Proceedings of the Fourth IASTED International Conference
Communications, Internet, and Information Technology
October 31 - November 2, 2005, Cambridge, USA

PARALLEL PROCESSING FOR DEBLOCKING FILTER 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 propose an efficient parallel architecture for the adaptive deblocking filter in H.264/AVC video coding standard. We use six forwarding shift register arrays (of which each contains 4×4 8-bit shift registers) with two transposing operations and two sets of filter operation (each set contains four edge filter operations) to support simultaneous processing of the horizontal and vertical filtering. The proposed architecture is called "Parallel Filtering Architecture (PFA)." As a result, the performance of PFA is 390% faster than the advanced architecture of the previous proposal. Moreover, the number of total memory references is reduced by 63% and 25% respectively compared to the basic and advanced architectures of the previous proposal.

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

1. Introduction

H.264/AVC [1] is the latest coding standard jointly developed by the Video Coding Experts Group (VCEG) of ITUT-T and the Moving Picture Experts Group (MPEG) of ISO/IEC. Comparing the H.264/AVC video coding tools like multiple reference frame, quarter per motion compensation, deblocking filter or integer transform to the tools of previous video coding standard, H.264/AVC brought in the most algorithm discontinuities in the evolution of video coding standards. It is significantly outperforms the previous ones in bit-rate reduction. The functional blocks of H.264/AVC, as well as their features, are shown in Figure 1. Previous studies [2] 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+ [3] and MPEG-4 advanced simple profile.

The block-based structure of the video coding architecture produces artifacts known as blocking artifacts. 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 deblocking filter has been successfully applied in H.263+ and MPEG-4 part 2 implementations as a post-processing filter. However this technique is non optimal as the reference frame used for the inter prediction has not been filtered. The blocking artifacts that appear in this reference frame result in less accurate prediction comparison between the current frame and reference frame. Consequently the coding efficient is reduced. In H.264/AVC, the deblocking filter is moved inside the motion-compensated loop to filter block edges resulting from the prediction and residual difference coding stages of the decoding process. As a result from previous study [2], the adaptive deblocking filter reduces the bit rate typically by 5-10% while producing the same objective quality as the non-filtered video. Since it is within the motion compensation prediction loop, the deblocking filter is often referred to as an "in-loop filter". A detailed description of the adaptive deblocking filter can be found in [4].

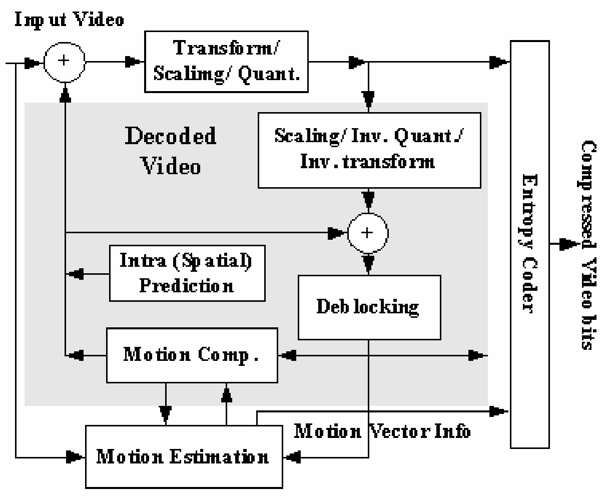


Figure 1: Block Diagram of H.264/AVC

As shown in Figure 1, H.264/AVC uses an in-loop deblocking filter to reduce the blocking artifacts introduced in the motion compensation loop. The deblocking filter is highly adaptive that adjusts its strength depending upon compression mode of a macroblock (Intra or Inter), the quantization parameter, motion vector, frame or field coding decision and the pixel values. This coding scheme is highly adaptive and is typically not suitable for DSP or SIMD computational architectures. Due to intensive computations, in [5] and [6] dedicated hardware was developed for acceleration. Our proposed architecture outperforms the previous proposals through parallel processing and significant reduction in memory reference.

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.

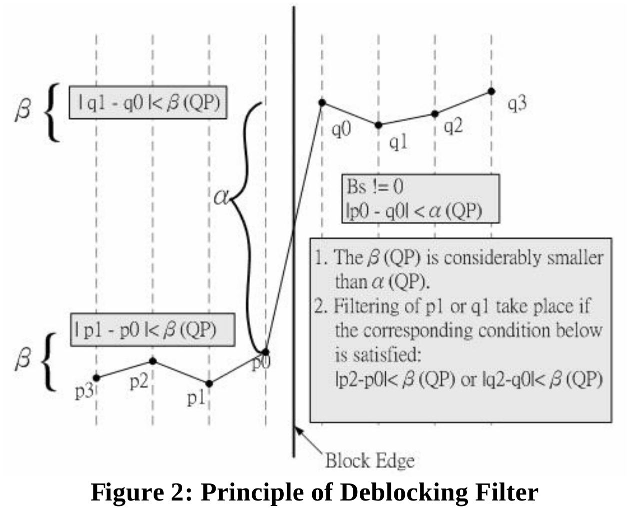
2. Algorithm of Deblocking Filter

On the sample processing level, content of a set of sample and quantization parameter threshold can turn on/off the filtering for each individual sample. For example as shown in Figure 2, 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). Thus filtering of p0 and q0 only takes place if the following content activity check is satisfied:

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

Correspondingly, filtering of p1 or q1 takes place if the condition below is satisfied:

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



β
$|q1 - q0| < \beta(QP)$
α
q0
q1
q2
q3
Bs != 0
$|p0 - q0| < \alpha(QP)$
1. The β(QP) is considerably smaller than α(QP).
2. Filtering of p1 or q1 take place if the corresponding condition below is satisfied:
$|p2-p0| < \beta(QP)$ or $|q2-q0| < \beta(QP)$
β
$|p1 - p0| < \beta(QP)$
p3
p2
p1
p0
Block Edge

Figure 2: Principle of Deblocking Filter

3. Architecture

3.1 Edge Filter Operation

One of the most time consuming operation of H.264/AVC deblocking filter is edge filtering operation. It employs highly adaptive filtering scheme, which requires several conditional processing on each block edges and sample levels. As described in the previous section, the threshold value of Alpha and Beta, the table-derived operations, and edge filtering operations are known to be very time consuming. Therefore, we implemented an efficient VLSI architecture that includes these content activity checks, the table-derived operations and filter operations into edge filter unit to accelerate the horizontal and vertical filtering on the boundary of two adjacent basic 4x4 blocks as shown in Figure 3. For parallel processing of each 4x4 block at one clock cycle, the horizontal and vertical filtering unit each contain four edge filtering operations to improve the overall system performance. A detailed description of the edge filtering operation can be found in our previous proposed VLSI architecture [7].

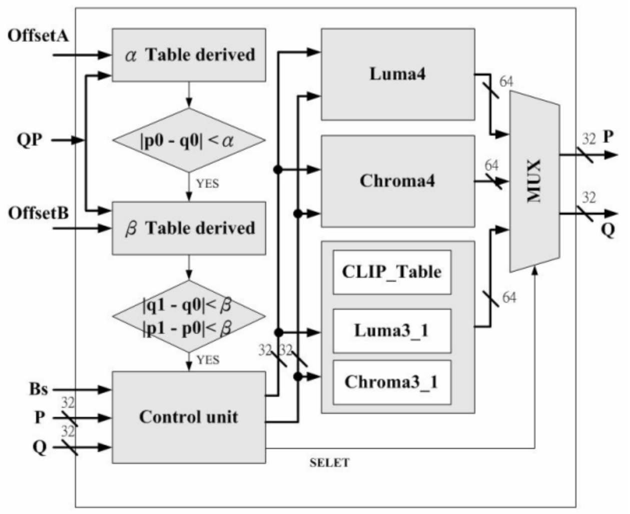


Figure 3: Edge Filtering Unit

3.2 Parallel Processing Architecture

Another time consuming part of H.264/AVC 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 requirement of on-chip SRAM bandwidth and increase the throughput of the filter processing, we propose an efficient parallel processing scheme as shown in Figure 4, which can perform simultaneous processing of horizontal and vertical filtering and reduce the number of memory access of each 4x4 block to one.

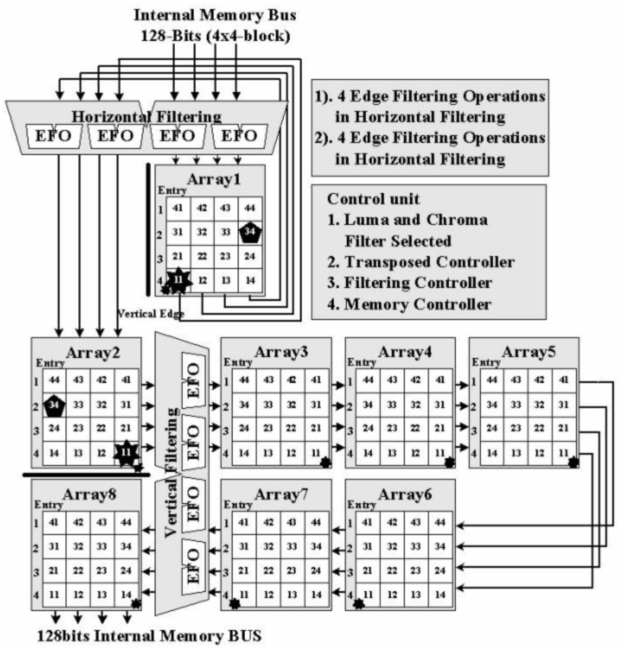


Figure 4: Parallel Filtering Architecture

3.3 Parallel Processing Order

Our architecture utilizes an alternative processing order, which allows the simultaneous processing of horizontal and vertical filtering as shown in Figure 5. The processing order begins from V6 to V7 ("V" denotes a vertical edge and "6" the sixth cycle). And then at the eighth cycle, the vertical edge V8 and horizontal edge H8 ("H" denotes a horizontal edge and "8" the eighth cycle) are simultaneously processed. Then V9, H9 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 PFA arrays.

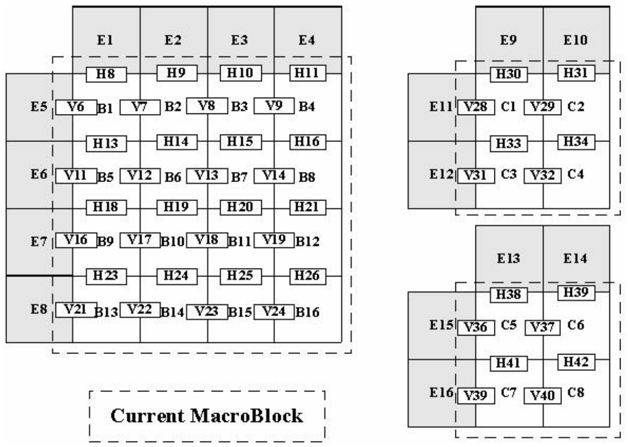


Figure 5: Parallel Processing Order

3.3 Components of Proposal Architecture

There are three major functions in our proposed architecture PFA. The first component is the Shift Operation Array (SOA). There are six forwarding shift register arrays in our proposed architecture (for example, Array1, 3, 4, 5, 6, and 7). Each array contains 16 processed samples. The operation of SOA can shift a basic 4x4-block to next Array at a time. The shift direction is shown in Figure 4. The second function of our proposed architecture is the transposing operation as shown in Figure 4. The Array2 and Array8 latched the filtered samples that are transposed from Array1 and Array7 respectively. And the final important functions are the horizontal and vertical filtering units. Each contains 4 edge filtering units which are described in section 3.1.

3.4 Data Flow

In this subsection, the basic processing order for a macroblock is shown in Figure 5. The data flow of the proposed architecture PFA for each macroblock is presented in Table 1. Assuming the memory data bus is 128-bit width. Therefore the proposed architecture PFA is able to filter one block at a time. For a basic 4x4-block, it takes 13 cycles to process the first block B1 and the number of total processing time for each macroblock is 32 cycles. In the first 5 cycles, the blocks of E1, E2, E3, E4, and E5 are loaded from internal memory to PFA' s Array5, Array4, Arrar3, Array2, and Array1 respectively and no filtering operations are performed. In the next two cycles (the sixth and seventh cycles), the horizontal filtering of vertical edge V6 and V7 are performed respectively. In the eighth cycle, the proposed architecture PFA can simultaneously process horizontal filtering of vertical edge V8 (the boundary of block B1 and B2) and vertical filtering of horizontal edge H8 (the boundary of block E1 and B1) and write the block E1 to internal memory in next cycle (the ninth cycle). In the thirteenth cycle, the vertical filtering of horizontal edges H13 (the boundary of lock B1 and B5) are performed. At this time, the block B1 has been filtered 4 times with the boundaries of block E5, B2, E1, and B5. Finally, PFA writes the block B1 to memory at the fourteenth cycle. As a result, the number of cycles to process a macroblock is 32. Therefore, the number of total block cycles required for a QCIF picture is 32x99=3168.

Table 1: Data Flow in the PFA

State	**Cycles**							
	5	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Array1	E5	B1	B2	B3	B4	E5	B5	B6
Array2	E4	E5	B1	B2	B3	B4	E5	B5
Array3	E3	E4	E5	B1	B2	B3	B4	E5
Array4	E2	E3	E4	E5	B1	B2	B3	B4
Array5	E1	E2	E3	E4	E5	B1	B2	B3
Array6		E1	E2	E3	E4	E5	B1	B2
Array7			E1	E2	E3	E4	E5	B1
Array8				E1	E2	E3	E4	E5
MEM					E1	E2	E3	E4

4. Result

As described in the previous section, the key feature of our proposed architecture is the utilization of the Pipeline Filtering Architecture (PFA) 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 by 63% 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 [5]. 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 References per Macroblock

Author	**Architecture**	**MEM Ref**
Huang[5]	Basic+Single-port SRAM	512
Huang[5]	Advanced+Dual-port SRAM	256
Huang[5]	Basic+Two-port SRAM	512
Huang[5]	Dual Arrays+Two-port SRAM	256
PFA	Dual-port SRAM or Two Single port SRAM	192

As described in previous section, the number of cycles for filtering each luma macroblock takes 32 cycles. Therefore, the number of total cycles for filtering one luma and two chroma macroblocks takes 32 and (6+8)x2 = 28 cycles respectively. As a result, the total filtering takes 60 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 [5]. Table 3 illustrates the cycle count required for each different implementation.

Table 3: Memory Cycles per Macroblock

Author	Architecture	MEM Ref
Huang[5]	Basic+Single-port SRAM	558
Huang[5]	Advanced+Dual-port SRAM	494
Huang[5]	Basic+Two-port SRAM	462
Huang[5]	Dual Arrays+Two-port SRAM	294
PFA	Dual-port SRAM or Two Single port SRAM	60

5. Conclusion

In this paper, we propose a parallel architecture with novel processing scheme to accelerate the operations of deblocking filter for H.264/AVC video coding and implement it in Verilog HDL. Key features include a novel processing scheme to reduce the bandwidth of memory system and two sets of parallel edge filtering unit to speed up the overall system performance. As a result, the PFA can be used in a high performance system which only requires a simple bus interface for the integration into video SoC platforms that support a wide range of applications such as video telephony, video conferencing, video streaming, digital video authoring, delivery of high-definition of DVD content, highest quality video for digital cinema, and many others.

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. (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] 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.
[3] ITU-T Recommend H.263, *Video Coding for Low Bit Rate Communication*, 1998.
[4] 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.
[5] 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.
[6] 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.
[7] Chung-Ming Chen, Chung-Ho Chen, "An Efficient VLSI Architecture for Edge Filtering in H.264/AVC," *IASTED CSS*, 2005.