical and horizontal

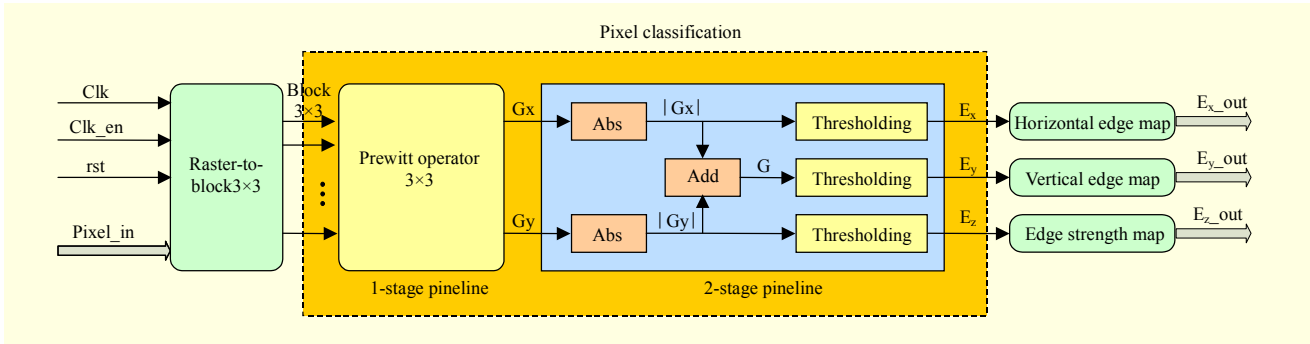


Fig. 7. Architecture of the pixel classification unit.

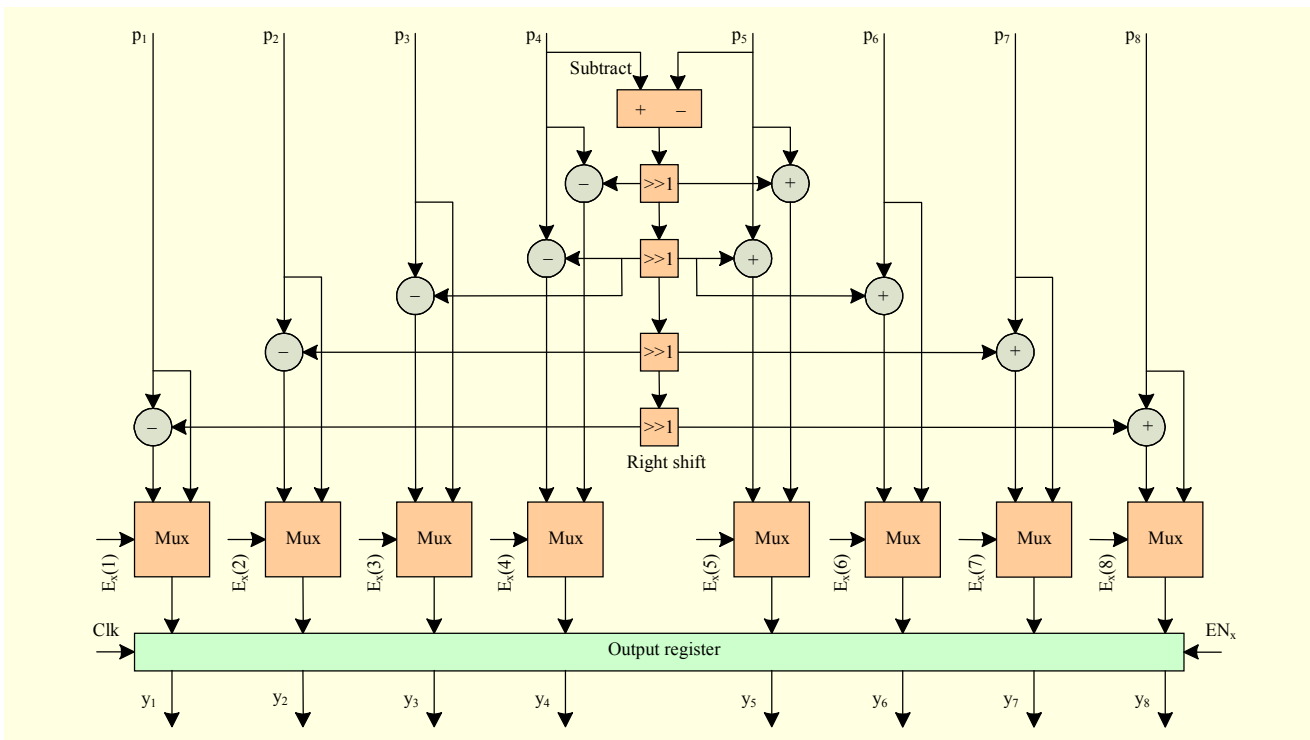


Fig. 8. Architecture of the offset filter block.

blocky noise. However, there is a difference between two these offset filters in the input data format. The horizontal offset filter receives the horizontal pixel vector and horizontal edge-protection map as input data, while the vertical offset filter accepts the vertical pixel vector and vertical edge-protection map as input data. The block diagram of the proposed architecture is shown in Fig. 8. The output results of the filter are registered by flip-flops for the pipeline stages.

5. Edge-Preserving Filter

The edge-preserving filter receives a 3×3 pixel block and an E_z map as input data. The convolution mask of the input 3×3 block is calculated by three steps: intensity distance computation, coefficient extraction, and coefficient

normalization. Then spatial filtering is performed by multiplying the convolution mask by the input neighbor pixel values.

The block architecture of the edge-preserving filter is shown in Fig. 9. The critical path includes 4 multipliers, 2 subtractors, 1 absolute, 1 adder, 1 multiplexer, and a divider. To avoid long delays for the critical path, a 12-stage pipelined architecture is proposed for this unit (Fig. 9). The multiplier is designed with 8-bit architecture, and the output product is a 16-bit value. Therefore, we take 8 high bits of the product value to be the input for the next stage of computation. The precision is slightly decreased, but it is acceptable for visual quality.

6. Address Generator and Controller

The address generator is to generate an addressing counter in

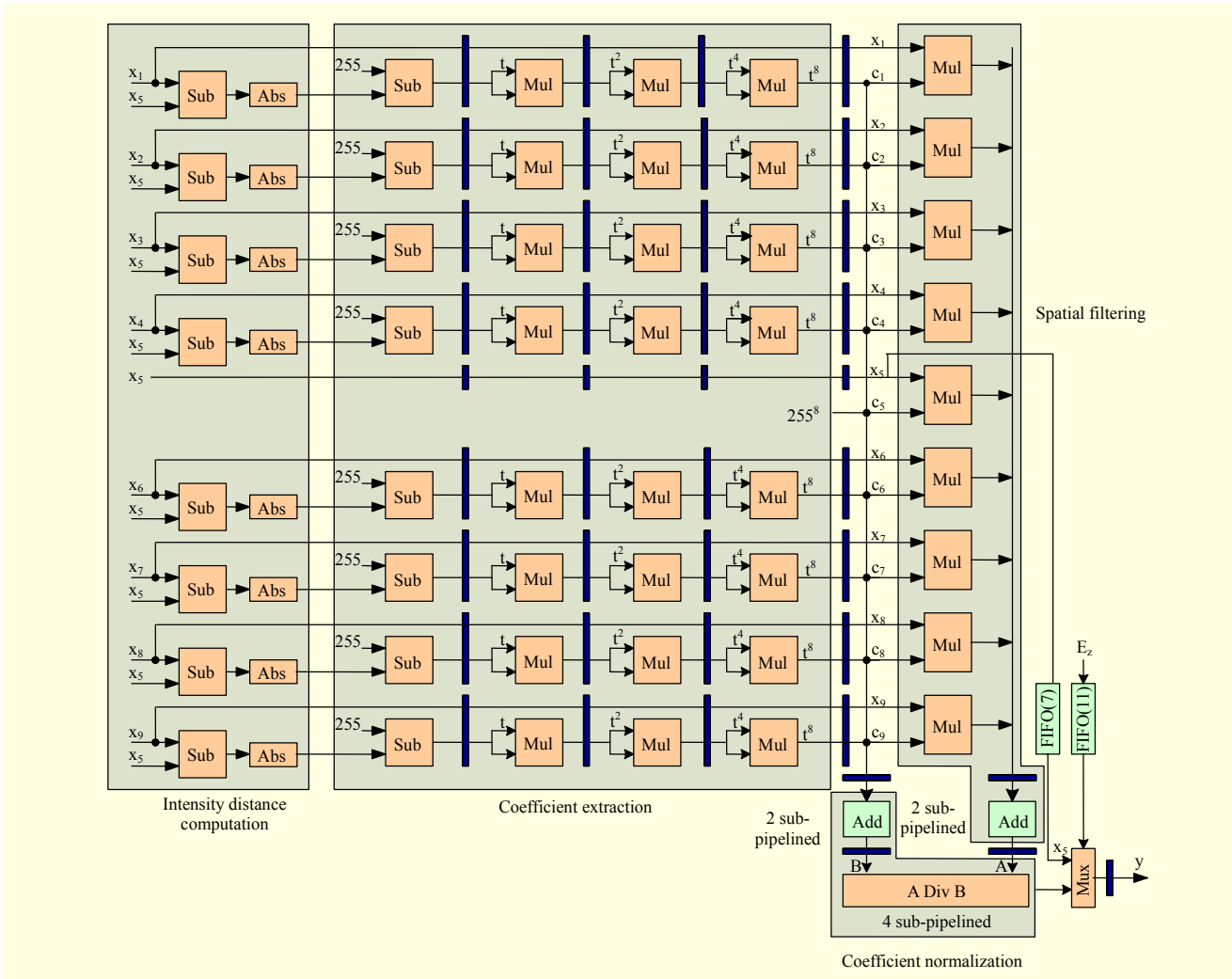


Fig. 9. Full parallel pipelined architecture for edge-preserving filter.

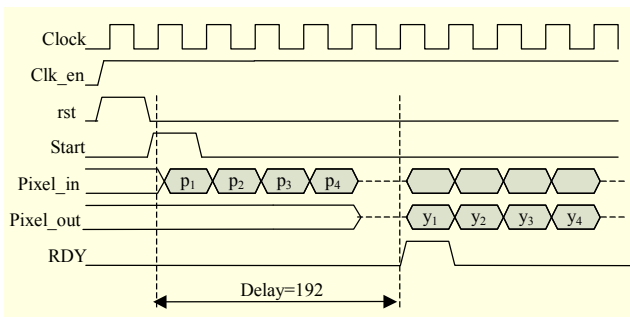


Fig. 10. Timing diagram for the deblocking filter.

vertical raster scan format. The address generator receives two input data: width size and height size of the input frame. We designed the address generator with a finite state machine (FSM) to count address values according to the raster scan as shown in Fig. 4(b).

The control unit is also realized by an FSM to control the enable signals, EN_x , EN_y , and EN_z , for offset filter and edge-

preserving filter. This FSM activates the enable signals when the 8×8 sliding blocks for the offset filter and the 3×3 sliding blocks for the edge-preserving filter are available at inputs of each filter unit. Four rows and four columns of pixels at the boundary of frames are not processed by the offset filter because of the lack of information on neighbor pixels.

The timing diagram of the deblocking filter core is shown in Fig. 10. The total number of pipeline stages of the design is 192, so the delay between input and output is 192 clock cycles. The first result comes out after several clock cycles, which is indicated by RDY signal.

IV. Experiment Results

In order to verify and evaluate the algorithm and its architecture for the deblocking filter, experiments for both algorithm simulation and VLSI architecture verification have been performed. The results from the reference model,

behavior simulation, and FPGA verification were compared and evaluated together. In our experiment, we first simulated the algorithm by Matlab 7.4.0, modeled the architectures in VHDL, simulated the waveform on ModelSim 5.6, and verified the design on FPGA platform. Finally, we synthesized the design by using 0.25 μm ANAM CMOS cell library.

1. Algorithm Simulation

For the algorithm simulation on Matlab, we first experimented with a set of test images including Lena, Barbara, and Peppers. All the test images are 512 \times 512 with 8-bit grayscale resolution. PSNR metric was used to evaluate and compare the results with some previous deblocking schemes. To evaluate the performance of our proposed algorithm, we compared our method with some other deblocking techniques using spatial-domain classification-based approaches [4]-[7]. The algorithm in [4] proposed by J. Kim and others classifies pixel blocks by computing pixel activities and then applying offset-and-shift technique. The algorithm in [5] proposed by L. Shao and others uses local entropy analysis for block classification and low-pass filters for block artifact reduction in different regions. The algorithm in [6] proposed by A.Z. Averbuch and others classifies pixel blocks into uniform and nonuniform, then applies iterations of the WABG scheme for uniform blocks and a single iteration of the deblocking frames of variable size (DFOVS) method for nonuniform blocks. The algorithm in [7] proposed by T. Chen and others uses the sum of AC coefficient energy to classify blocks into a low- or high-activity class, then applies the adaptive filter for different classes of blocks. All these methods use fixed-size block classification before applying deblocking filtering. Table 1 shows the PSNR comparative results for different images at various bit rates using some previous algorithms and our proposed method. Figure 11 shows the deblocked images of grayscale Lena512 by applying some deblocking methods. As in the experiment result, our proposed method achieves the highest PSNR values for most images and bit rates.

In the second test, we used test sequences “Foreman,” “News,” and “Silent.” The test sequences were compressed by MPEG-4 XviD encoder whose configuration was set to the simple @L3 profile, single pass encoding type, and 100, 200, and 300 kbps bit rates. Figure 12 shows the 15th frame of sequence “Foreman” before and after applying the proposed deblocking algorithm. Table 2 shows performance comparisons of our algorithm and previous methods for test sequences.

2. Hardware Implementation

The proposed architecture of the deblocking filter was implemented in VHDL at RTL level, simulated in ModelSim 5.8,

Table 1. PSNR comparison of different deblocking methods.

Image	Bit rate (bpp)	PSNR (dB)					
		JPEG	Our method	Ref. [4]	Ref. [5]	Ref. [6]	Ref. [7]
Lena 512	0.25	30.04	31.48	31.37	30.91	31.01	31.22
	0.36	32.95	33.62	33.48	32.88	32.85	33.37
	0.46	34.26	34.60	34.62	34.52	33.70	34.51
Peppers 512	0.25	30.18	31.29	31.19	30.62	30.74	30.75
	0.37	32.47	33.24	33.06	32.43	32.45	32.72
	0.47	33.61	34.14	34.02	33.12	33.22	33.72
Barbara 512	0.44	26.99	27.40	27.38	26.67	27.11	27.23
	0.54	28.25	28.55	28.55	27.71	28.08	28.23
	0.69	30.16	30.25	30.28	30.10	29.33	29.73

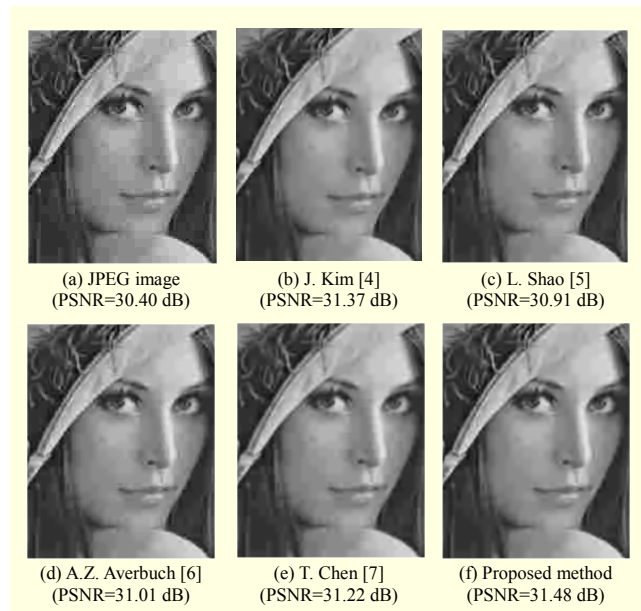


Fig. 11. Deblocking test image grayscale Lena512 using some previous approaches and the proposed method.

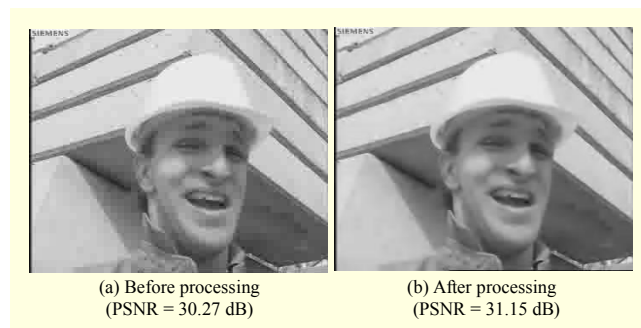


Fig. 12. The 15th frame of the 100 kbps sequence “Foreman” before and after applying the proposed deblocking algorithm.

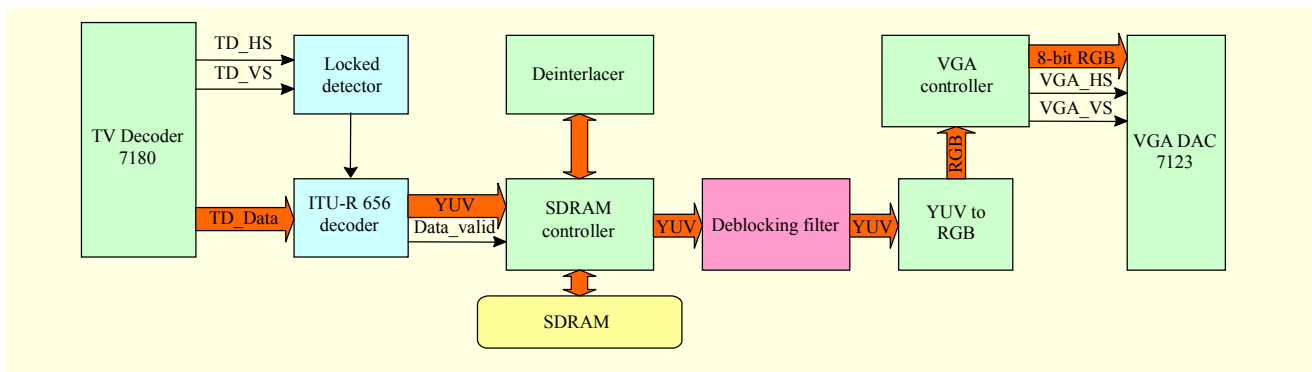


Fig. 13. Verification model for deblocking filter.

Table 2. PSNR improvement for test sequences.

Test sequences	Bit rate (kbps)	Average PSNR (dB)	Average PSNR improvement (dB)			
			[4]	[5]	[7]	Our method
Foreman (CIF, 300frs)	100	27.64	0.612	0.511	0.554	0.707
	200	27.71	0.592	0.494	0.562	0.698
	300	27.84	0.561	0.465	0.550	0.689
News (CIF, 300frs)	100	28.10	0.360	0.425	0.401	0.562
	200	28.17	0.333	0.419	0.382	0.549
	300	28.32	0.284	0.410	0.371	0.528
Silent (CIF, 150frs)	100	28.13	0.542	0.498	0.525	0.616
	200	28.19	0.525	0.484	0.506	0.601
	300	28.48	0.476	0.439	0.470	0.571

Table 3. Synthesis result of proposed deblocking filter.

Features	Ref. [17]	Ref. [18]	Our design
Algorithm	In loop filter	In loop filter	Edge-preserving filter
PSNR improvement	0 to 1 dB	0 to 1 dB	0.2 to 1.5 dB
Pipeline	Non	Non	192 stage
Process (μ)	0.25	0.18	0.25
Cycle/MB	615	250	512
Frequency required for HD at 30fps (MHz)	300	120	150
Memory (bytes)	640	30,720	310
Gate count (k)	20.66k	19.64k	50.1k

and verified on FPGA Cyclone II. For architecture simulation, we used the library std.textio to read and write pixel data from and to a file. The result file was then compared with the simulation result from Matlab. The FPGA verification was realized on Kit DE2-70 using FPGA EP2C70F896 at 50 MHz for VGA image frames.

The verification model is shown in Fig. 13.

Finally, the proposed architecture was synthesized with 0.25 μ m ANAM CMOS cell library by Synopsys Design Compiler. The constraint delay of the critical path is set to 6.7 ns for HD video processing. Therefore, this design can work at a clock frequency up to 150 MHz and perform deblocking filtering for 1080p video sequences. Our design was successfully synthesized with the constraints.

Table 3 shows the comparison between our design and other designs using an in-loop deblocking filter for H.264/AVC. Our algorithm is more complex than an in-loop deblocking filter. Therefore, the hardware implementation requires more resources than a conventional deblocking filter. However, the PSNR improvement of our design is higher, thus the visual quality of the output images is better.

V. Conclusion

In this paper, we proposed an edge-protection deblocking algorithm and also presented its efficient VLSI architecture for a deblocking filter. Being different from previous block classification based approaches, the proposed method classifies each pixel into smooth or edge regions by an edge detection process and generates edge-protection maps. Based on these maps, two types of filters, including an offset filter and edge-preserving filter, are applied to remove grid noise, staircase noise, and corner outlier noise. Our experiments on some standard test images show that the proposed method can reduce block artifacts more effectively than some previous approaches while preserving the detail of the image.

The hardware architecture for the proposed deblocking algorithm is implemented. The pipeline technique is applied, and parallel structure for the offset filter and edge-preserving filter is used to achieve the highest throughput for real-time video processing. Memory-reduced architecture for the block buffer is employed to minimize memory usage. The synthesis results show that the architecture can process HD resolution

video frames with a 150 MHz operation clock. Furthermore, our hardware implementation can be integrated in HD video post processing systems.

References

- [1] ISO/IEC 13818-2, "Generic Coding of Moving Pictures and Associated Audio Information," Part 2: Video, Nov. 1994.
- [2] MPEG Video Group, "MPEG-4 Video Verification Model Version 8.0," ISO/IEC JTC1/SC29/WG11 N1796, July 1997.
- [3] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification," (ITU-T Rec. H.264—ISO/IEC 14496-10 AVC), May 2003.
- [4] J. Kim, M. Choi, and J. Jeong, "Reduction of Blocking Artifacts for HDTV Using Offset-and-Shift Technique," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, Nov. 2007, pp. 1736-1743.
- [5] L. Shao and I. Kirenko, "Coding Artifact Reduction Based on Local Entropy Analysis," *IEEE Trans. Consum. Electron.*, vol. 53, no. 2, May 2007, pp. 691-696.
- [6] A.Z. Averbuch, A. Schlar, and D.L. Donoho, "Deblocking of Block-Transform Compressed Images Using Weighted Sums of Symmetrically Aligned Pixels," *IEEE Trans. Image Process.*, vol. 14, no. 2, Feb. 2005, pp. 200-212.
- [7] T. Chen, H.R. Wu, and B. Qiu, "Adaptive Postfiltering of Transform Coefficients for the Reduction of Blocking Artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, May 2001, pp. 594-602.
- [8] Z. Li and E.J. Delp, "Block Artifact Reduction Using a Transform-Domain Markov Random Field Model," *IEEE Trans. Cir. Sys. Video Technol.*, vol. 15, no. 12, Dec. 2005, pp. 1583-1593.
- [9] J.J. Zou and H. Yan, "A Deblocking Method for BDCT Compressed Images Based on Adaptive Projections," *IEEE Trans. Cir. Sys. Video Technol.*, vol. 15, no. 3, Mar. 2005, pp. 430-435.
- [10] G.R. Kwon et al., "An Efficient POCS-Based Post-Processing Technique Using Wavelet Transform in HDTV," *IEEE Trans. Cons. Elec.*, vol. 51, no. 4, Nov. 2005, pp. 1283-1290.
- [11] Y. Zhao, G. Cheng, and S. Yu, "Postprocessing Technique for Blocking Artifacts Reduction in DCT Domain," *Electron. Lett.*, vol. 40, no. 19, Sept. 2004, pp. 1175-1176.
- [12] Y. Luo and R.K. Ward, "Removing the Blocking Artifacts of Block-Based DCT Compressed Images," *IEEE Trans. Image Process.*, vol. 12, no. 7, Jul. 2003, pp. 838-842.
- [13] H. Choi and T. Kim, "Blocking-Artifact Reduction in Block-Coded Images Using Wavelet-Based Subband Decomposition," *IEEE Trans. Cir. Sys. Video Technol.*, vol. 10, no. 5, Aug. 2000, pp. 801-805.
- [14] N.C. Kim et al., "Reduction of Blocking Artifact in Block-Coded Images Using Wavelet Transform," *IEEE Trans. Cir. Sys. Video Technol.*, vol. 8, no. 3, June 1998, pp. 253-257.
- [15] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and color Images," *Proc. IEEE Int. Conf. Comput. Vision*, 1998, pp. 59-66.
- [16] C.T. Johnston, K.T. Gribbon, and D.G. Bailey, "Implementing Image Processing Algorithms on FPGAs," *Proc. Eleventh Electronics New Zealand Conf.*, Palmerston North, New Zealand, Nov. 2004, pp. 118-123.
- [17] Y.W. Huang et al., "Architecture Design for Deblocking Filter in H.264/ JVT/AVC," in *Proc. IEEE Int. Conf. Multimedia Expo*, MD, 2003, pp. I-693-I-696.
- [18] T. M. Liu et al., "A Memory-Efficient Deblocking Filter for H.264/AVC Video Coding," *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, 2005, pp. 2140-2143.



Truong Quang Vinh received the BE in electronics engineering from Ho Chi Minh University of Technology, Vietnam, in 1999, and the ME in computer science from the Asian Institute of Technologies, Bangkok, Thailand, in 2003. He is now a PhD candidate at Chonnam National University, Gwangju, Korea. His research interests include digital circuits, SoC, embedded system design, and video post-processing.



Young-Chul Kim received his PhD from Michigan State University, USA, the MS from the University of Detroit, USA, and the BS in electronics engineering from Hanyang University, Korea. In 1993, he joined the Department of Electronics Engineering at Chonnam National University (CNU) where he is currently a professor. From 2000 to 2004, he was a director of IDEC at CNU. From 2004 to 2005, he was a Vice Dean of the College of Engineering in this university. Since 2004, he has become the chief of the LG Innotek R&D center at CNU. His research interests are SoC design and development using IPs and low power design.