PAPER Special Section on Solid-State Circuit Design—Architecture, Circuit, Device and Design Methodology

A 64 Cycles/MB, Luma-Chroma Parallelized H.264/AVC Deblocking Filter for 4 K × 2 K Applications

Weiwei SHEN†, *Nonmember*, Yibo FAN†a), *Member, and* Xiaoyang ZENG†, *Nonmember*

SUMMARY In this paper, a high-throughput debloking filter is presented for H.264/AVC standard, catering video applications with 4 K × 2 K (4096 × 2304) ultra-definition resolution. In order to strengthen the parallelism without simply increasing the area, we propose a luma-chroma parallel method. Meanwhile, this work reduces the number of processing cycles, the amount of external memory traffic and the working frequency, by using triple four-stage pipeline filters and a luma-chroma interlaced sequence. Furthermore, it eliminates most unnecessary off-chip memory bandwidth with a highly reusable memory scheme, and adopts a "slide window" buffer scheme. As a result, our design can support 4 K × 2 K at 30 fps applications at the working frequency of only 70.8 MHz.

key words: *H.264/AVC, deblocking filter, pipeline, parallelism, 4 K × 2 K*

1. Introduction

The deblocking filter (DF) [1] is an important feature in H.264/AVC, the most recent video coding standard [2]. It not only reduces blocking distortion and artifacts of each 16 × 16 macroblock (MB) after reconstruction, but also offers up to 9% bit-rate-saving [3]. Compared with filters in previous video coding standard like H.263 or MPEG-2, the DF of H.264/AVC is much more complex because of its high adaptability of filter processing. High-throughput VLSI implementation of the DF is certainly essential for high-resolution video applications.

Nowadays, high-definition TV (HDTV) is gaining prevalence in TV broadcasting and video entertainment. Even higher video definition, such as QFHD (3840 × 2160), has gradually emerged. With a soaring demand on resolution, the design of video codec is becoming more challenging.

In H.264/AVC, the DF is one of the most data and computation intensive modules. In total, it may account for the one-third of the computational requirements of a H.264/AVC decoder [1], and cause the bottleneck in the processing speed of the decoder. Therefore, our work is motivated to propose a high-throughput deblocking filter, which can process one MB in 64 cycles for H.264/AVC video codec.

A straight method to enhance the throughput of DF is to raise the clock frequency. However, it is difficult and inefficient to raise the clock frequency, especially for power and thermal issues. Most efforts have been focused on reducing the clock cycles to enhance the performance. Pipelining and parallelism are well-known techniques to increase the system performance. Xu's [4] adopts 5-stage pipeline. However, it causes the data hazard due to the dependency between neighboring filter operations, and apply a complicated forwarding-circuit to solve it. So we adopt 4-stage pipeline to avoid the data hazard with a new structure.

Lin's [3], Chen's [5] and Tobajas's [6] employ the parallelism by using double-filter structure; nevertheless, the area of this architecture is also doubled as throughput increases. And zhou's [7] proposes a four edge filters organized in two groups for simultaneously processing vertical and horizontal edges to enhance its throughput. As another alternative of double-filter structure, Zhou's [8] processes the luma and chroma edges simultaneously, so as to enhance the parallelism of the filter while not notably increasing the area. However, there exists a distinct difference between the total amount of luma and chroma edges calculation, rendering to Zhou's [8] impossible to accomplish them at the same time. One of the filters for processing chroma edges would be idled, while its counterpart for luma is still at work.

Therefore, a triple-filter structure with a novel filter order is presented. Two filters are exclusive for processing luma edges, while one filter is exclusive for processing chroma edges. Additionally, this work applies on-chip SRAM, "slide window" buffer scheme and "luma-chroma interlaced sequence" to eliminate off-chip memory bandwidth. The proposed design meets the demand of high performance DF for H.264/AVC standard in 4 K × 2 K (4096 × 2304) ultra-definition video applications, while making a good tradeoff between off-chip memory bandwidth and on-chip memory usage.

The rest of this paper is organized as follows. In Sect. 2, the deblocking filter algorithm of H.264/AVC is briefly introduced. In Sect. 3, we describe our proposed architecture. In Sect. 4, we present our implementation results. Finally, in Sect. 5, we draw some conclusions.

2. Deblocking Filter Algorithm

In H.264/AVC, the DF takes place on the basis of MB as shown in Fig. 1, and the MB in a frame is filtered in raster scan order. Both 16 × 16 luma block and 8 × 8 chroma block can be divided into 4 × 4 blocks. For each 4 × 4 blocks, vertical edges should be processed from left to right at first, and then horizontal edges from top to bottom.

In order to filter the vertical and horizontal edges, the

Manuscript received August 10, 2011.
Manuscript revised November 4, 2011.

†The authors are with State Key Lab of ASIC and System, Fudan University, Shanghai, 200240, China.
a) E-mail: fanyibo@fudan.edu.cn

DOI: 10.1587/transele.E95.C.441

DF processes the data in the form of vertical and horizontal lines of pixels as shown in Fig. 2. Each line consists of eight pixels between every 4×4 block, denoted by $p3$, $p2$, $p1$, $p0$, $q0$, $q1$, $q2$, $q3$.

In H.264/AVC, the DF is used to update the pixels of every vertical and horizontal line by using several low-pass filters. For each edge, the selection of the appropriate low-pass filter is determined by the boundary strength (BS), threshold α, β, and the content of line of pixels.

The BS (the value ranges from 0 to 4) is determined by some prediction coding information, described in H.264/AVC standard [2]. Threshold α, β is decided by the quantization parameter and some syntax elements. Based on these conditions, each line of pixels can be decided whether it required to be filtered. And only if the four conditions of Eq. (1) are true, the low-pass filter would be applied to the line of pixels which meets the demand.

$$BS \neq 0, |p0 - q0| < \alpha, |p1 - p0| < \beta, |q1 - q0| < \beta \quad (1)$$

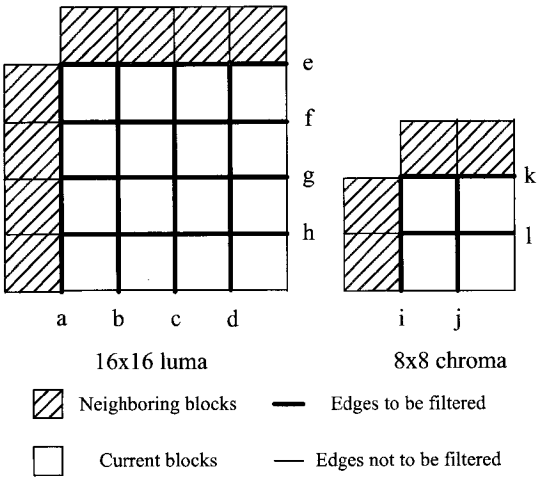


Fig. 1 Luma and chroma edges to be filtered.

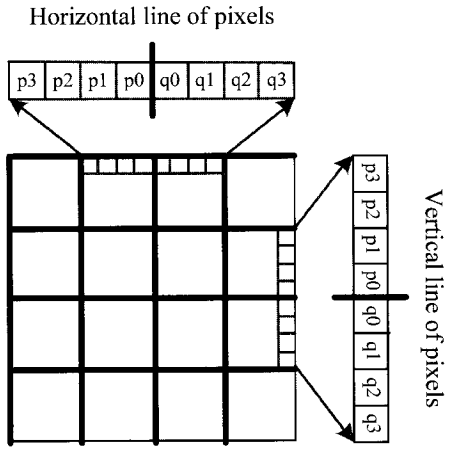


Fig. 2 Line of pixel samples.

3. Proposed Hardware Architecture Of Deblocking Filter

3.1 Top Architecture

The top architecture of deblocking filter is shown in Fig. 3.

The DF is mainly composed of five components: Control Logic Unit, SRAM & Register Unit, Generate BS & Threshold Unit, Triple-Filter Core and Write Back Unit. The inputs of the DF consist of some control signal such as clk (the system clock), rst (reset signal), the reconstruction MB and MB information (the reference information to produce BS value). The detailed description is as follows.

The design of luma-chroma parallelism is well discussed in the introduction. In order to accomplish the task of filtering luma and chroma components at the same time and avoid the idle time between two MBs, we propose the triple-filter architecture to increase the throughput of the DF drastically. And two filters are exclusive for processing luma edges, while the other one is exclusive for processing chroma edges.

3.2 Proposed 4-Stage Pipeline Architecture

In above discussion, we proposed the four-stage pipeline architecture instead of the five-stage pipeline architecture to avoid data hazard. The five-stage pipeline architecture is traditionally used in [4]. For example, Fig. 4 shows three adjacent 4 × 4 blocks with a filter order from step0 to step7. Each step takes the four aligned pixels locating on the left and right side simultaneously. For instance, step 0 takes

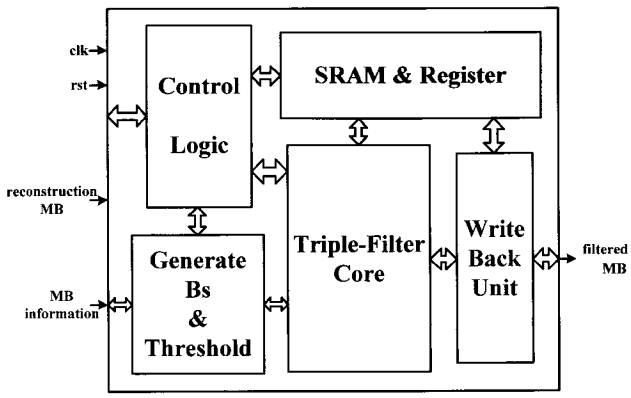


Fig. 3 Top Architecture of deblocking filter.

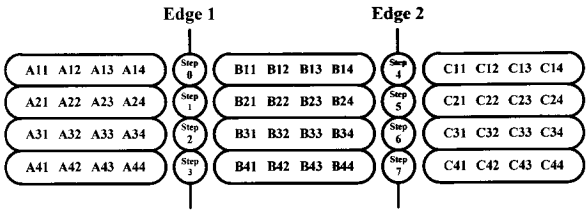


Fig. 4 Three adjacent 4 × 4 blocks and the filter order.

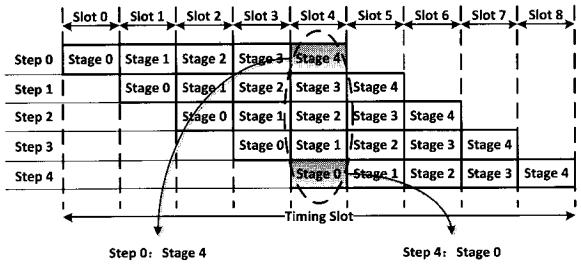


Fig. 5 The data hazard in five-stage pipeline architecture.

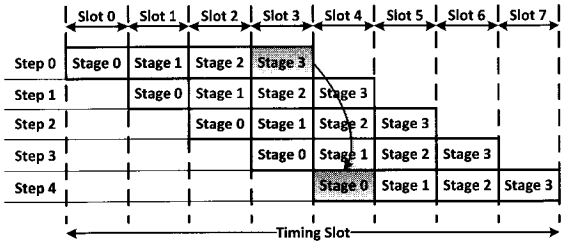


Fig. 6 Four-stage pipeline architecture without data hazard.

A11, A12, A13, A14 (left four pixels) on the left as well as B11, B12, B13, B14 (right four pixels) on the right. Noting that, B11, B12, B13, B14 are also used in step 4.

Data hazard is hard to eliminate on adoption of the five-stage pipeline architecture as depicted in Fig. 5. B11, B12, B13, B14 should be ready for access during step 4 at slot 4 and step 0 at slot 4 at the same time; under this scenario, a data collision occurs, and the four pixels would not be available until slot 5.

To solve this issue, a four-stage pipeline architecture is proposed, as shown in Fig. 6. B11, B12, B13 and B14 finish filtering when they are required for step 4 at slot 4. Thus, there exists no data collision any more.

In order to avoid data hazard, we adopt four-stage pipeline architecture to increase the throughput. The average time required for one filtering operation is reduced significantly. And the filtering operations are divided into four balanced stages. The filter for luma edges is depicted by Fig. 7, and the filter for chroma edges is similar to it.

The function of each stage is illustrated as follows:

Stage 0: read pixels to be filtered from on-chip SRAM or register array.

Stage 1: calculate the threshold α, β, the BS value according to the MB coding information, and evaluate Eq. (1) for the following filtering operation.

Stage 2: according to the BS and other threshold obtained in stage 1, different taps of filters are applied to filter the pixels across the current edge. The strong filter would be applied to the edges to be filtered if the BS equals 4, while the weak filter would be applied to it if the BS equals 1, 2 or 3. And the detailed filtering computation can be found in H.264/AVC standard [2].

Stage 3: filtered data would be wrote back to on-chip SRAM or register arrays (to be further filtered), or to the external memory (to be referenced for the other modules of the video codec or to be displayed).

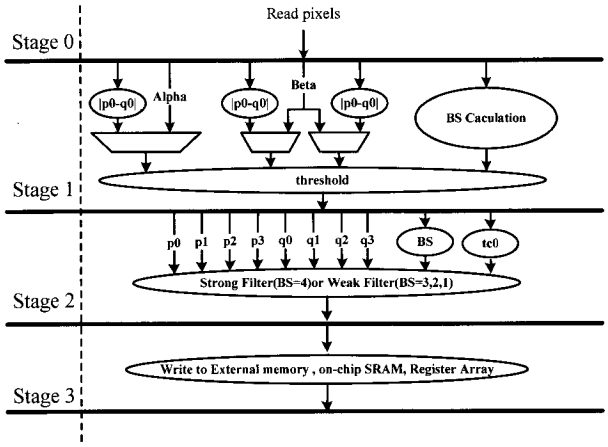


Fig. 7 Four-stage pipeline for luma edges.

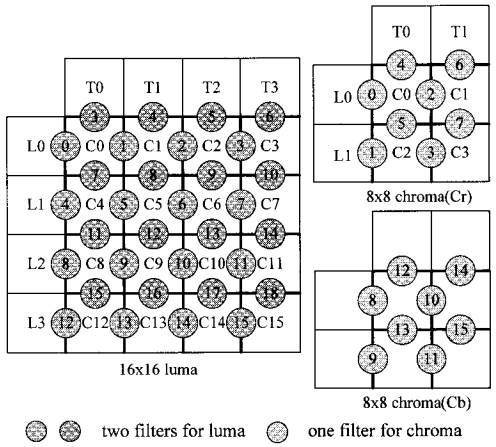


Fig. 8 Proposed filter order.

3.3 Proposed Filter Order

The proposed filter order is showed in Fig. 8. In each 4×4 block edge, the vertical edge is filtered from left to right (if any), and the horizontal edge is filtered from top to bottom (if any). Although proposed filtered order is not totally the same as to those specified in H.264/AVC standard, the final filtered data are the same. The filter order, together with the proposed memory organization and memory update scheme, raises the four stage pipeline throughput. Each circle stands for 4 clock cycles, and the whole time needed for filtering one MB is 64 clock cycles owing to pipeline architecture.

3.4 Proposed Memory Organization

The proposed filtered order alone does not have advantage over other designs. It should be combined with an efficient and suitable memory organization to eliminate data hazard and enhance data reuse, which achieves high performance of the DF.

Based on the previous analysis, the memory blocks are

Table 1 Memory structure.

Memory structure	Size (bit)			Memory type
	Luma	Chroma	Total	
Left neighboring Memory	512	256	768	Register array
Upper neighboring Memory	512	256	768	Dual-port SRAM
Current MB Memory	2048	512	2560	Dual-port SRAM
Intermidate data Buffer (D0-D6)	512	256	768	Register array

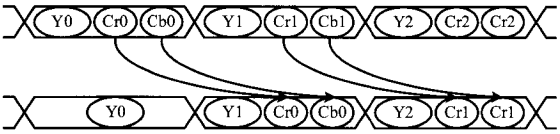


Fig. 9 Luma-chroma interlaced sequence.

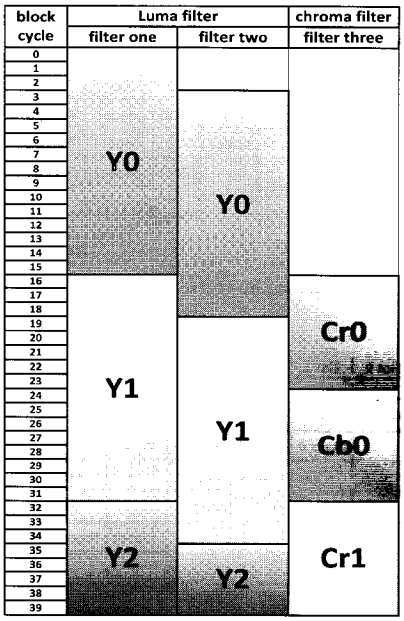


Fig. 10 Filtering schedule of the proposed work.

required to provide the original pixels and store the intermediate pixels (to be further filtered). The left neighboring memory stores the 16 × 4 pixels (four 4 × 4 blocks) of luma, 16 × 2 pixels (two 4 × 4 blocks) of chroma from the left neighboring MB, which will be filtered at the edge a, edge i in Fig. 1. The upper neighboring memory stores the 16 × 4 pixels (four 4 × 4 blocks) of luma, 16 × 2 pixels (two 4 × 4 blocks) of chroma from the upper neighboring MB, which will be filtered at the edge e, k. The current MB memory stores the raw pixels, and 16 × 16 pixels (sixteen 4×4 blocks) of luma, while 16×4 pixels (four 4×4 blocks) of chroma. Besides these three memory modules, we also employ six 4 × 4 pixels register arrays. These six register arrays keep the temporary data between adjacent edges, and they are also employed as transposition buffer. The specific implementation of the each memory is showed in Table 1.

Figure 9 illustrates a luma-chroma interlaced sequence method. The calculation of the BS value is a complicated process according to H.264/AVC standard. Owing to luma and chroma edges share the same BS, and each chroma (Cr&Cb) edges' BS can be found from luma (Y) edges. In other designs, the results of Y and Cr&Cb will be outputted at the same round. In our design, inside one MB, the Y component is filtered at first, meanwhile the BS which will be reused in the processing of Cr&Cb are stored. At the next round, the Y component of next MB and the Cr&CB of current MB will be processed. So the Y and Cr&CB's output of the current MB will be interlaced. The proposed architecture does not affect the pipeline's efficiency.

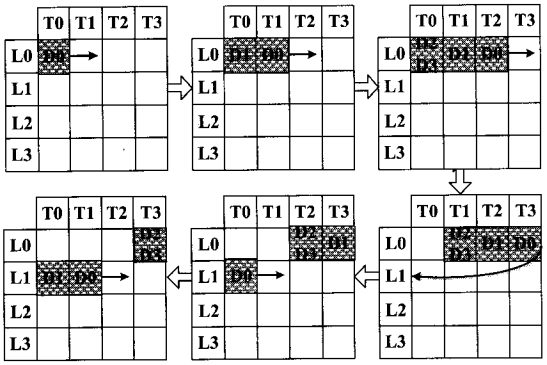
Figure 10 demonstrates the filtering schedule of the proposed work. Each block cycle stands for 4 clock cycles. Two filters are used to process luminance component (Y component), while one filter is used to process chrominance component (Cr&Cb component). And the whole time needed for filtering one MB is 16 block cycles (64 clock cycles) owing to pipeline architecture.

D0-D3 register arrays: here, we propose a novel slide-window buffer scheme for luma component. It achieves a tradeoff between on-chip memory's access and register arrays' size. Each circle in Fig. 8 stands for 4 clock cycles. In step 0, the raw pixels are from left neighboring register arrays L0 and current MB's on-chip SRAM C0 (shown in Fig. 8). After filtering step 0, L0 are written back to external memory (finish filtering), and C0 are stored into D0 (to be further filtered). In step 1, the raw pixels are from C1 (on-chip SRAM) and D0, and the filtered pixels are stored into D0 and D1 in Fig. 11(a). In step 2, the raw pixels are from C2 and D0, and the filtered pixels are still stored into D0 and D1. Meanwhile, the whole data of D1 are coped to D2 for horizontal line filter, because at the filtering process of step 1, the total results of 4 × 4 blocks are stored into D1 which needs four cycles according to the pipeline architecture. In step 3, the two luma filters start to work simultaneously. And the register arrays of D0-D3 start to slide as showed in Fig. 11(a). At the end of the current MB, when at step 15, one filter for vertical filter are finished its work of current MB, and the other filter for horizontal filter are still working in current MB. Meanwhile the filter for vertical filter will start to process the following MB. Thus, there would be no time wasted.

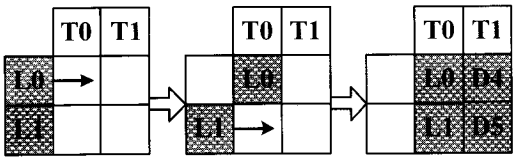
D3 register arrays: owing to the discontinuity between two rows of luma filter for vertical edges. D3 is used to store the intermediate data for further horizontal filter.

Above all, the C0-C15 for first filter is from on-chip SRAM. After first filter, they are stored in D0-D3 for further filter. The detailed register array schedule for luma is showed in Fig. 12. The several blank ones on both the top and bottom of the table represent the data which are belongs to the last and next MB respectively.

D4-D5 register arrays: as shown in Fig. 11(b), the memory organization of D4-D5 of chroma's is simpler than



a. Slide-window buffer scheme for luam



b. Buffer scheme for chroma

Fig. 11 Proposed buffer scheme.

	D0	D1	D2	D3
Step 0				
Step 1	C0			
Step 2	C1	C0	C0	
Step 3	C2	C1	C1	
Step 4	C3	C2	C2	
Step 5	C4	C3	C3	
Step 6	C5	C4	C4	C3
Step 7	C6	C5	C5	C3
Step 8	C7	C6	C6	C3
Step 9	C8	C7	C7	C3
Step 10	C9	C8	C8	C7
Step 11	C10	C9	C9	C7
Step12	C11	C10	C10	C7
Step 13	C12	C11	C11	C7
Step 14	C13	C12	C12	C11
Step 15	C14	C13	C13	C11
Step 16	C15	C14	C14	C11
Step 17		C15	C15	C11
Step 18				C15

Fig. 12 Register array schedule for luma.

luma's. At step 0, the raw pixels are from left neighboring register arrays L0 and current MB's on-chip SRAM C0 (shown in Fig. 8). After filtering the step 0, the filtered pixels from L0 are written to the external memory, and the filtered pixels from C0 are written L0. The process of step 1 is similar to step 0. Thus, the following temporary data are all stored in L0, L1, D4, and D5.

4. Implementation Results

The proposed architecture has been implemented in Verilog-HDL and synthesized under a timing constraint of 200 MHz with SMIC 0.13-μm library. Gate count of every module is summarized in Table 2.

According to Table 2, the cost of the computational circuits and that of the storage part (excluding the local SRAM) are found to be equivalent as 13 K and 13.1 K, respectively. It also demonstrates that the proposed architecture achieves a good trade-off between performance enhancement and memory size while some previous designs adopt internal buffers of heavy cost. In addition, the control logic occupies 4.5 K.

Table 2 Gate count summary.

Purpose	Gate count(NAND2)
Control logic	4.5K
Edge filter	13.1K
Write back (including D0-D6)	13K
Total	30.6K

Table 3 Details of proposed design.

technolgy	0.13-um
target real-time performance	4Kx2K(4096x2304) at30fps
operating frenquency	70.8Mhz
gate count (NAND2)	30.6K
edge filter number	3
dual-port SRAM	416bytes

Table 3 summarizes the details of the implementations of the proposed design. The proposed work can process one MB in 64 cycles, and it can achieve the real-time performance requirement of 4 K × 2 K at 30 fps with working frequency at 70.8 MHz. The area cost is 30.6 K gates (excluding the local SRAM).

Table 4 shows the comparison between this work and the state-of-art architectures. Compared with the existing approaches [3], [4], [6], [8], [9], our architecture requires the smallest processing cycles per MB. The proposed design can achieve the data throughout requirement of the real-time 4 K × 2 K at 30 fps resolution with only 70.8 MHz clock frequency. The low operating frequency also contributes to the decrease of power consumption and is able to support much higher data throughout than other architectures in Table 4. The size of gate count and local SRAM is moderate compared with other designs.

5. Conclusion

This paper proposes a high-throughput architecture to accelerate the speed of filtering operation for the H.264 /AVC deblocking filter. With the featured four-stage balanced pipeline, the maximum operating speed can reach 200 MHz. Meanwhile, the luma-chroma parallel architecture is proposed to reduce the filtering operation to merely 64 cycles for a MB. Six register arrays are employed to store intermediate filtered data. As a result, our design can support 4 K × 2 K at 30 fps applications at the frequency of only 70.8 MHz, and it is capable for even higher resolution video

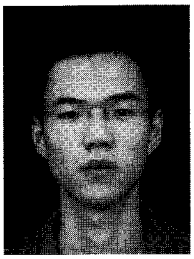
Table 4 Implementation comparsion.

Reference	[3]	[4]	[6]	[8]	[9]	Proposed
Year	2009	2008	2008	2009	2009	2011
Cycle/MB	100	204	110	136	192	64
SRAM size (byte)	416	768+2Nx4	192	1056	512+2Nx4	416
Gate count (K)	22.9	21.5	12.6	17.9	26.01	30.6
Technology (um)	0.13	0.18	0.18	0.09	0.18	0.13
Number of edge filter	2	1	2	2	1	3
Application Target	QFHD@30fps (3840x2160)	QFHD@30fps (3840x2160)	1080p@30fps (1920x1088)	QFHD@30fps (3840x2160)	QFHD@30fps (3840x2160)	4Kx2K@30fps (4096x2304)
Frequency (Mhz)	98	198	100	133	187	70.8

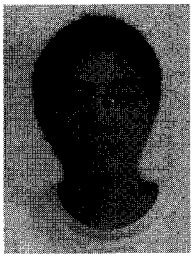
applications.

References

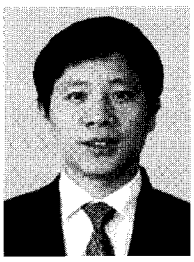
[1] P. List, A. Joch, J. Lainema, G. Bjntegarrd, and M. Karczewicz, "Adaptive deblocking filter," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.614–619, July 2003.
[2] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264\ISO/IEC 14496-10 AVC), JVT-G050.
[3] Y.-C. Lin and Y.-L. Lin, "A two-result-per-cycle deblocking filter architecture for QFHD H.264/AVC decoder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.17, no.6, pp.838–843, 2009.
[4] K. Xu and C.-S. Choy, "A five-stage pipeline, 204 cycles/MB, single-port SRAM-based deblocking filter for H.264/AVC," IEEE Trans. Circuits Syst. Video Technol., vol.18, no.3, pp.363, 2008.
[5] C.-M. Chen and C.-H. Chen, "Configurable VLSI architecture for deblocking filter in H.264/AVC," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.16, no.8, pp.1072–1082, 2008.
[6] F. Tobajas, G.M. Callico, P.A. Perez, V. de Armas, and R. Sarmiento, "An efficient double-filter hardware architecture for H.264/AVC deblocking filtering," IEEE Trans. Consum. Electron., vol.54, no.1, pp.131–139, 2008.
[7] D. Zhou, J. Zhou, J. Zhu, and S. Goto, "48 cycles/MB H.264/AVC deblocking filter architecture for ultra high definition applications," IEICE Trans. Fundamentals, vol.E92-A, no.12, pp.3203–3210, Dec. 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," IEEE International Conference on Multimedia and Expo, 2009. ICME 2009, p.1134, 2009.
[9] N.T. Ta, J.S. Youn, H.G. Kim, J.R. Choi, and S.-S. Han, "Low-power high-throughput deblocking filter architecture for H.264/AVC," 2009 International Conference on Electronic Computer Technology, p.627, 2009.



Weiwei Shen received the B.S. degrees in Microelectronics Department from Fudan University in 2010. He now is working at the State Key Lab. of ASIC and System in Fudan University, Shanghai, China. His research interests include video and image processing, digital signal processing, and VLSI design.



Yibo Fan received the B.E. degree in electronics and engineering from Zhejiang University, China in 2003. M.S. degree in Micro electronics from Fudan University, China in 2006, and Ph.D. degree in engineering from Waseda University, Japan in 2009. From 2009 to 2010, he worked as an Assistant Professor in Shanghai Jiaotong University, And currently, he is the Assistant Professor in Department of Microelectronics of Fudan University. His research interesting includes information security, video coding and associated VLSI architecture.



Xiaoyang Zeng received the B.S. degree from Xiangtan University, China in 1992, and the Ph.D. degree from Changchun Institute of Optics and Fine Mechanics, Chinese Academy of Sciences in 2001. From 2001 to 2003, he worked as a post-doctor researcher at the State Key Lab of ASIC & System, Fudan University, P.R. China. Then he joined the faculty of Department of Micro-electronics at Fudan University as an professor. His research interests in signal processing, and communication systems. Prof. Zeng is the Chair of Design-Contest of ASP-DAC 2004 and 2005, also the TPC member of several international conferences such as ASCON 2005 and A-SSCC 2006, etc.