FPGA Implementation of Real-Time Edge-Preserving Filter for Video Noise Reduction

Truong Quang Vinh, Ju Hyun Park, Young-Chul Kim, Sung Hoon Hong
Department of Computer Engineering
Chonnam National University, Gwangju, Korea
tqvinh@soc.chonnam.ac.kr

Abstract

In this paper, VLSI architectures and FPGA implementation for edge-preserving filter are presented. We proposed two architectures for edge-preserving filter: full parallel pipelined and structure-shared architectures. The edge-preserving filter uses adaptive coefficient mask based on the intensity distance in filter blocks. Compared with the bilateral filter, the proposed edge-preserving filter provides significantly noise reduction. We implement the proposed architecture on Cyclone II EP2C70F896C8 FPGA device from Altera Corp. Our experiments show that the PSNR improvement is up to 5.3 dB for Gaussian noisy images.

1. Introduction

The demand for post-processing to enhance the visual quality of the reconstructed video is now increasing much. For low-bit rate compress videos or noisy pictures, noise removal is an essential process in video enhancement. The denoising function requires a simple but effective algorithm to adapt the real-time implementation. Especially for HDTV post-processing systems, the hardware architectures for processing functions should be considered for delay time in the critical path. Thus, denoising algorithm for these systems prefer to be non-iterative, low complex while preserving good performance.

The conventional method for noise reduction is linear smooth filter, but it significantly blurs the edge of an image. Therefore, many edge-preserving filtering algorithms have been proposed [1, 2, 4]. One of the well-known edge-preserving filters is the bilateral filter proposed by Tomachi et al. [1]. This filter extends the concept of Gaussian smoothing by weighting the filter coefficients with the corresponding relative pixel intensities. Pixels that are very different in intensity from the central pixel are weighted less than pixels that are close intensity from the central pixel. As a non-iterative and simple filtering technique, bilateral filtering has received considerable attention in areas of image processing and computer vision. Unlike the conventional linear filters of which the coefficients are pre-determined, the bilateral filter adjusts its coefficients to the geometric closeness and photometric similarity of the pixels. Therefore, this filter has good applicability at edge-preserving smoothing for image restoration applications, such as noise reduction and digital coding artifacts reduction [3].

To apply the edge-preserving filter for noise removal in video post-processing, we proposed a simple and non-iterative algorithm for hardware implementation based on the fundamental idea of bilateral filter. Two architectures for this algorithm are presented for FPGA implementation: full parallel pipelined and structure-shared architectures. The full parallel architecture provides fastest processing but requires most resources; while the structure-shared architecture is optimized for the resource usage but has low throughput.

The paper is organized as follows. Section 2 presents the definition and equation of bilateral filter. Section 3 shows our modified edge-preserving filter based on the idea of bilateral filter. In Section 4, we describe the VLSI architecture for edge-preserving filter. The experiment results are shown in Section 5. Finally, we make the conclusion in Section 6.

2. Bilateral filter

Bilateral filter is a nonlinear filter that smoothes the noise while preserving edge of an image. The shift-variant filtering operation of the bilateral filter is given by:

$$\hat{I}(p_0) = C^{-1} \sum_{p \in N(p)} e^{\frac{-\|p-p_0\|^2}{2\sigma_d^2}} \, e^{\frac{-|I(p)-I(p_0)|^2}{2\sigma_r^2}} \, I(p), \quad (1)$$

where σ_d and σ_r are parameters to determine the level of smoothes. If σ_r is set to zero, the bilateral filter is reduced to a simple Gaussian smoothing filter. N(p) is a spatial neighborhood of pixel p_0, and

C is the normalization constant to maintain zero-gain for the output image.

$$C = \sum_{p \in N(p)} e^{\frac{-\|p-p_0\|^2}{2\sigma_d^2}} e^{\frac{-|I(p)-I(p_0)|^2}{2\sigma_r^2}} \quad (2)$$

The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain. Two pixels can be close to one another, that is, occupy nearby spatial location, or they can be similar to one another, that is, have nearby values, possibly in a perceptually meaningful fashion.

3. Modified edge-preserving filter

Based on the fundamental idea of the bilateral filter, we propose a modified edge-preserving filter to remove video noise. Our approach is to make the algorithm simple and implementable for real-time processing hardware. This filter can smooth along edge and thus enhance the detail of images. Consider a 3x3 spatial window of the input image, intensity distances from the center pixel is calculated:

$$d_i = p_i - p_5 \quad (3)$$

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

$$c_i = (255 - d_i)^p \quad (4)$$

That is, c_i receives larger values for smaller values of d_i. Similar to bilateral filter, the coefficients is changed in exponent rate with the spatial distance between neighbor pixels and the center pixel. This concludes to the following convolution mask:

$$\frac{1}{\sum_{i=1}^{9} c_i} \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix} \quad (5)$$

The convolution mask creates smooth effect along edges and thus can remove noise while not blurring the edges. The factor p is to control the amount of edge smoothing performed on the center pixel of the 3x3 window. The fixed value 8 is set for factor p.

4. VLSI architecture for modified edge-preserved filter

We propose two architectures for modified edge-preserving filter: full parallel pipelined architecture and share-resource architecture. Our target is to apply edge-preserving filter into noise deduction for real-time video post-processing. The filter block processes a 3x3 pixel block and then produces a pixel at the output. Pipelined architecture is employed to improve the throughput.

The full parallel pipelined architecture is shown in Fig. 1. This architecture provides the highest throughput, but it consumes the maximum resources. The hardware resources required include: 16 subtractors, 2 adders, 32 multipliers, and 1 divider. The critical path includes 5 multipliers, 2 subtractors, and a divider. To avoid the long delay for critical path, an 11-stage pipelined architecture is proposed.

The second architecture for edge-preserving filter is shown in Fig. 2. This design uses structure-shared architecture to reduce resource usage. The required resource can be reduced up to 30% compared to the parallel structure. The filter block needs 10 clock cycles to filter one pixel with a 3x3 sliding window. Due to the high delay time for processing, we design more pipelined stages for the structure-shared architecture. Thus, 19-stage pipelining is applied. The delay for the first coming sample is 29 (19+10) clock cycles. The detail specification for both architectures is described in the Table 1.

In both proposed architectures, we use 8-bit precision for hardware computation. For 8-bit multiplier, the product is a 16-bit value. Therefore, we take the high significant byte of the product for the next computation. The precision is slightly decreased, but it is acceptable for visual quality.

Table 1. The specification of proposed architectures for edge-preserving filter

Specification	Full parallel architecture	Structure-shared architecture
Processing type	parallel	serial
Number of Pipeline stages	11	19
Resource usage	high	low
Delay for one sample	1 cycle/ sample	10 cycles/ sample
Delay for first coming sample	11 clock cycles	29 clock cycles
Capability of video processing	720p60, 1080p60 XGA, WXGA	480p60, SVGA

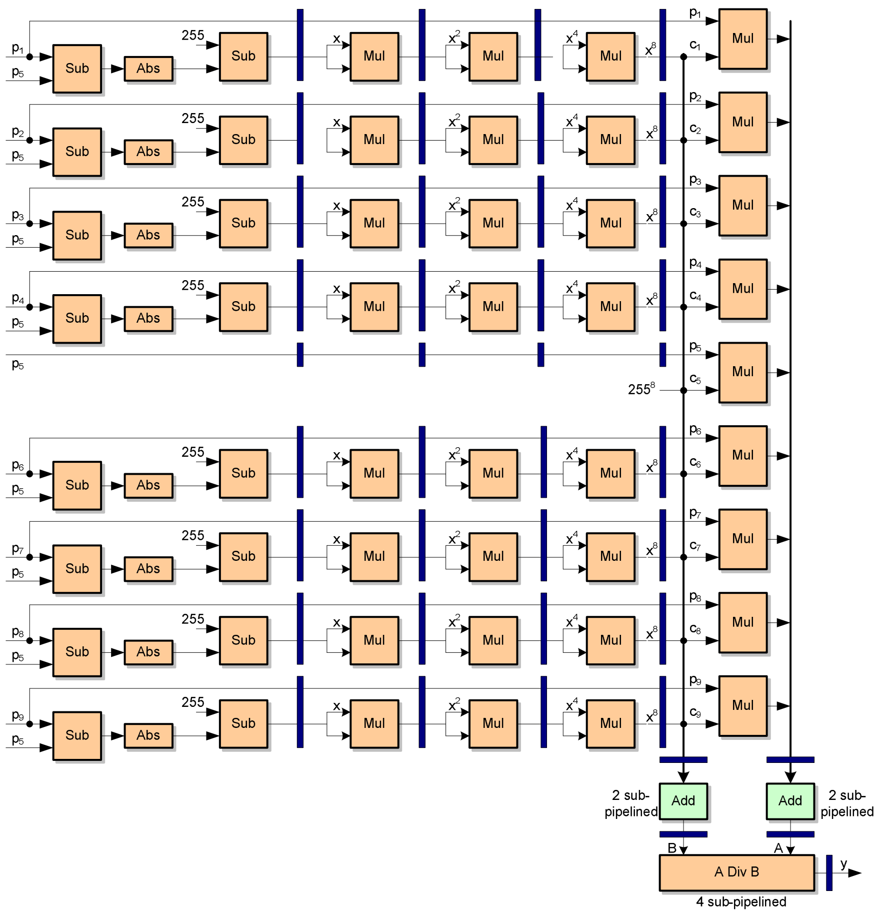

Figure 1: Full parallel pipelined architecture for edge-preserving filter

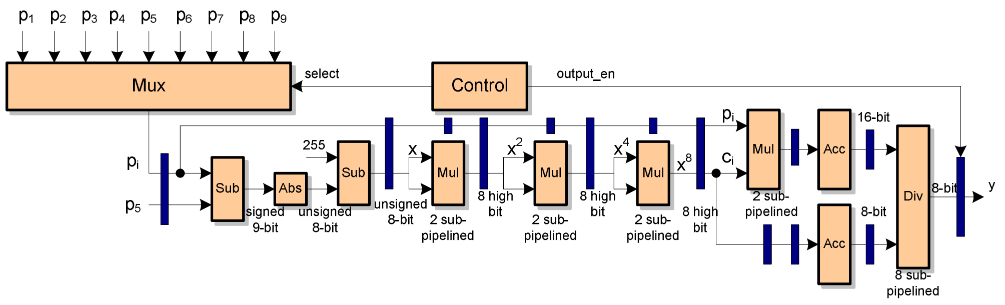

Figure 2: Structure-shared architecture for edge-preserving filter.

5. Experiment results

The proposed edge-preserving filter is at first simulated by Matlab 7.4.0. Then we describe the architectures of the proposed filter by VHDL and simulate on ModelSim 5.6. The Quartus II 8.0 tool is used to synthesize the design. The FPGA platform used for verification is facilitated by Cyclone II EP2C70F896C6 device.

For the algorithm simulation on Matlab, we use a set of test images including: Lena, Barbara, and Goldhill images. All the test images are gray scale with size 512x512. The PSNR comparison between proposed algorithm and bilateral filter is shown in Table 2. This result shows that our modified edge-preserving filter achieves higher performance than the original bilateral filter.

For verification, we do experiment on Altera DE2-70 Kit using Cyclone II EP2C70F896C6. The synthesis report produced by Quartus II software is shown in Table 3. The experiment proves that the design is capable for real-time processing.

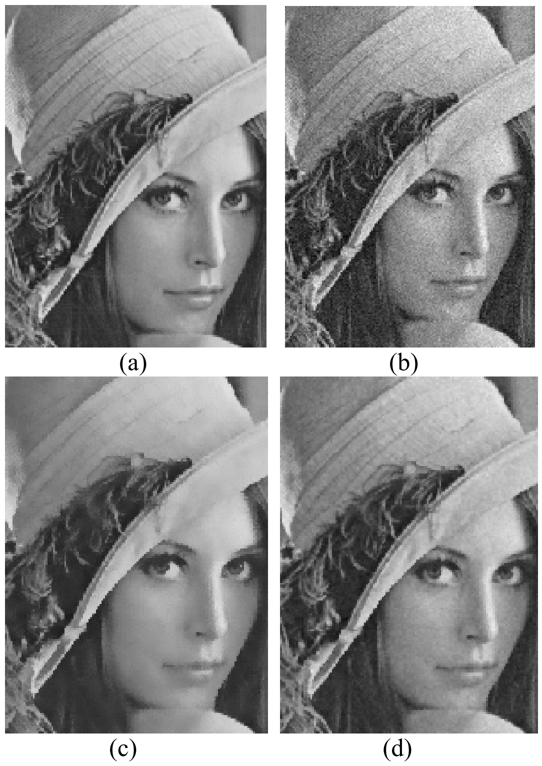

Figure 3. (a) Original image (b) Noisy image PSNR (c) Bilateral filter (d) Proposed method

Table 2. The PSNR comparison for edge-preserving filter

Input image	σ	Noise image	Bilateral filter [2]	Proposed method
Lena	10	28.12	31.37	33.45
	20	22.13	27.02	27.10
Barbara	10	28.12	30.53	30.90
	20	22.19	26.23	26.81
Goldhill	10	28.13	31.17	32.22
	20	22.15	27.03	26.09

Table 3. The synthesis result for edge-preserving filter

Architecture	Full parallel architecture	Share-structure architecture
FPGA Device	EP2C70F896C6	EP2C70F896C6
Total logic elements	567	450
Total logic register	227	398
Multiplier elements	32	4
Maximum path delay	6.3 ns	4.4 ns
Maximum frequency	159 MHz	226.8 MHz

6. Conclusions

We have presented a new architecture for edge-preserving filter. Compared with bilateral filter, our edge-preserving filter provides better performance in noise reduction. The experiments show that our filter can achieve up to 5.3 dB improvement for some test noisy images. The maximum operation frequency for full parallel architecture is 159 MHz, which is sufficient for HDTV real-time processing.

7. References

[1] S. M. Smith and J. M. Brady, "SUSAN—A new approach to low level image processing," Int. J. Comput. Vision, 1997, vol. 23, pp. 45–78.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images", in Proc. IEEE Int. Conf. Comput. Vision, 1998, pp. 59–66.

[3] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," IEEE Trans. Pattern Anal. Mach. Intell., Jun., 2002, vol. 24, no. 6, pp. 844–847.

[4] M. Elad, "On the origin of the bilateral filter and ways to improve it," IEEE Trans. Image Process., Oct. 2002, vol. 11, no. 10, pp. 1141–1151.

[5] C. Charoensak, and F. Sattar, "FPGA Design of a Real-Time Implementation of Dynamic Range Compression for Improving Television Picture", ICICS 2007, pp. 210-214.

[6] B. Zhang, and J. p. Allebach, "Adaptive Bilateral Filter for Sharpness Enhancement and Noise Removal", ICIP 2007, pp. IV-417 – IV-420.

[7] M. Zhang, and B. Guntrk, "A New Image Denoise Method Based on The Bilateral Filter", ICASSP 2008, pp. 929-932.