Proceedings of the Third IASTED International Conference
CIRCUITS, SIGNALS, AND SYSTEMS
October 24-26, 2005, Marina del Rey, CA, USA

AN EFFICIENT VLSI ARCHITECTURE FOR EDGE FILTERING IN H.264/AVC

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 study and analyze the computational complexity of H.264/AVC baseline profile decoder based on SimpleScalar/ARM simulator. The simulation result shows that the memory reference, the operations of content activity check, and the edge filtering are known to be very time consuming in the embedded system. In order to reduce the memory reference and improve overall system performance, we proposed a new efficient VLSI architecture to accelerate the processing of deblocking filter. The proposed architecture is called "Adaptive Edge Filtering Operation (AEFO)," which could be embedded in a platform-based architecture as a co-processor. As a result, the performance of the embedded system using AEFO is 1.66 times faster than software implementation. Moreover, the number of total memory references for loading and storage is reduced by 34% and 36% respectively.

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

1. Introduction

Video compression is the critical component in today's multimedia systems. The limited transmission bandwidth or storage capacity for applications such as DVD or digital television, and internet video streaming stresses the demand for higher video compression rates. To meet this demand 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] has been developed. It significantly outperforms the previous one (H.263) [3] in bit-rate reduction. The functional blocks of H.264/AVC, as well as their features, are shown in Figure 1.

As previously studied [4], the most time consuming parts of H.264/AVC decoder is deblocking filter. Therefore, this paper focuses on adaptive deblocking filter to remove coding artefacts around block edges. These blocking artifacts can occur from both quantization of the transform coefficients and block-based motion compensation. In order to reduce the blocking artifacts, the overlapped block motion compensation (OBMC) [5] is adopted into H.263 standard. Unlike the OMBC in H.263, H.264/AVC adopts an adaptive deblocking filter [6] that has shown to be a more powerful tool in reducing artifacts and in 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].

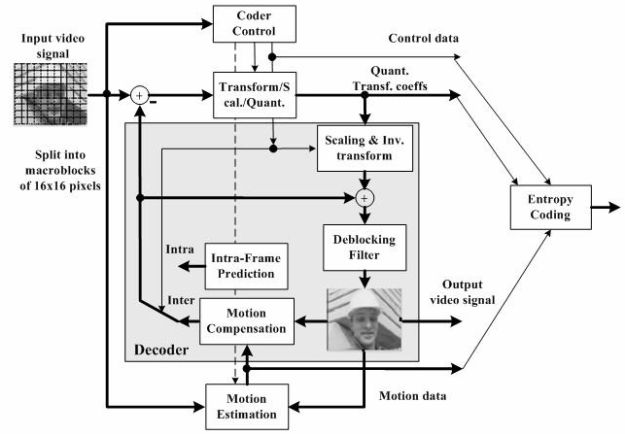

Figure 1: Block Diagram of H.264/AVC

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 architecture. Due to intensive computations, dedicated hardware was developed for

acceleration in [8] and [9]. But these proposals did not mention or embed the computation of boundary strength (Bs), the table-derived operations, and the content activity check operations in the VLSI architecture of edge filter. In order to reduce the number of total conditional processing operations and improve overall system performance. We proposed a VLSI architecture that embedded the computation of boundary strength (Bs), the table-derived operations, and several conditional processing such as the threshold value of Alpha and Beta in the edge filtering unit. As a result, our proposed architecture can outperform the software implementation of H.264/AVC codec.

The organization of this paper is as follows: In Section 2, the algorithm of the deblocking filter is explained. Section 3 analyzes the computational complexity of H.264/AVC baseline decoder. Section 4 illustrates the block diagram of the proposed architecture and functionality of each module. Section 5 shows the simulation result. Finally, conclusion is presented in Section 6.

2. The Algorithm of Deblocking Filter

For each luminance macroblock, the left-most edge of the macroblock is filtered first, followed by the other three internal vertical edges from left to right. Similarly, the top edge of macroblock 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.

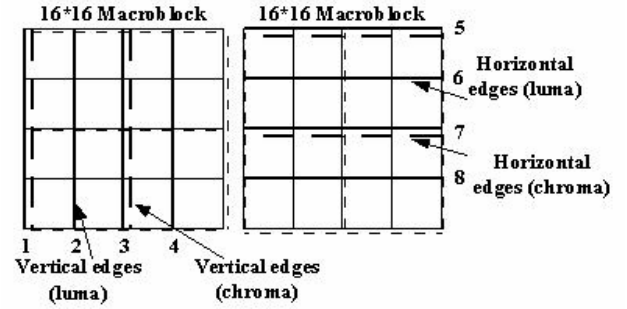

Figure 2: Edge Filtering Order

On the sample processing level, content of samples and quantization parameter threshold can turn on/off the filtering for each individual boundary. 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 turn on. Whether the samples p0 and q0 as well as p1 and q1 are filtered is determined by using quantization parameter (QP), dependent threshold Alpha(QP) and Beta(QP), and content of a set of sample. Thus filtering of p0 and q0 only takes place if each of the following condition is satisfied:

$$Bs \mathrel{!}= 0 \quad (1)$$

$$|p0 - q0| < Alpha(QP) \quad (2)$$

$$|p1 - p0| < Beta(QP) \text{ and } |q1 - q0| < Beta(QP) \quad (3)$$

Where the Beta(QP) is considerably smaller than Alpha(QP). Accordingly, filtering of p1 or q1 will take place if the corresponding condition below is satisfied:

$$|p2-p0| < Beta(QP) \text{ or } |q2-q0| < Beta(QP) \quad (4)$$

The dependency of Alpha and Beta on the quantizer, links the strength of filtering to general quality of the reconstructed picture prior to filtering. For small quantizer values the thresholds both become zero, and filtering is effectively turned off altogether.

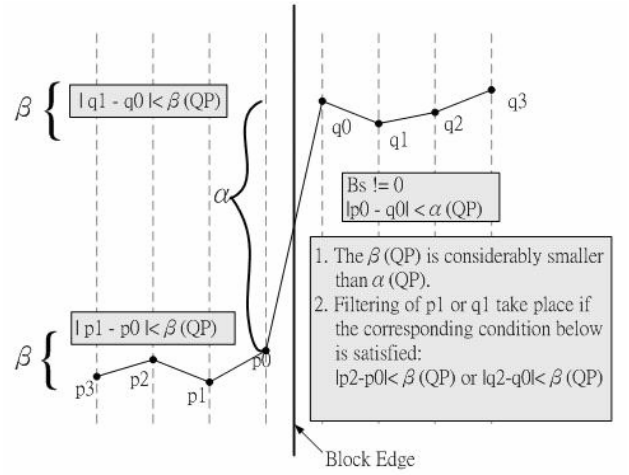

Figure 3: Principle of Deblocking Filter

The basic idea is that if a relatively large absolute difference between samples near a block edge is measured, it is quite likely to be a blocking artifact and should therefore be reduced. However, if the magnitude of that difference is so large that it can no longer be explained by the coarseness of the quantization used in the encoding, the edge is more likely to reflect the actual behavior of the source picture and should not be smoothed over.

3. Computation Complexity

One of the most important issues in computational complexity of H.264/AVC decoder is the distribution of time complexity among its major sub-function. In our simulation result, as shown in Table 1, deblocking filtering (36%) is the largest component, followed by interpolation (22%), and bitstream parsing and entropy decoding (13%), and inverse transfers and reconstruction (13%).

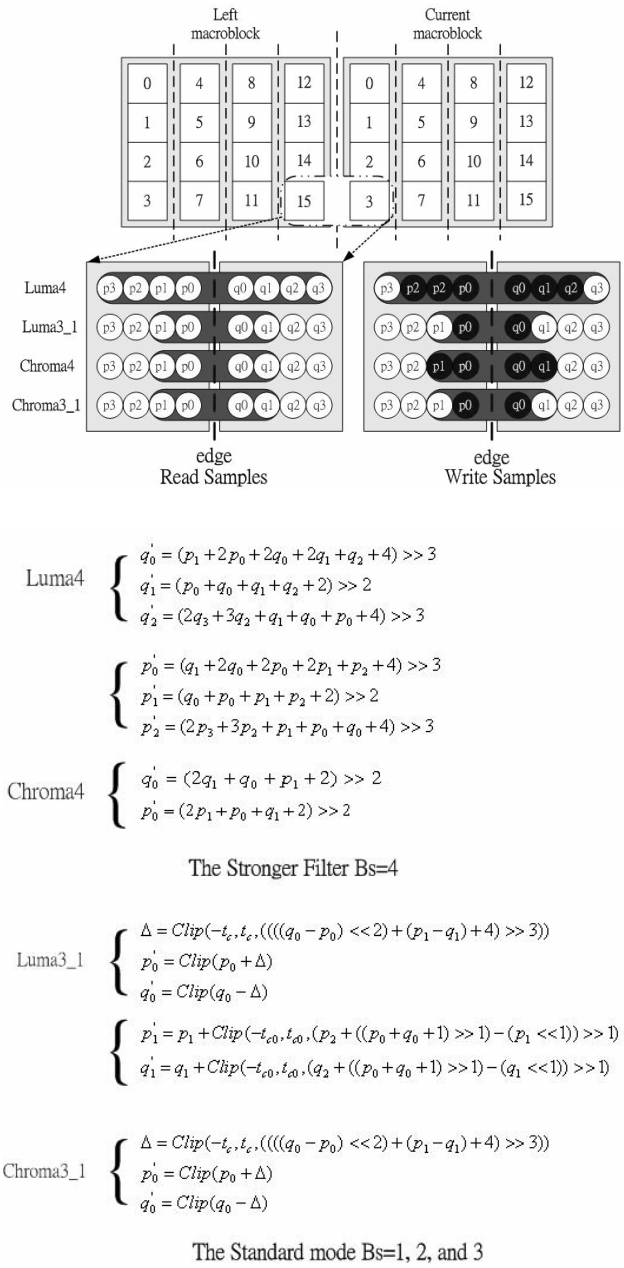

Figure 4: The Filtering Operations

Table 1: The Computational Complexity of Decoder

Item	Function	Complexity
1.	Deblocking Filtering	36%
2.	Interpolation	22%
3.	Entropy Coding	13%
4.	Inverse Transfers and Reconstruction	13%

As our experiment result indicates, the operation of the deblocking filter, which is the most time consuming parts of H.264/AVC decoder, can be separated into two major sub-functions. The first sub-function is the computation of the "Boundary Strength" (Bs) parameter for each edge filter operation. The purpose of this computation is to determine whether a block artifact may have been produced across the boundary, and thus determine the strength (Bs) of the filter to be used on the edge. A Boundary Strength (Bs) is assigned an integer value from 0 to 4. A strongest filter (Bs=4) is used if one or both sides of edges are intra coded and the boundary is a macroblock boundary, whereas a value of 0 means no filtering is applied on this specific edge. In the standard mode of filtering which is applied for edges with Bs from 1 to 3, the value of Bs affects the maximum modification of the sample values that can be caused by filtering. Table 2 shows how the value of Bs depends on the modes and coding conditions of the two adjacent blocks. In the table, conditions are evaluated from top to bottom, until one of the conditions holds true, and the corresponding value is assigned to Bs.

Table 2: The Filter Strength Bs

Bs	Block Modes and Conditions
4	One of the blocks is Intra and the edge is a macroblock edge
3	One of the blocks is Intra
2	One of the blocks have coded residuals
1	1. Difference of block motion >= 1luma sample distance 2. Motion compensation from different reference frames
0	Otherwise

The second important sub-function is the content activity check and filtering operations as shown in Figure 3 and Figure 4 respectively. In order to separate the true edge and blocking artifact, the sample values across every edge to be filtered are analyzed. As stated in Section 2, filtering does not take place for edges with Bs equal to zero. For edges with nonzero Bs values, a pair of quantization-dependent parameters, referred to as Alpha and Beta, are used in the content activity check that determines whether each set of samples is filtered. Both table-derived threshold Alpha and Beta are dependent on the average quantization parameter (QP) employed over the edge, as well as encoder selected offset values that can be used to control the properties of the deblocking filter on the slice level.

The filtering operations and the content activity checks which require conditional processing on the block edge and sample level, are known to be very time consuming and are also equally challenging for parallel processing in DSP or SIMD computing architecture. In order to reduce the number of conditional operations and improve the overall system performance, we submit herewith (see next sections) a proposed VLSI architecture that includes computation of boundary strength, table-derived operations (threshold Alpha and Beta), and content activity check in the edge filtering unit.

4. Proposed Architecture

In order to reduce the number of memory reference and branch operations, and then improve overall system performance, we proposed an efficient VLSI architecture that embeds the computation of boundary strength, the table-derived operations, content activity check, and filtering operations in the edge filtering unit which is called "Adaptive Edge Filtering Operation (AEFO)" as shown in Figure 5. There are five major sub-functions in our proposed VLSI architecture as described below.

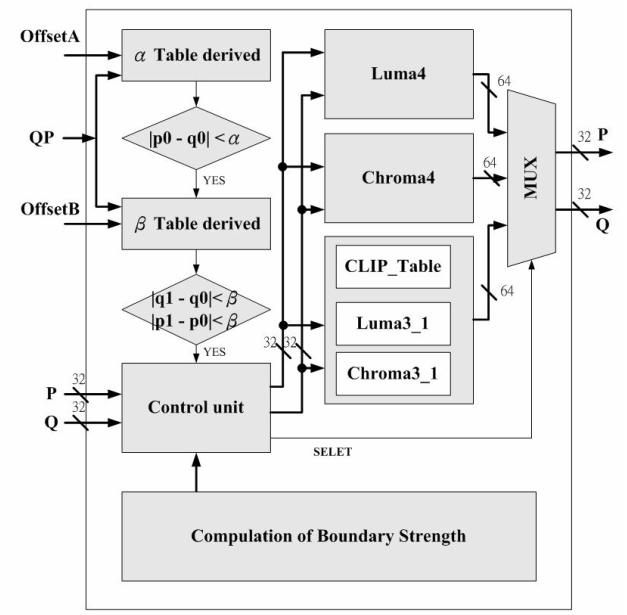

Figure 5: Adaptive Edge Filtering Architecture

The Computation of Boundary Strength: The purpose of this computation is to determine whether a block artifact may have been produced across the boundary, and thus determine the appropriate strength (Bs) of the filter to be used on the edge. A detailed description of the computation of boundary strength can be found in Section 3 or [6].

The Filtering Operation: The most important function of deblocking filter is the filtering operation, which is divided into two modes. A special mode of filtering that allows for stronger filtering is applied when Bs is equal to 4. The others are standard mode of filtering with a Bs parameter of 1 to 3 as shown in Figure 4.

Clipping Operation: The filtering operation would result in too much low-pass filtering (blurring). A significant part of the adaptive filter is obtained by limiting these values. This process is called clipping. There are eight clipping operations in our proposed architecture as shown in Figure 4. A detailed description of the clipping operation can be found in [1].

***Content Activity Check Operation*:** Conditional branches which are described below almost inevitably appear in the inner most loops of the algorithm. The major content activity checks (conditional branches) are listed below and described in Section 2

Content Activity Check for p0 and q0

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

Content Activity Check for p1 and q1

4. |p2-p0| < Beta(QP) or |q2-q0| < Beta(QP)

Table-derived Operations: In order to simultaneously access Alpha, Beta, and Clip tables, and because most values of these tables are zero, we used combinational logic to implement Alpha, Beta and Clip tables instead of using memory buffer. It can save most of the space of memory buffer and improve overall system performance.

5. Result

The simulators used in this study are derived from the SimpleScalar/ARM tool set [10], a suite of functional and timing simulation tools for ARM ISA. The timing simulator executes only user-level instructions, performing a detailed timing simulation of an aggressive 4-way dynamically scheduled microprocessor with two levels of instruction and data cache memory. Our baseline simulation configuration models the Intel's StrongARM SA-110 processor. The hardware parameter is described in Table 3 below.

Table 3: Simulator Parameter

Parameter	Value
Fetch Queue size	4
Fetch Speed	1
Decode Width	1
Issue Width	1
Commit Width	1
D-Cache	32-way, 32-byte lines, LRU, 1-cycle hit, total 16KB
I-Cache	32-way, 32-byte lines, LRU, 1-cycle hit, total 8KB
Memory Latency	12
Memory Width	4 bytes

Table 4: The Performance Comparison

Item	Software based	AEFO Embedded Platform	Reduce by
Inst.	128640967	75123050	42%
Load	30443106	20180448	34%
Store	16098837	10295823	36%
Branch	14324486	7901023	49%
Cycles	220929397	132532824	40%

The simulation results are shown in Table 4. The performance of embedding AEFO as a co-processor is 1.66 times faster than the software implementation. Moreover, the number of total memory references for load and store is reduced by 34% and 36% respectively.

We implemented the proposed architecture by Verilog HDL and synthesized the design using TSMC 0.18um Artisan CMOS cell library using Synopsys Design Compiler with critical path constraint set to 5 ns (200MHz). The synthesized gate count is shown in Table 5.

Table 5: The Area of Adaptive Edge Filter

Item	Function	Gate count
1.	Alpha Table derived	137
2.	Beta Table derived	87
3.	CLIP Table	66
4.	Luma4	1372
5.	Chroma4	247
6.	Luma and Chroma	811
7.	Conditional Circuit	1104
8.	Computation of Boundary Strength	1752
Total	Edge Filter	5576

6. Conclusion

In this paper, we proposed an efficient VLSI architecture to accelerate the deblocking filter of H.264/AVC video coding and use Verilog HDL to implement it. The major idea is to reduce the number of conditional operations through embedded the computation of boundary strength, the table-derived operations, content activity check, and filtering operations in edge filter unit. Simulation results show that the processing capability of the proposed architecture AEFO is very appropriate for real-time deblocking of high-definition television (HDTV, 720x1280 pixels/frame, 60 frames/s video signals) video operating at 150MHz. 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.

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 thanks Elan Microelectronics Corp. (Fabless Semiconductor 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.

[10] Douglas C. Burger and Todd M. Austin, The SimpleScalar Tool Set, Version 2.0. University of Wisconsin, Madison Tech. Report. 1997.