A PLATFORM-BASED ARCHITECTURE OF LOOP FILTER FOR AVS

Bin Sheng[1] *Wen Gao*[1, 2, 3] *Di Wu*[1]

[1](Department of Computer Science and Technology, Harbin Institute of Technology, China)
[2](Institute of Computing Technology, Chinese Academy of Sciences, China)
[3](Graduate School of Chinese Academy of Sciences, China)
E-mail: {bsheng, wgao, dwu}@jdl.ac.cn

ABSTRACT

AVS is Chinese new audio and video coding standard, in which a loop filter has been applied to remove blocking artifacts. A platform-based architecture for the loop filter of AVS standard is proposed in this paper. The advantages of column-separated SRAM organization and two configurable matrix transposers have been adopted to accelerate loop filtering in this architecture. The architecture has been described in Verilog HDL, simulated with VCS and synthesized using 0.18μm CMOS cells library by Synopsys Design Compiler. The circuit totally costs about 38k logic gates when the working frequency is set to 150MHz. Simulation results show that the architecture can support real-time loop filter of HDTV (1280x720, 60fps) AVS video. This architecture is valuable for the hardware design of AVS CODEC.

1. INTRODUCTION

Audio Video Coding Standard Working Group of China is finalizing a national standard for the coding of video and audio. The new standard is known as AVS [1]. Comparing with similar international standards such as MPEG-2 [2], MPEG-4 [3] and H.264 [4], the advantages of AVS include higher performance, lower complexity, lower implementation cost and sample licensing. The functional blocks of AVS encoder and decoder are shown in Fig. 1 and Fig. 2 respectively.

The transformation algorithm adopted by AVS is 8x8 integer DCT (Discrete Cosine Transform). In DCT-based standards, such as H.263 [5], MPEG-2 and MPEG-4, annoying blocking artifacts rise when compression ratio is high. This effect is caused by the non-overlapping nature of the block-based DCT coding and the quantization of DCT coefficients. Nowadays, there are two kinds of techniques to deal with the blocking artifacts, one of which is to reduce their occurrence and the other is to remove them.

H.263 and MPEG-4 adopt OBMC (Overlapped Block Motion Compensation) [6] to reduce the blocking artifacts. Although OBMC works well with the blocking artifacts, its computation requirement of encoder motion estimation is much higher. Therefore, AVS adopts loop filter to remove blocking artifacts and to achieve much better subjective visual effects. The filter processing is applied after the inverse transformation and before reconstruction of the macroblock in both decoder and encoder, as shown in Fig. 1 and Fig. 2.

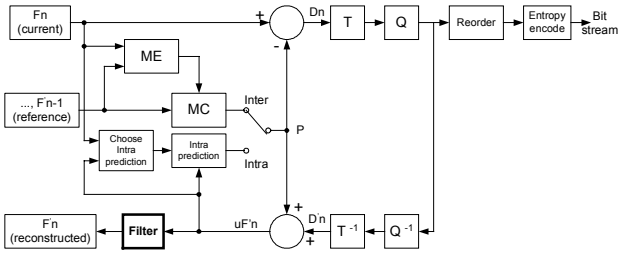


Fig. 1 Functional Blocks of AVS Encoder

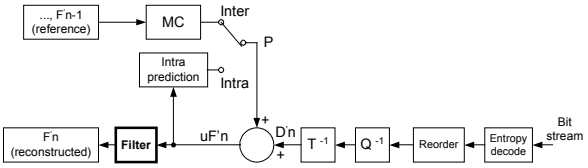


Fig. 2 Functional Blocks of AVS Decoder

The main application of AVS is HDTV playback. Going alone with a higher compression ratio, the algorithms of AVS are more complex. Its computation requirement is so high that pure software CODEC running on an ordinary processor cannot provide real-time encoding/decoding for HDTV 720p (1280x720, 60fps) video. So hardware accelerators are needed. Since AVS is a very new standard, neither hardware CODEC has been developed, nor hardware accelerator for loop filter has been described in detail. In this paper, we propose a platform-based loop filter accelerator, which is valuable for the hardware design of AVS real-time CODEC. Furthermore, the accelerator can be easily embedded into AVS CODEC SoC (System on Chip).

The remainder of the paper is organized as follows. In Section 2, algorithm of the loop filter applied by AVS will be explained. We will describe the implemented architecture of loop filter accelerator in Section 3. Simulation results and VLSI implementation will be shown in Section 4. Finally, we will draw a conclusion in Section 5.

2. ALGORITHM

In AVS, the transformation is 8x8 integer DCT, and the smallest block size of motion estimation is 8x8 too. Therefore, loop filter should be applied to all 8x8-block boundaries of a picture, except

boundaries at the edge of the picture or any boundary for which the loop filter is disabled by a special coding flag.

In AVS standard, the processing order of loop filter is described as follows. The loop filter is applied to the luminance and chrominance components separately. For each macroblock, vertical boundaries are filtered first, from left to right, and then horizontal boundaries are filtered from top to bottom, as described in Fig. 3. Pixels in macroblock above or left to the current one, which may be modified by loop filter on previous macroblocks, shall be used as input and may be further modified during the filtering of current macroblock. Pixels, modified during filtering of vertical boundaries, are used as input and may be further modified during the filtering of horizontal boundaries for the same macroblock.

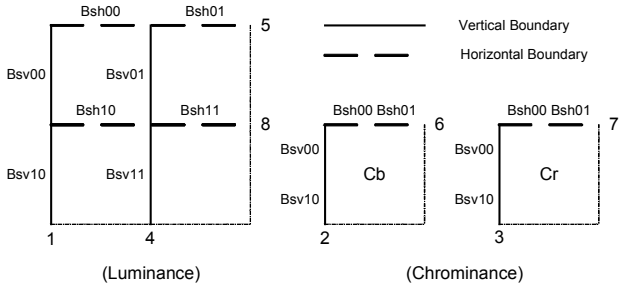


Fig. 3 AVS Filtering Order of Block Boundaries (4:2:0)

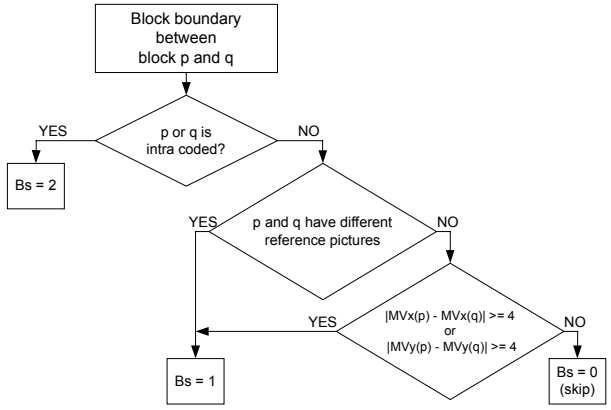


Fig. 4 Assignment of Boundary Strength (Bs)

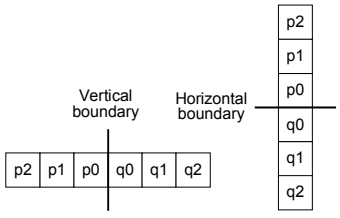


Fig. 5 Pixels across Vertical or Horizontal Boundaries

The loop filter of AVS is conditional. Its output depends on the boundary strength and the gradient of image samples across the boundary. The boundary strength parameter "Bs" is assigned as described in Fig. 4. If either of the neighboring blocks across the boundary is intra coded, the strongest filter processing is applied (Bs=2). Otherwise, if the two blocks have different reference frames or different motion vector values, a medium strength is assigned (Bs=1). When none of the previous conditions is satisfied, the filter processing is not evoked (Bs=0). The boundary strength of any chrominance component is same with that of luminance component on the same boundary, as shown in Fig.3.

Fig. 5 shows three pixels on either side of a vertical or horizontal boundary in adjacent blocks p (p0, p1, p2) and q (q0, q1, q2). Each filtering operation affects up to two pixels on either side of the boundary, which is dependent on the values of Bs and two thresholds, α and β. Here α and β are derived from two quantization parameters (QPs) of the neighboring blocks and two control parameters "alpha_c_offset" and "beta_offset". Only if the following conditions are all satisfied, a group of samples from the set {p1, p0, q0, q1} will be filtered to produce filtered output {P1, P0, Q0, Q1}.

$$Bs \neq 0, |p0 - q0| < \alpha, |p1 - p0| < \beta, |q1 - q0| < \beta$$

(a) If $Bs = 2$, P0 is produced by a 3-tap linear filtering of p1, p0 and q0, Q0 is produced by 3-tap linear filtering of q1, q0 and p0. For luminance component, if $|p2 - p0| < \beta$, P1 is produced by a 3-tap linear filtering of p1, p0 and q0. Similarly, if $|q2 - q0| < \beta$, Q1 is produced by a 3-tap linear filtering of q1, q0 and p0.

(b) If $Bs = 1$, a 4-tap filter is used to produce filtered output P0 and Q0. For luminance component, if $|p2 - p0| < \beta$, a 5-tap filtering is used to produce filtered output P1. Similarly, if $|q2 - q0| < \beta$, a 5-tap linear filtering is used to produce filtered output Q1.

For more information about the loop filter algorithm, please refer to [1].

3. IMPLEMENTED ARCHITECTURE

The implemented architecture of loop filter is designed as a platform-based accelerator, which can be easily embedded into an AVS CODEC SoC, as shown in Fig. 6.

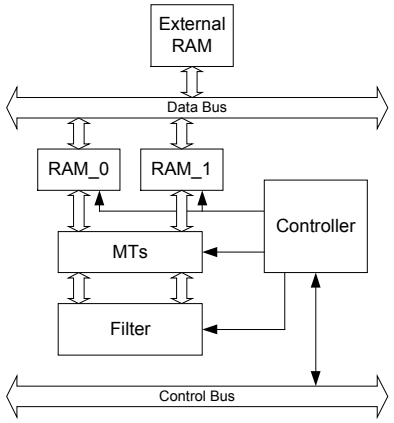


Fig. 6 Proposed Loop Filter Architecture

Because the main application of AVS is HDTV playback, the throughput of system bus is much higher than that of any other

application. Therefore, a 64-bit or even wider system data bus is necessary. Here, we assume the system data bus is 64-bit width. RAM_0 and RAM_1 in Fig. 6 are two on-chip SRAMs. In order to simplify our design, RAM_0 and RAM_1 are assigned to two-port SRAM (one read port and one write port). Before the beginning of loop filtering for a macroblock, pixels of the macroblock and some necessary pixels in its upper or left macroblock should be loaded from external RAM to the on-chip SRAMs. The two on-chip SRAMs are carefully organized, as shown in Fig. 7, to support the parallel loop filtering on a high processing speed. The Filter module is a configurable linear filter. The MTs module consists of two configurable matrix transposers, which are buffers during horizontal loop filter and transposers during vertical loop filter. The Controller module can configure Filter by coding information received from system control bus. It can also configure MTs and arrange cooperation of the sub-modules.

3.1 Organization of SRAMs

Fig. 7 shows the organization of on-chip SRAM modules. The bit width of SRAM modules is 64 bits for eight pixels, which is the same with that of system data bus. Being consistent with the common organization, we store eight pixels in a row of an 8x8-block as a 64-bit word. In order to accelerate horizontal loop filtering across vertical boundaries, we store the data of adjacent 8x8-blocks in different SRAM modules, as shown in Fig. 7. Depending on this kind of organization, we can access pixels of neighbor 8x8-blocks needed for any horizontal loop filtering at the same time. AVS loop filtering algorithm shows that the maximum number of pixels needed at each side of filtering boundaries are three. Therefore, we store only three rows of pixels for the upper 8x8-blocks, labeled as U2, U3, U4 and U5 in Fig. 7.

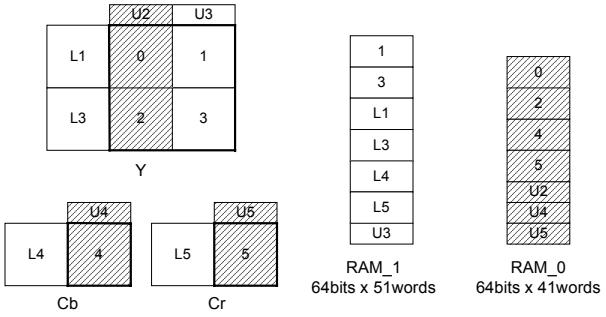


Fig. 7 Organization of on-chip SRAMs

3.2 Loop Filtering Order

According to AVS requirements of loop filtering order, which are presented in Section 2, we adopt the filtering order of boundaries described in Fig. 8.

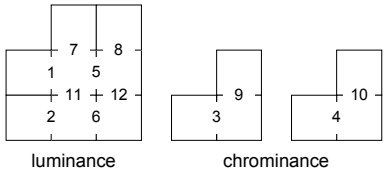
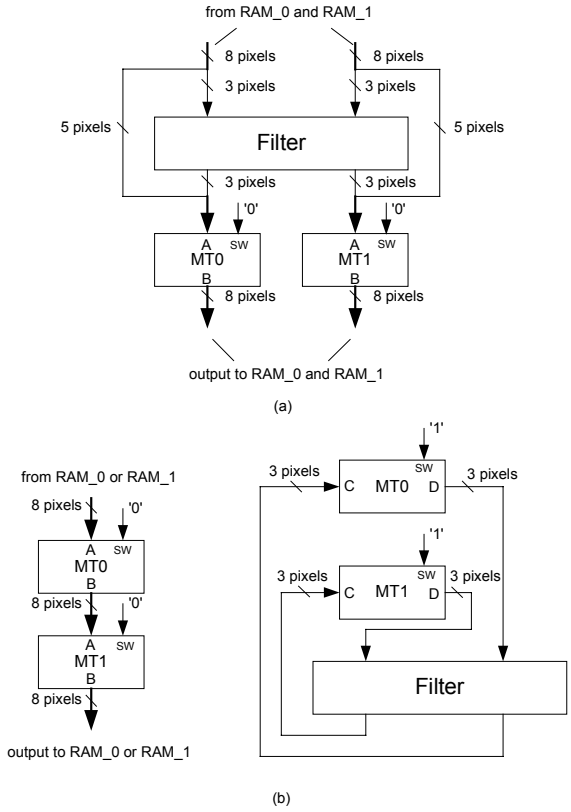


Fig. 8 Adopted Filtering Order of 8x8-Block Boundaries

3.3 Loop Filtering Procedure

Fig. 9 shows the implemented architecture of loop filter. The Filter is a conditional linear filter, with six pixels input and six pixels output (8 bits/pixel), which has been described in Section 2. MT0 and MT1 are two configurable matrix transposers, as shown in Fig. 10. One matrix transposer is a 3x8-cell array, where each cell is an 8-bit register. Each matrix transposer has two input ports (A and C) and two output ports (B and D). Port A and B are 64-bit width. Port C and D are 24-bit width. The input "sw" is a switcher, taking charge of selection for data path.



(a) Architecture for Horizontal Loop Filter
(b) Architecture for Vertical Loop Filter
Fig. 9 Configurable Architecture

Fig. 9 (a) is the architecture for horizontal loop filter. Firstly, pixels are read out from RAM_0 and RAM_1, and sent to Filter directly. Then, the filtered pixels are written to MT0 and MT1, which are two buffers functionally. When the buffers are full, Controller will stop reading and filtering the unfiltered pixels from RAM_0 and RAM_1, and write back the filtered pixels in MT0 and MT1 to RAM_0 and RAM_1 respectively. MT0 and MT1 can buffer at most three rows of 8-pixel data respectively. Therefore, to filter a vertical boundary of an 8x8-block, the procedure of reading (filtering) and writing should be repeated three times. For each procedure, it takes 3 cycles to read and filter pixels, and another 3 cycles to write them back. If the depth of the two buffers is four rather than three, the procedure should be repeated only twice. However, the matrix transposer must be a 4x8-cell array, which will cost extra 128-bit register and enhance the difficulty of place and route. Therefore, in the implemented design, we finally

used two 3x8-cell arrays as the buffer.

Fig. 9 (b) shows the architecture for vertical loop filter. Firstly, six rows of 8-pixel data, that is three rows on each side of a horizontal boundary, are loaded into MT0 and MT1 from RAM_0 or RAM_1 row by row. Then, the switchers of MT0 and MT1 are toggled and six pixels in a column (across a horizontal boundary) are sent to Filter at a time. Filtered pixels are stored back to MT0 and MT1 respectively. After all pixels in MT0 and MT1 are filtered, the switchers are toggled again and the filtered pixels are stored back to RAM_0 or RAM_1. To filter a horizontal boundary of an 8x8-block, it takes 6 cycles to load pixels, 8 cycles to filter, and 6 cycles to store them.

A finite state machine in Controller can arrange the order of loop filter according to Fig. 8.

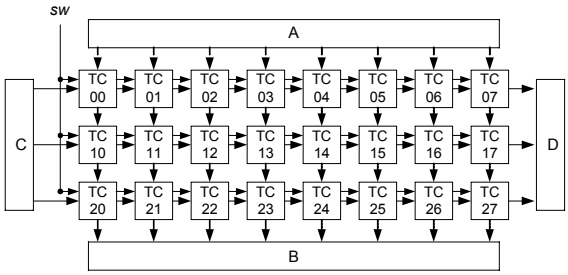


Fig. 10 Architecture of the Matrix Transposer

Before Loop Filtering (QP=41, PSNR=33.241)

After Loop Filtering (QP=41, PSNR=33.472)

Fig. 11 Loop Filter's Effects

In conclusion, it takes 92 cycles to load unfiltered pixels from external RAM to on-chip SRAM RAM_0 and RAM_1, (3+3) x3x6=108 cycles to filter in horizontal direction, (6+8+6) x6=120 cycles to filter in vertical direction, and 92 cycles to store filtered pixels to external RAM via system data bus. In addition, about 48 cycles are needed to load coding information. To sum up, the number of total cycles for loop filtering each macroblock is 460.

4. SIMULATION RESULTS

We described the design mentioned above in Verilog HDL at RTL level, which is synthesizable. According to AVS1.0 verification model [7], a C-program model of loop filter was also developed to generate input simulation vectors for VCS digital simulator. By testing with eight HDTV (1280x720, 60fps) bitstreams (100 frames per bitstream), VCS simulation results show that our Verilog code is functionally identical with the loop filter of AVS1.0 verification model. Fig. 11 shows effects of our loop filter on the first frame (Intra) of "Foreman" sequence. Subjective views and PSNR values are all improved obviously.

The validated Verilog code was synthesized using 0.18μm CMOS cells library by Synopsys Design Compiler. The circuit totally costs about 38k logic gates when the working frequency is set to 150MHz. Table 1 is our synthesized results. The implemented architecture costs 460 cycles to perform loop filter for each macroblock, which is sufficient to realize the real-time loop filter for HDTV (1280x720, 60fps) AVS bitstreams.

Table 1 Synthesized Results

Technology	0.18μm
Working Frequency	150MHz
Gate Count (Without SRAM)	25K
SRAM	13K
Cycles/MB	460
Capacity	1280x720 90.58fps
Data Bus Bandwidth (Mbytes/s) 1280x720 30Hz 1280x720 60Hz	 158.976 317.952

5. CONCLUSIONS

In this paper, we implemented a platform-based accelerator for the loop filter of AVS standard. Firstly, we described the algorithm of AVS loop filter. Then the architecture was proposed. Our main idea is to use column-separated SRAM organization and two configurable matrix transposers to accelerate loop filtering. Finally, we gave out simulation results. The architecture was synthesized using 0.18μm CMOS cells library by Synopsys Design Compiler. The synthesized results show that our design can support real-time loop filter of HDTV (1280x720, 60fps) AVS video. The architecture is valuable for the hardware design of AVS CODEC.

6. ACKNOWLEDGEMENT

This work has been supported by National Hi-Tech Development Programs of China under grant No. 2003AA1Z1290.

7. REFERENCES

[1] Audio Video Coding Standard FCD Part 2: Video, Dec. 2003.
[2] ISO/IEC IS 13818, "General Coding of Moving Picture and Associated Audio Information," 1994.
[3] ISO/IEC FCD 14496, "Information technology – Coding of audio-visual objects – Part 2: Visual," July 2001.
[4] 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
[5] ITU-T, Draft ITU-T Recommendation H.263, "Video Coding for Low Bit Rate Communication," 1997.
[6] M.T. Orchard and G. J. Sullivan, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach," IEEE Trans. Image Processing, pp. 693-699, September 1994.
[7] AVS1.0 RM5, December 2003.