ANALYSIS AND INTEGRATED ARCHITECTURE DESIGN FOR OVERLAP SMOOTH AND IN-LOOP DEBLOCKING FILTER IN VC-1

Yen-Lin Lee and Truong Nguyen

ECE Dept., UCSD, La Jolla, CA 92093-0407
Email: yel004@ucsd.edu, nguyent@ece.ucsd.edu
http://videoprocessing.ucsd.edu

ABSTRACT

Unlike familiar macroblock-based in-loop deblocking filter in H.264, the filters of VC-1 perform all horizontal edges (for in-loop deblocking filtering) or vertical edges (for overlap smoothing) first and then the other directional filtering edges. The entire procedure is very time-consuming and with high memory access loading for the whole system. This paper presents a novel method and the efficient integrated architecture design, which involves an 12×12 overlapped block that combines overlap smoothing with loop filtering for performance and cost by sharing circuits and resources. This architecture has capability to process HDTV1080p 30fps video and HDTV 2048×1536 24fps video at 180MHz. The same concept is applicable to other video processing algorithms, especially in deblocking filter for video post-processing in a frame-based order.

Index Terms— Overlap smoothing, in-loop deblocking filter, VC-1, SMPTE-421M, VLSI architecture

1. INTRODUCTION

VC-1 [1] is a video codec specification that has been standardized by the Society of Motion Picture and Television Engineers (SMPTE) and adopted in next-generation optical media formats like HD-DVD and Blu-ray. Comparing to H.264/AVC [2], the design approach of VC-1 lowers computational complexity without significant performance loss since H.264/AVC uses many complex techniques for better visual quality. Lower complexity leads to practical architecture, lower cost, and smaller power consumption and heat dissipation. Although VC-1 operates at lower complexity, it still approaches a compression ratio similar to H.264/AVC [3]. Fig. 1 shows the encoding loop of a VC-1 codec.

Deblocking filter is an important part of the advanced codec because it improves visual quality of decoded frames. Hence, VC-1 operates an in-loop filtering prior to a current decoded frame as the future reference picture. Besides this loop filter, VC-1 also uses another filter, overlap smooth, for smoothing real blocking artifacts [4]. However, the procedures of these two filters are both arranged by frame-based orders, which means that all horizontal edges (for loop filtering) or vertical edges (for overlap smoothing) should be performed first and followed by the other directional filtering edges. Although there are many well-designed methods and architectures that specifically deal with loop filtering in a macroblock-based order [5][6], none of them is appropriate for VC-1 frame-based filtering. The reason is that these methods fail to filter all edges within the current macroblock due to the problem of data dependency. In other words, some filtering edges should not be performed prior to particular edges on boundary between the current macroblock and the neighboring unreconstructed macroblocks. If these previous methods directly apply to VC-1 deblocking filter, they could cause many unnecessary processing cycles and inefficient memory access.

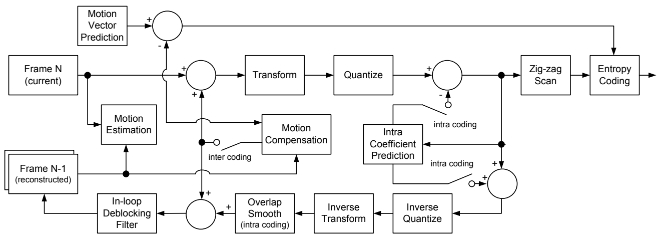


Fig. 1. Encoding loop of VC-1 codec.

In this paper, we propose a method to solve the data dependency problem and accelerate the processing time. Our method defines a 12×12 overlapped block and processes both overlap smoothing and loop filtering within it. This method is a key technique that helps us perform VC-1 deblocking filter in a block-based order but not a frame-based order so as to pipeline with block reconstructing. Besides, it also combines two VC-1 filters in order to improve the performance and reduce the cost by sharing circuits and resources.

The rest of this paper is organized as follows. In Section 2, two VC-1 filters are described. In Section 3, we proposed processing techniques and integrated architecture. Implementation results and performance are in Section 4. Finally, conclusions will be given in Section 5.

2. DEBLOCKING FILTERS IN VC-1

2.1. Overlap Smooth

An overlap smoothing operation should conditionally be performed across the edge of two neighboring intra blocks, for both the luma and chroma data. This lapped transform performs a pre-processing step in spatial domain during the encoding procedure and a post-processing step following inverse transform during the decoding procedure [4]. With a well designed overlapped transform, blocking artifacts can be minimized, and decoded frames can retain most original information. Moreover, computational complexity of overlap smoothing is lower than normal loop filters and could be applied to simple applications without loop filter.

A portion of a P frame including three 8×8 intra blocks and one 8×8 inter block is shown in Fig. 2. An overlap smoothing operation is applied on four pixels, i.e., two pixels on each side of the block boundary. It performs on vertical edges first and then hori-

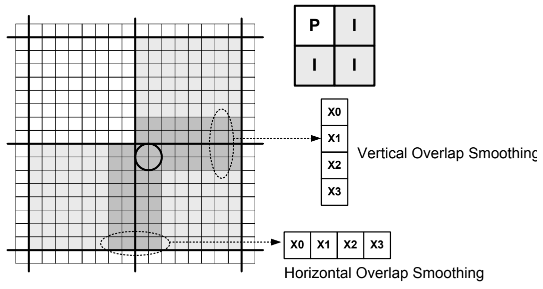


Fig. 2. Example showing overlap smoothing

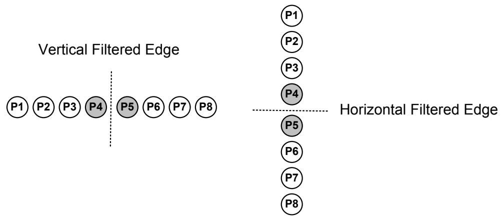


Fig. 3. 8 Pixels involving in loop filtering operation

zontal edges in a specific order within a frame or slice. If the pixels are filtered for both vertical and horizontal edges, such as the pixels in the 2×2 corner marked with the dark circle, vertical filtering operation must be performed first and followed by horizontal filtering operation. Overlap smoothing will carried out on unclamped 10 bit reconstructed pixels. Subsequent to filtering, the constant value of 128 will be added to each pixel of the block, which will then be clamped to the 8-bit range to produce the reconstructed output.

2.2. In-loop Deblock Filter

An in-loop deblock filtering operation is done prior to using the reconstructed frame as a reference for motion predictive coding. In order to lower computational complexity, the filtered edges in VC-1 standard are divided into 4-pixel segments. In each 4-pixel segment, the third pair will be filtered first, and the result of this filtering operation determines whether the other three pixel pairs in the segment are also filtered. As Fig. 3 shows, the filtering process takes a discontinuity measurement involving pixels P1 through P8 that seeks to detect whether the discontinuity is above a certain threshold. If a discontinuity is found, then a loop filtering operation is done on pixels P4 and P5 for smoothing the discontinuity.

The procedure of loop filtering, which performs horizontal edges prior to vertical edges, is also different from overlap smoothing, which performs vertical edges prior to horizontal edges. When there are more than one slice in a picture, the filtering for each slice should be performed independently. For I and B pictures, loop filter has to detect and perform on all 8×8 block boundaries. First, all blocks or subblocks that have a horizontal boundary along the 8^{th}, 16^{th}, 24^{th}, etc horizontal lines would be filtered. Next, all blocks or subblocks that have a vertical boundary along the 8^{th}, 16^{th}, 24^{th}, etc columns would be filtered. For P pictures, blocks could be intra or inter-coded. Inter-coded blocks adopt 8×8, 8×4, 4×8, or 4×4 size for inverse transform, and the boundary between these transform blocks or subblocks should be filtered. First, all blocks or subblocks that have a horizontal boundary along the 8^{th}, 16^{th}, 24^{th}, etc horizontal lines should be filtered. Next, all subblocks that have a horizontal boundary along the 4^{th}, 12^{th}, 20^{th}, etc horizontal lines should be filtered. Next, all blocks or subblocks that have a vertical boundary along the 8^{th}, 16^{th}, 24^{th}, etc columns should be filtered. Lastly, all blocks or subblocks that have a vertical boundary along the 4^{th}, 12^{th}, 20^{th}, etc columns should be filtered. Fig. 4 shows an example for all filtered edges relative to the luma data of a macroblock. For I or B-frame, the loop filtering procedure should be $H0$, $H1$, $H2$, $V0$, $V1$, and $V2$. For P-frame, the loop filtering procedure should be $H0$, $H1$, $H2$, $h0$, $h1$, $V0$, $V1$, $V2$, $v0$, and $v1$. Comparatively, the overlap smoothing procedure should be $V0$, $V1$, $V2$, $H0$, $H1$, and $H2$. These two filters have different processing order.

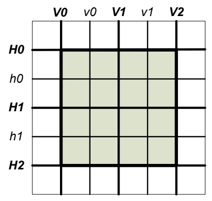


Fig. 4. All filtered edges relative to the luma data of a macrolbock.

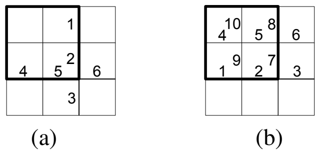


Fig. 5. Modified processing order. (a) An overlap smooth processing order. (b) A loop filter processing order.

3. PROPOSED METHODS AND ARCHITECTURE

3.1. Modified Processing Order

In order to shorten processing time and reduce unnecessary memory access cycles, an architecture that cooperates with block reconstruction is proposed. In our method, this block-based procedure adopts a 12×12 overlapped block as our basic filtered block size including a filtered area and an overlapped area. The pixels in the filtered area will fully be filtered and will be written back to external buffer for displaying or post-processing. The pixels in the overlapped area would be partially filtered in the current overlapped block and will be filtered by other future overlapped blocks again. This proposed overlapped block is useful and employed as a common processing field for both overlap smoothing and loop filtering.

Two modified processing orders for overlap smoothing and loop filtering are shown in Fig. 5. We design these orders in a block size for processing in block-based order but not original frame-based order. Fig. 5(a) shows an overlap smooth processing order applied to an overlapped block. Any edge with a predefined index in this figure could be a filtered edge during processing, and the filtered edge with a smaller index should be processed prior to that with a larger index. However, not every predefined edge should be filtered during the procedure of most overlapped blocks. When these edges are filtered by a previous overlapped block, they cannot be filtered again. When pixels of the block complete all filtering operations and will not be used for succeeding overlap smoothing, the constant value of 128 should be added to each pixel and then be clamped to the 8-bit range for producing the reconstructed output before loop filtering. Fig. 5(b) shows a loop filter processing order applied to an over-

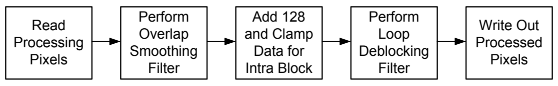

Read Processing Pixels
Perform Overlap Smoothing Filter
Add 128 and Clamp Data for Intra Block
Perform Loop Deblocking Filter
Write Out Processed Pixels

Fig. 6. Processing flow of an overlapped block.

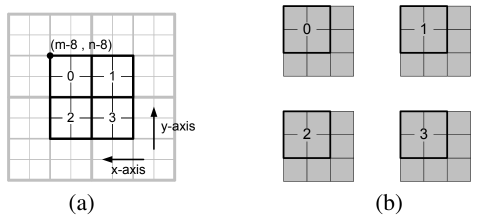


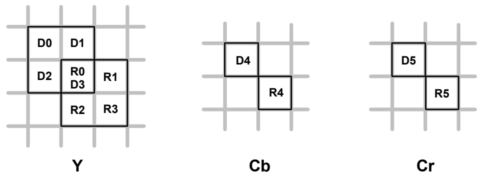
Fig. 7. (a) A moving filtered macroblock. (b) Four separated overlapped blocks from the filtered macroblock.

lapped block. Similarly, a filtered edge with a smaller index should be performed earlier than that with a larger index, and the edge filtered by previous block should not be filtered again. Actually, only the first overlapped block of a frame or slice would be processed on all predefined overlap smoothing or loop filtering edges. By performing filtering operations in our modified processing order within an overlapped block, it can integrate these two VC-1 filters in hardware. The advantage is that the second modified processing order for loop filtering can follow the first processing order without any memory access. Fig. 6 shows a processing flow of an overlapped block.

3.2. Pipeline Processing by Moving Macroblock Position

One important reason that the proposed method attempts to modify original processing order to a block-based procedure is to pipeline the filtering step with the reconstruction step. Nevertheless, it still fails to filter all edges within the current macroblock because some filtering edges should not be performed prior to particular edges on boundary between the current macroblock and neighboring unreconstructed macroblock. For example, vertical edges should be filtered prior to horizontal edges when the related pixels belong to both directional overlap smoothing edges. A method that shifts the filtered macroblock position is proposed here. In particular, the position of a filtered macroblock is shifted left 8 pixels in x-axis and up 8 pixels in y-axis on a frame or slice from the position of a current reconstructed macroblock. When the position of a current macroblock is located on (m , n), the position of this filtered macroblock will be located on (m-8 , n-8) shown in Fig. 7(a). If there are pixels of the filtered macroblock that protrude the border of the current frame or slice, these pixels of the filtered macroblock do not exist and no operation will be performed on them. Based on this shifting filtered macroblock, we separate a filtered macroblock into four basic overlapped blocks shown in Fig. 7(b), which could be performed by our proposed modified processing orders for both overlap smoothing and loop filtering. Fig. 8 shows the pipeline schedule during the period of performing a reconstructed macroblock and a filtering macroblock.

In the proposed method, we need the temporal data buffer to store partially filtered or unfiltered pixels. There are two kinds of buffer: the temporal data buffer implemented by on-chip SRAM that stores short-time pixels for the current macroblock and the next macroblock; the temporal data buffer allocated in the external memory that stores long-time pixels for the next macroblock row. This external memory buffer size would be the width of the picture multiplied by the data size of 8 pixels and then multiplied by 2 (for both luma and chroma data). Data processing of performing a reconstructed macroblock and a filtering macroblock for luma data is shown in Fig. 9. The data of top three 8×8 blocks in Fig. 9 are temporal neighboring read data, which are partially filtered by previous filtered macroblocks. The data of bottom three 8×8 blocks in Fig. 9 would be temporal neighboring write data, which will partially be filtered by current macroblock and written back into the temporal data buffers. The data of top-left four 8×8 blocks in Fig. 9 will be filtered pixels and will be written into a decoded picture buffer of the external memory as the future reference data or for displaying. The data of bottom-right four 8×8 blocks in Fig. 9 are the reconstructed pixels. The procedure for chroma data is similar to that for luma data.



Time	T0	T1	T2	T3	T4	T5	T6
Reconstruction	R0	R1	R2	R3	R4	R5	-
Filtering	-	D0	D1	D2	D3	D4	D5

Fig. 8. Pipeline time schedule of a reconstructed macroblock and a filtering macroblock.

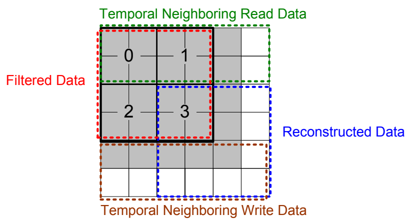

Temporal Neighboring Read Data
Filtered Data
0
1
2
3
Reconstructed Data
Temporal Neighboring Write Data

Fig. 9. Data processing for a reconstructed macroblock and a filtered macroblock.

3.3. Integrated Architecture Design

Block diagram of the proposed architecture is shown in Fig. 10. It performs both overlap smoothing and loop filtering based on an overlapped block mentioned above. Temporal Data Buffer (on-chip) stores unfiltered reconstructed data from Motion Compensation, unfiltered or partially filtered temporal neighboring data from the external memory or our proposed integrated filter. External Memory Controller manages the read/write cycles from/to the external memory. Both Motion Compensation and the proposed integrated architecture connect with System Controller that controls overall system and has a HW/SW system interface, such as Memory-mapped I/O (MMIO). Filter Control Unit is the central administration including two major finite state machines: one controls the flow among different overlapped blocks within a filtering macroblock; the other is in charge of the schedule among different filtering edges within an overlapped block. The parameters previously stored in the external memory should be read back into Filtering Parameter Registers

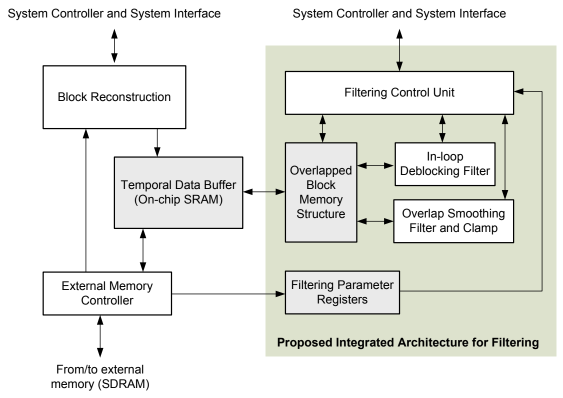

System Controller and System Interface
System Controller and System Interface
Block Reconstruction
Filtering Control Unit
Temporal Data Buffer (On-chip SRAM)
Overlapped Block Memory Structure
In-loop Deblocking Filter
Overlap Smoothing Filter and Clamp
External Memory Controller
Filtering Parameter Registers
Proposed Integrated Architecture for Filtering
From/to external memory (SDRAM)

Fig. 10. Block diagram of the proposed integrated architecture.

Table 1. Hardware cost of the proposed VC-1 filter architecture

Functional Block	Gate Counts
Overlap Smooth and Clamp	4802
In-loop Deblocking Filter	4076
Filtering Control Unit and Others	8346
Overlap Block Memory Structure	15333
Total	32557

before filtering operation. Overlapped Block Memory Structure is also a local buffer that stores all pixels within an overlapped block. Overlap Smoothing Filter and Clamp and In-loop Deblocking Filter access input data from Overlapped Block Memory Structure and perform the filtering operation respectively according to the proposed modified processing order. During the procedure, we pipeline all filtering operations so as to reduce the processing time. For a target to access pixels in a shortest time, we use an efficient structure that remaps and rotates the order of filtering pixels to implement Overlapped Block Memory Structure.

4. EXPERIMENTAL RESULT

Our integrated architecture is designed using VHDL and implemented by TSMC 0.18-μm technology on Artisan cell library. The implemented architecture operates with our VHDL- and C-model, and the result has been verified with the reference VC-1 decoder software. Table 1 shows logic gate count including several functional blocks and a synthesized memory buffer in the proposed VC-1 filter synthesized at 180MHz. For the sake of evaluating performance of the proposed architecture, we create several worst-case test patterns for I, B or P-frames in two highest resolutions of VC-1, 1080p and 2048$\times$1536. When performing these patterns, the filter will process on every boundary of all 8$\times$8 blocks or 4$\times$4 blocks for two kinds of filters. Table 2 demonstrates the processing cycles and processing time for different cases, and the result meets the requirement for HDTV1080p 30fps video and HDTV 2048$\times$1536 24fps video. Table 3 shows the analysis of external memory bandwidth and SRAM requirement for a worst-case 1080p in two different implemented architectures: the first one only implements VC-1 in-loop deblocking filter; the other implements an integrated architecture for both two filters. Compared with software, the former proposed architecture reduces 61.31% of external memory cycles, and the later reduces 68.3% of external memory cycles. Because the input to the overlap smoothing process should be the inverse transformed spatial block of pixels whose dynamic range is 10 bits, we use data width of 16-bits to store temporal data in the external memory and on-chip SRAM. That is why the proposed architecture with both filters requires more external memory cycles and larger SRAM size. In addition, the external memory bandwidth of software includes written reconstructed data.

Table 2. Processing time of the proposed VC-1 filter architecture

Frame Type (Worst Case)	Cycles	Time
1080p, 30fps I, B-frame	3829977	21.06 ms
1080p, 30fps P-frame	4771316	26.24 ms
2048$\times$1536, 24fps I, B-frame	5767373	31.72 ms
2048$\times$1536, 24fps P-frame	7184794	39.52 ms

Table 3. Memory Bandwidth and SRAM Requirement (1080p)

	Software	Proposed	SRAM Requirement
DB	18.74 MB/f	7.25 MB/f	1.088KB
OS+DB	35.87 MB/f	11.37 MB/f	2.176KB

5. CONCLUSION

This paper presents a novel processing method and an efficient integrated architecture for VC-1 filters based on an overlapped block. The proposed method uses overlapped block of size 12$\times$12 in this paper, but the same concept can be extended to other block sizes as well. For performance and cost, the proposed architecture integrates overlap smooth with loop filter and pipelines the procedure with block reconstructing even though the filtering procedure is frame-based described in VC-1 standard. The architecture is implemented in VHDL and synthesized by TSMC 0.18-μm technology on Artisan cell library. It has capability to process HDTV1080p 30fps video and HDTV 2048$\times$1536 24fps video at 180MHz. Moreover, the proposed method is also applicable to other video processing algorithms, especially in deblocking filter for video post-processing or other in-loop filters in a frame-based order.

6. REFERENCES

[1] *VC-1 Compressed Video Bitstream Format and Decoding Process (SMPTE 421M-2006)*, SMPTE Standard, 2006.

[2] *Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264/ISO/IEC 14496-10*, Mar. 2005.

[3] S. Srinivasan, S. L. Regunathan, "An overview of VC-1," Visual Communications and Image Processing, Proc. of SPIE, Vol. 5950, pp.720-728, 2005.

[4] T. D. Tran, J. Liang, and C. Tu, "Lapped transform via time-domain pre- and post-filtering", *IEEE Trans on Signal Processing*, vol.51, no.6, pp.1557-1571, June 2003.

[5] C. C. Cheng, T. S. Chang, and K. B. Lee, "An in-place architecture for the deblocking filter in H.264/AVC," *IEEE Trans. on Circuits and Systems - PartII: Express Briefs*, Vol.53, No.7, July 2006.

[6] T. C. Chen, S. Y. Chien, Y. W. Huang, C. H. Tsai, C. Y. Chen, and L. G. Chen, "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC Encoder," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.16, No.6, June 2006.