International Scholarly Research Network
ISRN Signal Processing
Volume 2012, Article ID 805346, 6 pages
doi:10.5402/2012/805346

Research Article

Sample-Level Filtering Order for High-Throughput and Memory-Aware H.264 Deblocking Filter

Guilherme Correa,[1] Luciano Agostini,[2] and Luis A. da Silva Cruz[1]

[1] *Instituto de Telecomunicações (IT) University of Coimbra—FCTUC, Polo II Universidade de Coimbra, Pinhal de Marrocos, 3030-290 Coimbra, Portugal*
[2] *Group of Architectures and Integrated Circuits (GACI) Federal University of Pelotas, UFPel Campus Universitário s/n, P.O. Box. 354, 96010-900 Pelotas, RS, Brazil*

Correspondence should be addressed to Guilherme Correa, guilherme.correa@co.it.pt

Received 5 March 2012; Accepted 10 April 2012

Academic Editors: E. Ciaccio and P. Yin

This paper presents a new sample-level filtering order for the Deblocking Filter process of the H.264/AVC video coding standard to be used instead of the traditional block-level order presented in previous works. This processing order allows a better exploration of the parallelism in the filtering process by reducing data dependencies in comparison to other works. The proposed sample-level order allows four parallel and independent samples filtering simultaneously, completing one complete macroblock filtering in fewer cycles and requiring less memory space than the related works. The proposed filtering order can be applied to the Deblocking Filter presented in a conventional H.264/AVC encoder or decoder and to the H.264/SVC interlayer Deblocking Filter. When compared to the original H.264/AVC filter and to the best related work found in the literature, the proposed scheme achieves a reduction of 72% and 25% in the number of clock cycles and a memory usage decrease of 75% and 43%, respectively.

1. Introduction

The quality of a video may be treated in both objective and subjective fields. In the first case, distortion metrics are used in order quantify the quality of the video through the calculation of the signal-to-noise ratio between the original video signal and the final video signal. The most widely used distortion metric is called PSNR (peak signal-to-noise ratio). On the other hand, the subjective quality of a video is much more difficult to be tested or measured. Since it depends on how the image is perceived by a viewer, its measurement is more expensive in terms of time and human resources. Several video coding standards provide tools that deal with both objective and subjective video quality. However, the H.264/AVC standard [1] defines as mandatory the use of a module responsible for increasing the image subjective quality. This module, called Deblocking Filter [2], increases the image subjective quality through smoothing operations over the pixels. These operations are performed in order to decrease the perception of some block-shaped artifacts introduced in the frame by a high quantization level. This is an adaptive filter, since it can distinguish between real borders of an image and the borders introduced by high quantization steps.

The Deblocking Filter of H.264/AVC is placed at the end of the coding process, as shown in Figure 1, where ME, MC, T, Q, T^{-1}, and Q^{-1} stand for Motion Estimation, Motion Compensation, Transform, Quantization, Inverse Quantization, and Inverse Transform, respectively. The information coded through interframe or intraframe prediction is added to the residuals and the resulting content is then filtered by the Deblocking Filter. After that, the image can be stored in order to be used during the coding of the next frame (in case of interframe coding) or during the coding of the next blocks of the same image (in case of intraframe coding).

The H.264/AVC standard supports scalability through its Annex G, called Scalable Video Coding (SVC) extension [3], whose operating principle defines a hierarchy of layers where a base layer represents the video signal at a given minimum temporal, spatial, and/or quality resolution, and additional

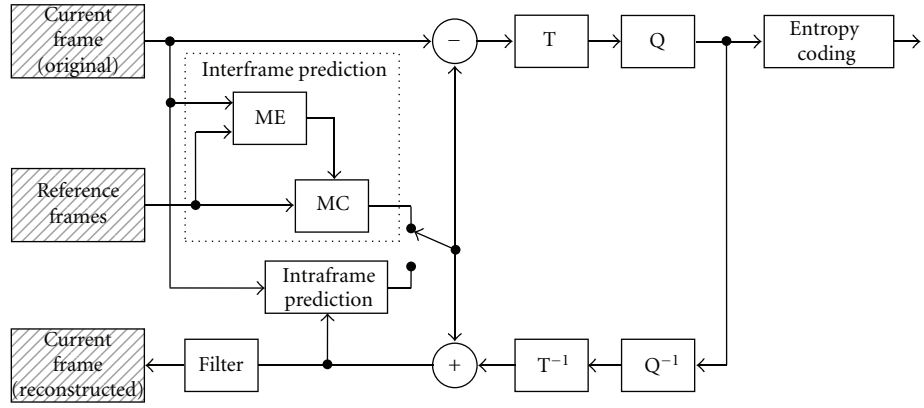

Figure 1: Block diagram of the H.264/AVC encoder.

layers are used to increase each one of the resolutions. In the scalable version of H.264/AVC (herein called H.264/SVC), the Deblocking Filter is also used between two coding (or decoding) layers in order to smooth the block-shaped artifacts introduced in the reference layer before they are used in the enhancement layer.

Due to their high complexity, a wide research has been done regarding the implementation of both H.264/AVC and SVC Deblocking Filters. The main source of their complexity can be attributed to the fact that each pixel must be read several times in different directions to filter a complete macroblock. To deal with this problem, several processing orders were proposed in previous works, all of them aiming at the decrease of the time and the amount of memory used in the filtering process.

Even though most of these processing orders can decrease the computational time needed to filter a complete frame, they are still based on block level operations, limiting the exploration of parallel computation. We have proposed in this work a new and optimized filtering order for the H.264 Deblocking Filter based on sample level operations. This novel filtering order can be applied to both H.264/AVC and H.264/SVC versions of the standard. It allows a better exploration of the parallelism level, reducing significantly the use of cycles to perform the filtering process in comparison with the previous solutions presented in the literature.

The paper is structured as follows: in Section 2 we outline the operation of both the H.264/AVC and H.264/SVC Deblocking Filters. Section 3 exposes the filter ordering solutions published in the technical literature. In Section 4, our novel proposed solution is presented and explained. Section 5 reports the results and compares them to related works. Section 6 concludes with a discussion about the current results and shows promising directions for future work.

2. H.264/AVC and H.264/SVC Deblocking Filters

As previously mentioned, the H.264/AVC Deblocking Filter can distinguish a real border in an image (which shall not be filtered) from an artifact generated by a high quantization

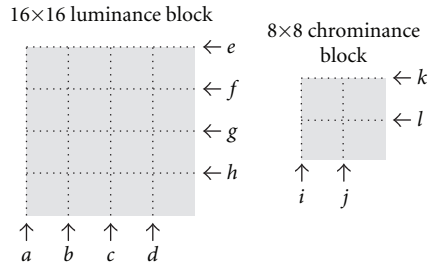

Figure 2: Filtering ordering of the luminance and chrominance borders.

step (which must be filtered) [2]. The filter is applied to the vertical and horizontal borders of each 4×4 luminance and chrominance blocks, as shown in Figure 2. At first, the four vertical borders of the luminance component are serially filtered (*a*, *b*, *c*, and *d*, in Figure 2). After that, the four horizontal borders of the luminance component are serially filtered (*e*, *f*, *g*, and *h*, in Figure 2). The same process is then applied to the chrominance component (*i* and *j* vertical borders and *k* and *l* horizontal borders in Figure 2).

Considering a luminance macroblock, each filter operation uses up to four samples from each of the two neighboring blocks (p and q) that are filtered (p_0, p_1, p_2, p_3, q_0, q_1, q_2, and q_3, where the index values define how far is the sample from the border) and can modify the values of up to three samples of each side (p_0, p_1, p_2, q_0, q_1, and q_2). The chrominance samples are filtered in a similar way, but the values of just five samples are used in the calculations (p_0, p_1, q_0, q_1, and q_2) and only two samples are modified during the filtering process (p_0 and q_0).

Before the filtering process starts, some parameters that are used in the decisions of the filtering process are computed. One of them, the filter strength, is adjustable and depends on (1) the quantization step (QP) used when the block was coded, (2) the coding mode of neighboring blocks, and (3) the pixels gradient values across the edge that is being filtered. The H.264/AVC standard specifies five different strengths parameterized as *bS*, which takes values 0, 1, 2, 3, or 4 (0 standing for "no filtering" and 4 indicating

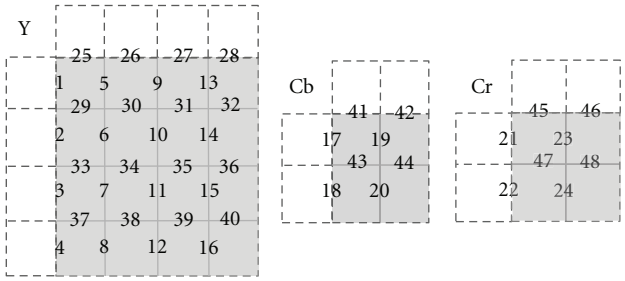

Figure 3: Original H.264/AVC filtering order [1].

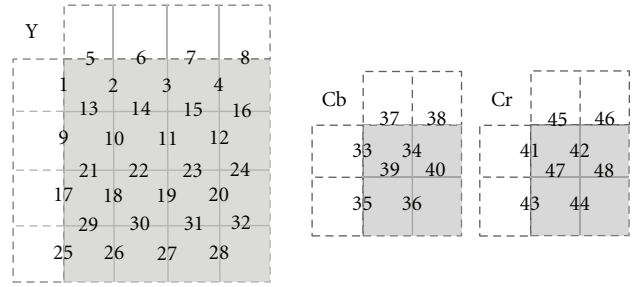

Figure 4: Filtering order proposed by Khurana [4].

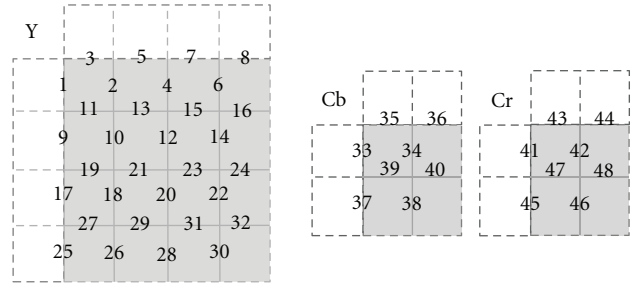

Figure 5: Filtering order proposed by Sheng [5].

maximum smoothing). The algorithm used to define the *bS* value is not the same in H.264/AVC and H.264/SVC, and this is the unique difference between the implementation of the two filters. The focus of this work is on the processing order of the filter, which is not dependent on the standard version and can be used in both of them.

3. Previously Proposed Filtering Orders

To filter a macroblock, the value of a pixel must be read several times and the intermediate results of the filterings must be locally stored, since they are used in the following steps. In order to improve the use of the local memory, it is possible to reorder the filtering operations in such a way that the intermediate results are used sooner by the Deblocking Filter. The only restriction imposed by the standard in relation to the processing order specifies that all the horizontal filterings that use a determined sample must occur before the vertical filterings that use this sample.

The processing order proposed by the H.264/AVC standard is presented in Figure 3. The numbers inside Figure 3 indicate the order of borders filtering. In Figure 3, the vertical borders of the luminance and chrominance blocks are all filtered before the horizontal borders. Since the results of the vertical filterings are used in the horizontal filtering, these intermediate results must be stored. Then, this processing order is extremely expensive in terms of memory use, since it demands the storage of 24 4×4 blocks (16 luminance blocks and 8 chrominance blocks) until the horizontal filtering occurs.

The filtering order proposed by Khurana et al. [4], presented in Figure 4, is based on an alternation between the horizontal and vertical filterings of the blocks. This solution results in a decrease of the local memory size, since just one line of 4×4 blocks must to be stored in order to be used by the next filtering steps. When the pixels are completely filtered (i.e., in both directions), they can be written back to the main memory in order to be shown or to be used as reference in the future.

A third processing order proposed by Sheng et al. [5], presented in Figure 5, is based on the same alternation principle proposed by Khurana [4]. However, in this case, the frequency of change between horizontal and vertical borders (and vice versa) is higher, occurring after most of the 4×4 border filterings. As a result, there is a higher decrease in the size of the used local memory.

The proposal of Li et al. [6] is based on both data reuse and concurrence principles. The processing order, shown in Figure 6, reduces significantly the number of cycles needed to filter a macroblock through the execution of horizontal and vertical filterings at the same time. For that, two filters are needed, one for each filtering direction. Repeated numbers in Figure 6 represent borders which are filtered simultaneously (each one by a different filter), in the proposed filtering order.

Similarly to Li et al. [6], Jing et al. [7] developed a design with two edge-filters that process the vertical and horizontal edges in parallel. The difference between them is that Jing's order [7], presented in Figure 7, starts the parallel processing in the fourth state instead of the third. This difference allows the use of both filters continuously, differently from Li's order [6], which performs only vertical filterings in stages 5, 10, 15, 20, and 21.

Through the use of other different techniques, Zhou et al. [8] and Y. C. Lin and Y. L Lin [9] reduce the number of cycles used to filter a macroblock. Zhou et al. [8] proposes a schedule for simultaneously processing luminance and chrominance edges. By following the same filtering order, both the luminance and chrominance edges are filtered with the same *bS*. For that, the filter operation is altered in order to allow the filtering of two edges at the same time. In the first two cycles, the two lines that compose the first chrominance block (Cb) and the two first lines of the 4×4 luminance block are fed into the edge filter. Then, in the next two cycles, the other chrominance block (Cr) and the two remainder luminance lines follow.

The proposal of Y. C. Lin and Y. L Lin [9] is based in three techniques: optimized filtering order, parallelism,

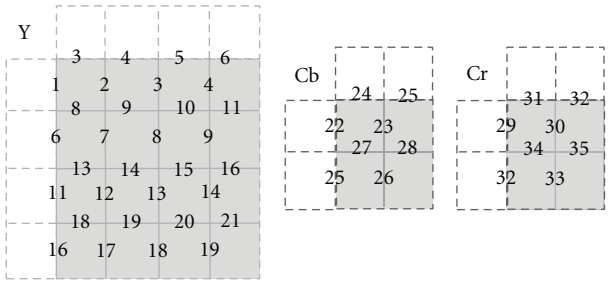

Figure 6: Filtering order proposed by Li et al. [6].

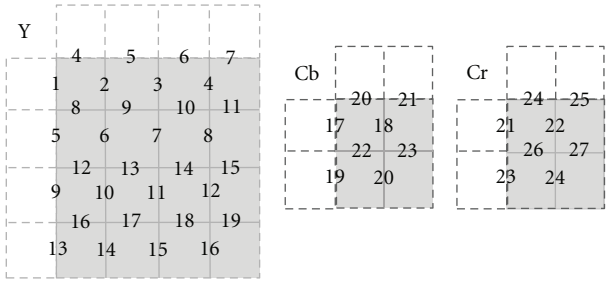

Figure 7: Filtering order proposed by Jing et al. [7].

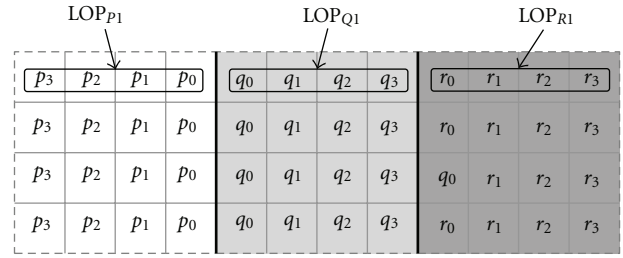

Figure 8: Sample-level concurrent filtering.

Table 1: Minimum frequency needed to filter videos in different spatial resolutions.

Resolution	F_{minimum}
QCIF (176 × 144 pixels)	157 KHz
QVGA (320 × 240 pixels)	477 KHz
VGA (640 × 480 pixels)	1.91 MHz
Full HD (1920 × 1088 pixels)	12.97 MHz
Quad HD (3840 × 2160 pixels)	51.52 MHz
Ultra HD (7680 × 4320 pixels)	206.06 MHz

and elimination of unnecessary filtering. The technique that presents most of the gains in terms of clock cycle reduction is the elimination of unnecessary filtering. It is based on the fact that two special macroblock modes, skip-top mode and skip-all mode, are very likely to occur in an interpredicted frame. This way, Y. C. Lin and Y. L Lin [9] states that a macroblock coded in skip-top mode has its *bS* set to zero at its top edge. In the same way, a macroblock coded in skip-all mode has all its *bS* values set to zero. Thus, no filtering and no access to external memory are needed.

Also based on Li's proposal [6], a concurrent architecture that uses four filtering cores is presented by Ernst [10]. However, as in all the previous block-based orders presented, the number of concurrent filterings is still very limited due to the data dependencies between the blocks. In order to solve this problem, we propose an alternative processing order based on sample-level filterings, as presented in the next section.

4. Proposed Filtering Order

All the processing orders presented in the previous section are performed in block level, that is, the filtering of a 4 × 4 block edge is performed serially by the same filter and the border of a block can be filtered only after the filtering of the 4 LOPs (*Line of Pixels*) of the previous block is complete (left or top block, in the case of horizontal and vertical filtering, resp.).

Since the filtering between the current block and the next one (right or bottom block) depends on the values of the filtering between the previous (left or top) block and the current one, it is not necessary to complete the filtering of the whole block edge. Then, the new processing order is based on sample-level filterings instead of block-level filterings so that the filtering of the first LOP of a block can start as soon as the result of the first LOP of the previous block is available.

For instance, in Figure 8, when the first filtering between LOP$_{P1}$ and LOP$_{Q1}$ is finished, the filtering between LOP$_{Q1}$ and LOP$_{R1}$ can start, even before the filterings between the other three LOPs of blocks *P* and *Q* are concluded. This approach facilitates the execution of parallel filterings, since the data dependency is treated in a smaller granularity level.

The processing order oriented to samples allows a better use of parallelism reducing the number of cycles used to filter a macroblock and decreasing significantly the use of local memory. Figure 9 shows the final proposed filtering order. The repeated numbers correspond to the filterings that can occur in parallel, in the same cycle, by different filter cores. In the first cycle, only one filtering can occur, since the filtering of all the next LOPs depends on the filtered result of the first LOP. In the second and the third cycles, two and three filterings can occur in parallel, respectively. From the fourth cycle on, four filterings can occur at the same time, thus demanding four filters in concurrent and continuous operation.

The luminance macroblock (Y macroblock in Figure 9) filtering is succeeded by the blue chrominance macroblock filtering (Cb macroblock in Figure 9) and by the red chrominance macroblock filtering (Cr macroblock in Figure 9), completing the whole macroblock filtering in 53 cycles.

As the proposed filtering order reduces significantly the number of cycles needed to filter a complete macroblock, this is an interesting solution to process high-resolution videos. Table 1 shows different video resolutions and the minimum frequency that an architecture should achieve in order to process real-time video coding (30 frames per second). As one can notice in Table 1, due to its reduced number of cycles, the filtering order can be applied in very low operation

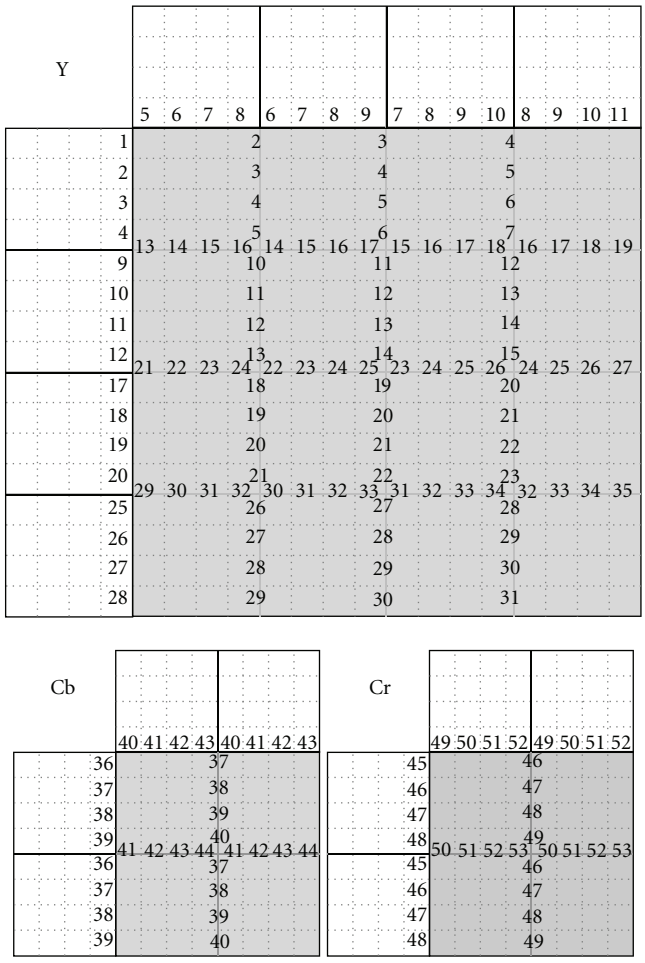

Figure 9: Filtering order proposed in this work.

frequencies for all the resolutions extensively used nowadays in order to reduce the power and the energy consumed in portable devices.

5. Results and Comparisons

Considering T as the time necessary to filter one border sample in all presented filtering orders, the lowest possible time for all filtering orders to process one macroblock is presented in Table 2. Besides, the local memory size needed for each implementation is shown in the right column of Table 2.

When compared with the filtering order proposed by the H.264/AVC standard [1], the presented solution is able to reduce the processing time in 72%. When compared to the best previous work in terms of clock cycles [10], which presents the same number of simultaneous filterings of this work, the proposed filtering order reduced in 25% the number of used cycles.

In terms of used memory, the proposed filtering order demands 128 bytes, which is 75% less than the original filtering order proposed by the H.264/AVC standard. Among the related works, the only filtering order that overcomes this work in terms of memory use is the one proposed in [6]. However, the number of cycles to filter a complete macroblock is almost three times higher in this case.

Table 2: Comparison between previous filtering orders and this work.

Author	Time	Simultaneous filterings	Memory size
H.264/AVC standard et al. [1]	$48*4T = 192T$	1	512 bytes
Khurana et al. [4]	$48*4T = 192T$	1	128 bytes
Sheng et al. [5]	$48*4T = 192T$	1	256 bytes
Li et al. [6]	$35*4T = 140T$	2	112 bytes
Jing et al. [7]	$27*4T = 108T$	2	640 bytes
Zhou et al. [8]	$32*4T = 128T$	2	579 bytes
Lin et al. [9]	$24*4T = 96T$	2	416 bytes
Ernst et al. [10]	$75T$	4	224 bytes
This work	$53T$	4	128 bytes

6. Conclusions

This paper presented a novel filtering order based on sample-level filterings, on the opposite of the previous works found in the technical literature, which are based on block-level filterings. The proposed filtering order allows a better and deeper exploration of the parallelism, since the sample-level approach decreases the amount of data dependency between two filterings.

The proposed processing order takes 53 cycles to complete a macroblock filtering. When compared to the order proposed by the H.264/AVC video coding standard [1], our proposal presents a reduction of 72% in terms of used cycles per macroblock. Also, when compared to the best related work found in the literature [10], our proposal still presents a reduction of 25% in terms of cycles. This high reduction in number of clock cycles allows our filtering order to be applied in real-time coders/decoders for all the video resolutions currently used in industry and academy. For example, the minimum frequency needed to filter a Full HD video in real time by the proposed order is around 13 MHz, which means that it can be used in devices that require low energy consumption.

It is important to notice that, although this filtering order presents a good gain in terms of processing time, it does not increases the size of the local memory used, since it stores at most eight blocks at the same time. In comparison to the proposals found in the literature, our filtering order requires less memory than most of them and only some bytes more than one of the proposals. Besides, even though the processing order is different, the basic filtering operations are still the same as the original filtering order, which means that no differences are noticed in terms of bit rate and image quality in the encoded video sequences.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[2] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 614–619, 2003.

[3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[4] G. Khurana, A. A. Kassim, T. P. Chua, and M. B. Mi, "A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 536–540, 2006.

[5] B. Sheng, W. Gao, and D. Wu, "An implemented architecture of deblocking filter for H.264/AVC," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '04)*, pp. 665–668, October 2004.

[6] L. Li, S. Goto, and T. Ikenaga, "A highly parallel architecture for deblocking filter in H.264/AVC," *IEICE Transactions on Information and Systems*, vol. E88-D, no. 7, pp. 1623–1628, 2005.

[7] H. Jing, H. Yan, and X. Xinyu, "An efficient architecture for deblocking filter in H.264/AVC," in *Proceedings of the IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 848–851, September 2009.

[8] J. Zhou, D. Zhou, H. Zhang, Y. Hong, P. Liu, and S. Goto, "A 136 cycles/MB, luma-chroma parallelized H.264/AVC deblocking filter for QFHD applications," in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME '09)*, pp. 1134–1137, July 2009.

[9] Y. C. Lin and Y. L. Lin, "A two-result-per-cycle deblocking filter architecture for QFHD H.264/AVC decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 6, pp. 838–843, 2009.

[10] E. Ernst, *Architecture design of a scalable adaptive deblocking filter for H.264/AVC*, MSc dissertation, Rochester Institute of Technology, New York, NY, USA, 2007.