AN EFFICIENT ARCHITECTURE FOR DEBLOCKING FILTER IN H.264/AVC VIDEO CODING

Chung-Ming Chen
Department of Electrical Engineering &
Institute of Computer and Communication Engineering
National Cheng Kung University
Taiwan, R.O.C.
cmchen@casmail.ee.ncku.edu.tw

Chung-Ho Chen
Department of Electrical Engineering &
Institute of Computer and Communication Engineering
National Cheng Kung University
Taiwan, R.O.C.
chchen@mail.ncku.edu.tw

ABSTRACT
In this paper, we propose an efficient architecture for the adaptive deblocking filter in H.264/AVC video coding standard. We use eight forwarding shift register arrays (of which each contains 4×4 8-bit shift registers) with two transposing operations and two filter units to support simultaneous processing of the horizontal and vertical filtering. The proposed architecture is called "Pipeline Buffer Shift Register (PBSR)." As a result, the performance of PBSR is 22.5% faster than the advanced architecture of the previous proposal. Moreover, the number of total memory references is reduced to 37% and 75% respectively compared to the basic and advanced architectures of the previous proposals.

KEY WORDS
Deblocking Filter, H.264/AVC, Memory Reference, Video Coding.

1. Introduction

The new video coding standard Recommendation H.264 of ITU-T [1] also known as International Standard 14496-10 or MPEG-4 Part 10 Advanced Video Coding (AVC) of ISO/IEC [2], significantly outperforms the previous ones [3] in bit-rate reduction. The functional blocks of H.264/AVC, as well as their features, are shown in Figure 1. Preliminary studies [4] using the software base of this new standard, suggest that H.264 offers up to 50% better compression than MPEG-2 and up to 30% better than H.263+ and MPEG-4 advanced simple profile.

The block-based structure of the H.264/AVC architecture produces artifacts known as blocking artifacts. These blocking artifacts can be occurred both due to the quantization of the transform coefficients and block-based motion compensation. To reduce the blocking artifacts, the overlapped block motion compensation (OBMC) [5] is adopted into the H.263 standard [3]. Unlike the OBMC in H.263, H.264/AVC uses an adaptive deblocking filter [6] that has been shown to be a more powerful tool in reducing artifacts and improving the video quality. Adaptive deblocking filter can also be used in inter-picture prediction to improve the ability to predict other picture as well. Since it is within the motion compensation prediction loop, the deblocking filter is often referred to as an "in-loop filter". As a result, the filter reduces the bit rate typically by 5-10% while producing the same objective quality as the non-filtered video [7]. A detailed description of the adaptive deblocking filter can be found in [6].

The filter described in the H.264/AVC standard is highly adaptive. Several parameters and thresholds, as well as the pixel characteristics of the picture itself, control the boundary strength of the filtering process. These issues are also equally challenging during parallel processing under DSP or SIMD computational architectures. Due to intensive computations, in [8] and [9] dedicated hardware was developed for acceleration. And our proposed architecture outperforms the previous proposals through significant reduction in memory reference.

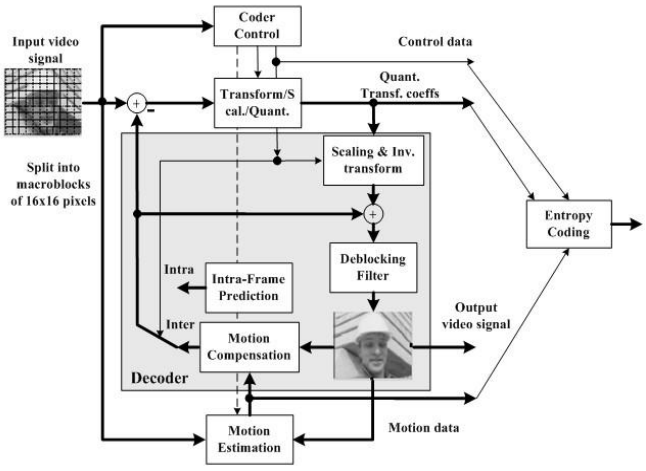

Figure 1: Block Diagram of H.264/AVC

The organization of this paper is as follows: In Section 2, the algorithm used in the deblocking filter is explained.

Section 3 illustrates the block diagram of the proposed architecture and functionality of each module. Section 4 shows the simulation results. Finally, conclusion is presented in Section 5.

2. The Algorithm of Deblocking Filter

For each luminance macroblock, the left-most edge of the macroblock (V1, V2, V3, and V4) is filtered first, followed by the other three internal vertical edges from left to right. Similarly, the top edge of macroblock (H17, H18, H19, and H20) is filtered first, followed by the other three internal horizontal edges from top to bottom. Chrominance filtering follows a similar order in each direction for each 8x8 chrominance macroblock as shown in Figure 2 ("V" denotes a vertical edge and "1" the first block cycles while "H" denotes a horizontal edge and "17" the seventeenth block cycle).

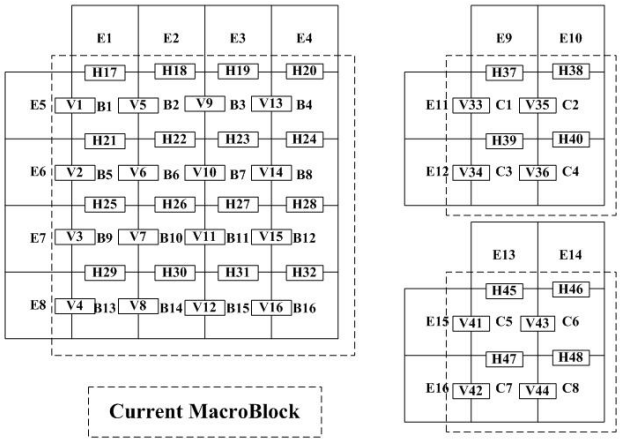

Figure 2: Processing Order of Standard

On the sample processing level, sample value and quantization parameter threshold can turn on/off the filtering for each individual sample. For example, Figure 3 illustrates the principle of the deblocking filter using a one-dimensional visualization of a block edge in a typical situation where the filter would be turned on. Whether the samples p0 and q0 as well as p1 and q1 are filtered is determined by using the quantization parameter (QP), dependent threshold α(QP), and β(QP). Thus filtering of p0 and q0 only takes place if each of the following condition is satisfied:

1. Bs != 0
2. |p0 - q0| <α(QP)
3. |p1 - p0|< β(QP) and |q1 - q0| < β(QP).

Where the β(QP) is considerably smaller than α(QP).
Hence, filtering of p1 or q1 take place if the corresponding condition below is satisfied:

|p2-p0| < β(QP) or |q2-q0 |< β(QP)

The dependency of α and β on the quantization parameter, link the strength of filtering to general quality of the reconstructed picture prior to filtering. For small quantization values, the thresholds both become zero, and filtering is effectively turned off altogether.

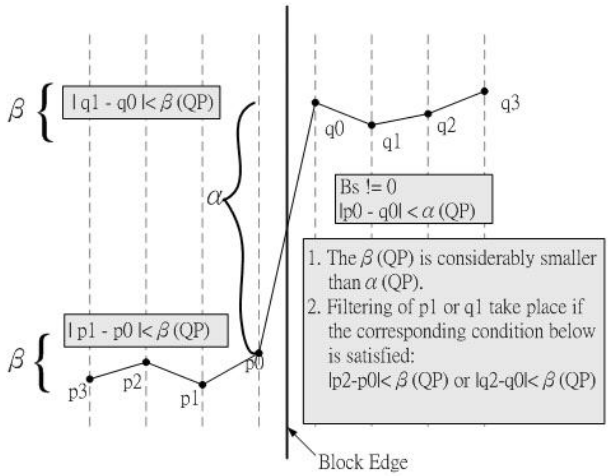

Figure 3: Principle of Deblocking Filter

The basic idea is that if a relatively large absolute difference between samples near a block boundary is estimated, it is quite likely to be a blocking artifact and should therefore be smoothed. However, if the amplitude of that difference is so large that it can no longer be explained an artifact produced by the quantization and motion compensation, the edge is more likely to be the actual behavior of the source picture and should not be filtered.

3. Proposed Architecture

3.1 Edge Filtering Operation

The complexity of H.264/AVC deblocking filter is mainly due to two reasons. The first one is the need of highly adaptive filtering, which requires several conditional processing on each block edges and sample levels. As the described in the previous section, the threshold value of α and β, the table-derived operations, and edge filtering operation are known to be very time consuming. These issues are also equally challenging during parallel processing under DSP or SIMD computational architectures. Therefore, we propose an efficient filtering unit that combines these parameters into edge filter to accelerate the horizontal and vertical filtering on the boundary of two adjacent basic 4x4 blocks as shown in Figure 4. The filtering operation is performed when previous content activity check is satisfied. The filtering operation can be divided into two modes. A special mode of filtering that allows for stronger filtering is applied when Bs is equal to 4 (Luma4 and Chroma4). The others are standard mode of filtering with Bs parameter from 1

to 3 (Luma3_1 and Chroma3_1). The chrominance is the same as luminance.

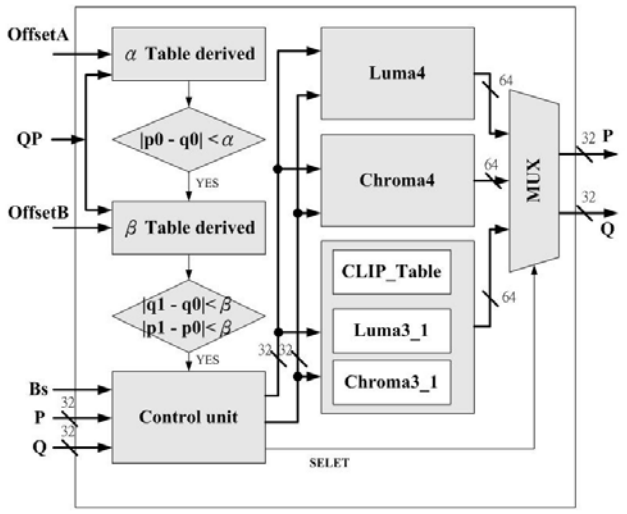

Figure 4: Edge Filtering Architecture

Another reason for the high complexity is the small block size employed for residual coding in the H.264/AVC video coding algorithm. With the 4x4 blocks and a typical filter length of 2 samples in each direction, each sample in a picture must be loaded/stored from/to memory 4 times; either to be modified or to determine if the neighboring samples will be modified. In order to reduce the numbers of memory reference and accelerate the overall system performance, we propose another efficient architecture as shown in Figure 5, which can perform simultaneous processing of horizontal and vertical filtering and reduce the number of access time of each block to one.

3.2 Alternative Processing Order

Our architecture utilizes an alternative processing order, which allows the simultaneous processing of horizontal and vertical filtering as shown in Figure 6. The processing order begins from V1 to V2 ("V" denotes a vertical edge and "1" the first block cycle). And then at the third block cycle, the vertical edge V3 and horizontal edge H3 ("H" denotes a horizontal edge and "3" the third block cycle) are simultaneously processed. Then V4, H4 follows, so on and so forth. This processing order enables concurrent horizontal and vertical filtering to reduce memory access by reusing data through forwarding in the PBSR arrays.

3.3 Components of the Proposed Architecture

There are three major functions in our proposed architecture PBSR. The first component is the Shift Operation Array. There are eight forwarding shift register arrays in our proposed architecture (for example, Array1, 2, 4, 5, 6, 7, 8, and 9). Each array has four entries which contain 4 processed samples. The shift direction is as shown in Figure 5. The second function of our proposed architecture is the transposing operation as shown in Figure 5. The Array3 and Array10 latched the 4x4 block sample values that are transposed from Array 2 and Array9 respectively. And the final important functions are the horizontal and vertical filter units. The operation of these components is described in details below.

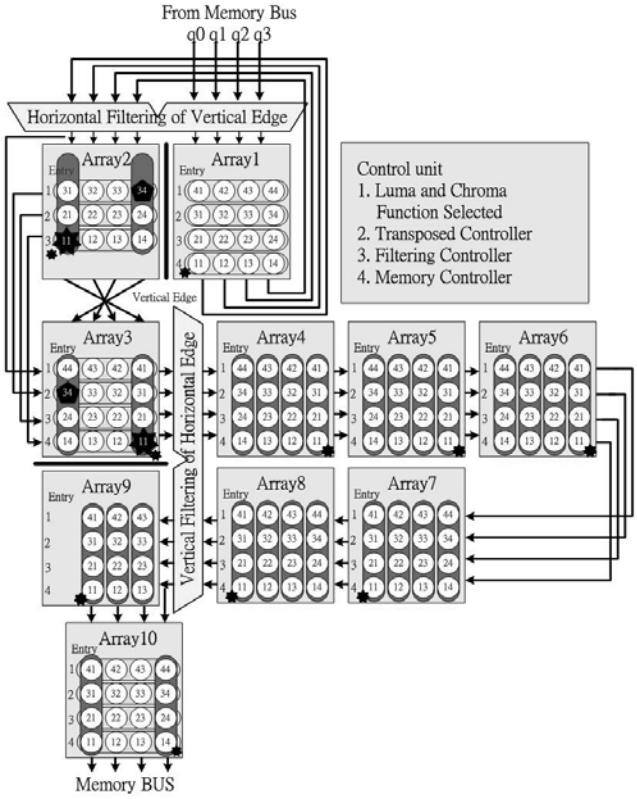

Figure 5: Proposed Architecture PBSR

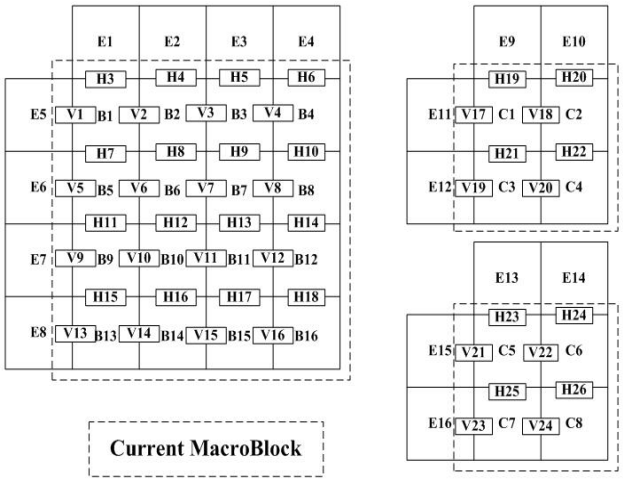

Figure 6: Alternative Processing Order

Horizontal Filter Unit: The function of horizontal filter is to filter the vertical edge across the boundary of the two adjacent blocks. The detailed design of the edge filter is based on [6]. The input parameters to this function are as follows.

- A set of samples which denoted q0, q1, q2, and q3 are transferred from the memory bus.
- Another set of samples which denoted p0, p1, p2, and p3 are shifted from the fourth entry of Array1 which has just completed the first horizontal filtering.
- Other input values are filtering parameter, such as boundary strength (Bs), quantization parameter (QP), α, and β offset (Offset A and Offset B).

With horizontal filtering, each basic 4x4 block needs to be filtered two times across the vertical edge of the left and right boundary of the current basic 4x4-block. Similarly, the first reference denoted q samples (which are transmitted from the memory bus) and the second reference denoted p samples (which are shifted from the fourth entry of Array1) as shown in Figure 5.

Vertical Filter Unit: The function of vertical filter is to filter the horizontal edge across the boundary of two adjacent blocks. The input parameters of vertical filter unit are the same as horizontal filter unit, except for the q samples which are shifted from Array3 and the p samples which are shifted from Array8.

The Transposed Unit: Those filtered samples of Array2 are transposed and latched into Array3 at the fourth cycle of the block cycles as shown in Figure 7. The function of Array10 is the same as Array3 except it latches the filtered samples from Arrar9.

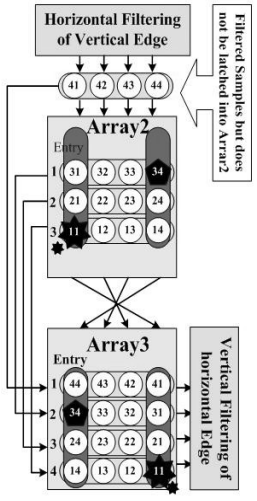

Figure 7: Transposing Operation

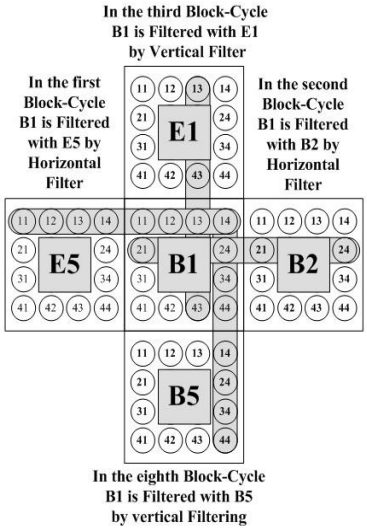

Figure 8: Current Block B1 Processing Order

Memory Interface: There are two 32-bit memory data buses for the proposed architecture. One data bus is used to input unfiltered samples from memory to PBSR. The other is used to output filtered samples from PBSR to memory.

Parameter Interface: Parameters used in adaptive deblocking filtering are transmitted from memory to the horizontal and vertical filter respectively. These parameters include boundary strength (Bs) and average quantization parameter (QP) for edge level adaptive, OffsetA and OffsetB for slice level adaptive.

Control Unit: The control unit has two major functions. The first is that it exactly controls the data flow. The other provides correct parameters to the horizontal and vertical filter.

3.4 Data Flow

The data flow of the proposed architecture is shown in Table 1. Initially, assume that the blocks of E1, E2, E3, E4, and E5 are processed in Array6, Array5, Array4, Array3, and Array1 respectively. In the first block-filtering cycle (1 block-filtering cycle = 4 clock cycles), the sample values of current 4x4-block B1 are transmitted from memory to PBSR and filtered with block E5 by the horizontal filter. In next block-filtering cycle, the current block B1 is filtered with B2 by the horizontal filter. In the third block-filtering cycle, the proposed architecture PBSR can simultaneously process horizontal filtering of vertical edge V3 (the boundary of block B1 and B2) and vertical filtering of horizontal edge H3 (the boundary of block E1 and B1). In the eighth block-filtering cycle, the vertical filtering of horizontal edges (boundary of bock B1 and B5) are performed as shown in Figure 8. Finally, PBSR writes the block B1 to memory at the ninth block-filtering cycle.

Table 1: Data Flow in the PBSR

State	Block-Filtering Cycle							
	0	1	2	3	4	5	6	7
Array1	E5	B1	B2	B3	B4	E5	B5	B6
Array2	E4	E5	B1	B2	B3	B4	E5	B5
Array3	E4	E5	B1	B2	B3	B4	E5	B5
Array4	E3	E4	E5	B1	B2	B3	B4	E5
Array5	E2	E3	E4	E5	B1	B2	B3	B4
Array6	E1	E2	E3	E4	E5	B1	B2	B3
Array7		E1	E2	E3	E4	E5	B1	B2
Array8			E1	E2	E3	E4	E5	B1
Array9				E1	E2	E3	E4	E5
Array10				E1	E2	E3	E4	E5
MEM					E1	E2	E3	E4

4. Result

4.1 Memory Reference

As ITU-T recommendation [1], an adaptive filtering shall be applied to all 4x4 block edges of a picture, except for the edges at the boundary of the picture. Therefore, most of 4x4 blocks need to be filtered 4 times with the adjacent

blocks (left, right, top, and bottom as shown in Figure 8). As a result, the number of total memory reference for a luminance macroblock, including read and write, is 4x4x2x16=512. For a picture in QCIF format, the number of total memory accesses is 49408 (Assuming the memory bus width is 32 bits).

As described in the previous section, the key feature of our proposed architecture is the utilization of the Pipeline Buffer Shift Register (PBSR) to reduce memory references and simultaneously process the horizontal and vertical filtering. As a result, the number of total memory reference for a luminance macroblock is reduced to 192. It is clear that the number of memory references is reduced to 37% of the software version. In other words, the memory performance of our scheme increases 3 times when compared to software implementation. Table 2 shows the comparisons of memories (Luminance only) with the previous proposals in [8]. Our architecture can save more than 63% of the memory bandwidth compared to the basic of the previous schemes. Hence, our architecture is able to significantly reduce power consumption.

Table 2: Memory Reference per Macroblock

Author	Architecture	MEM Ref
[8]	Basic+Single-port SRAM	512
[8]	Advance+Dual-port SRAM	256
[8]	Basic+Two-port SRAM	512
[8]	Dual Arrays+Two-port SRAM	256
PBSR	Dual-port SRAM or Two Single port SRAM	192

4.2 Performance

The first set of samples for luma/chroma macroblock is completed in 32/24 clock cycles and written into the memory at the next cycle. The next set of samples will complete filtering and be written into the memory one cycle at a time. Therefore, the total number of cycles for filtering one luma and two chroma macroblocks takes 32+96 = 128 and (24+32)x2 = 112 cycles respectively. As a result, the total filtering takes 240 cycles for a luma and two chroma macroblocks. Our filtering scheme takes less number of cycles when compared to 294 cycles of the architecture described in [8]. Table 3 illustrates the cycle count required for each different implementation.

Table 3: Memory Cycles per MB

Author	Architecture	Cycles/MB
[8]	Basic+Single-port SRAM	558
[8]	Advance+Dual-port SRAM	494
[8]	Basic+Two-port SRAM	462
[8]	Dual Arrays+Two-port SRAM	294
PBSR	Dual-port SRAM or Two Single port SRAM	240

5. Conclusion

In this paper, we propose an efficient architecture to accelerate the operations of deblocking filter for H.264/AVC video coding and implement it in Verilog HDL. The major idea is to reduce the number of memory references through the PBSR approach and simultaneously perform horizontal and vertical filtering. Simulation results show that the processing capability of the proposed architecture is very appropriate for real-time deblocking of 1280x720 30Hz video operating at 50-100MHz. According to the simulation results, our design is a good choice of deblocking filter for the platform-based design under H.264/AVC coding systems.

6. Acknowledgements

The work in this paper is in part supported by the National Science Council, Taiwan ROC, under NSC 93-2220-E-006-004. In addition, the authors thank Elan Microelectronics CORP. for support in VLSI design flow and simulation environment.

References

[1] ITU-T Recommendation H.264, *Advanced video coding for generic audiovisual services*, 2003.

[2] ISO/IEC 14496–10:2003, *Coding of Audiovisual Objects—Part 10: Advanced Video Coding*, 2003.

[3] ITU-T Recommend H.263, *Video Coding for Low Bit Rate Communication*, 1998.

[4] Jorn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi, Video Coding with H.264/AVC: Tools, Performance, and Complexity, *IEEE Circuit and Systems Magazine*, 2004, 7-28.

[5] M.I T. Orchard and G.J. Sullivan, Overlapped Bock Motion Compensation: An Estimation-Theoretic Approach, *IEEE Transactions on Image Processing*, 1994, 693-699.

[6] Peter List, Anthony Joch, Jani Lainema, Gisle Bjøntegaard, and Marta Karczewicz, Adaptive Deblocking Filter, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, 614-619.

[7] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, H.264/AVC baseline profile decoder complexity analysis, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, 715–727.

[8] Yu-Wen Huang, To-Wei Chen, Bing-Yu Hsieh, Tu-Chih Wang, Te-Hao Chang, and Liang-Gee Chen, Architecture Design for De-blocking Filter in H.264/JVT/AVC. *Proc. IEEE Conf. on Multimedia and Expo*, 2003, 693-696.

[9] Miao Sima, Yuanhua Zhou, and Wei Zhang, An Efficient Architecture for Adaptive Deblock filter of H.264/AVC Video Coding, *IEEE Transactions on Consumer Electronics*, Vol. 50, 2004, 292-296.