Window Architecture for Deblocking Filter in H.264/AVC

Chung-Ming Chen [1], Jian-Ping Zeng [2], Chung-Ho Chen [1], Chao-Tang Yu [2] and Yu-Pin Chang [2]
[1]Department of Electrical Engineering, National Cheng Kung University, Taiwan, R.O.C.
[2] Department of Electronic Engineering, Southern Taiwan University of Technology, Taiwan, R.O.C.
cmchen@ee.ncku.edu.tw, m9430107@webmail.stut.edu.tw,
chchen@mail.ncku.edu.tw, ctyu@mail.stut.edu.tw and ypchang@mail.stut.edu.tw

Abstract—*In this paper, we study and analyze the computational complexity of deblocking filter in H.264/AVC baseline decoder based on SimpleScalar/ARM simulator. The simulation result shows that the memory reference, content activity check operations and filter operations are known to be very time consuming in the decoder of this new video coding standard. In order to improve overall system performance, we propose a window processing approach with efficient VLSI architecture which simultaneously processes the horizontal filtering of vertical edge and vertical filtering of horizontal edge. As a result, the processing capability of the proposed architecture is very appropriate for real-time deblocking of high-definition television (HDTV, 1920x1080 pixels/frame, 60 frames/s video signals) video operation at 60MHz. Moreover, the memory and system performance of our proposal significantly outperforms the previous designs as shown in result section.*

Keyword—*Deblocking Filter, H.264/AVC, Video Coding*

I. 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 emphasizes the demand for higher video compression rates. To achieve 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, has been developed. It significantly outperforms the previous ones (H.261 [2], MPEG-1 Video [3], MPEG-2 Video [4], H.263 [5] and MPEG-4 Visual or part 2 [6]) in bit-rate reduction. The functional blocks of H.264/AVC, as well as their features, are shown in Fig. 1. Comparing the H.264/AVC video coding tools like adaptive deblocking filter [7], integer transform [8] instead of the DCT [9], multiple reference frame [10], new frame types (SP-frames and SI-frames) [11], further predictions using B-slices [12], quarter per motion compensation [13] or CABAC [14] to the tools of previous video coding standard, H.264/AVC brought in the most algorithm in the evolution of video coding as well as error robustness and network friendliness. At the same time, preliminary studies [15] using software based on 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.

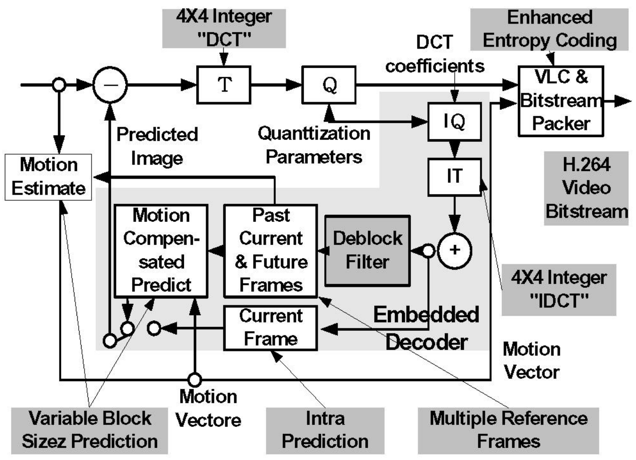


Fig. 1. Block diagram of H.264/AVC

As our experiment result indicates, the operation of the deblocking filter is the most time consuming part of H.264/AVC video decoder. The block-based structure of the H.264/AVC 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 overlapped block motion compensation (OBMC) [16] is adopted into H.263 standard. Unlike the OBMC in H.263, H.264/AVC adopts an adaptive deblocking filter [7] that has shown to be a more powerful tool in reducing artifacts and improving the video quality. As a result, the filter reduces the bit rate typically by 5-10% while producing the same objective quality as the non-filtered video [17]. 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". A detailed description of the adaptive deblocking filter can be found in [7].

The filtering operations of H.264/AVC standard require more instructions to process deblocking. Due to intensive computations, in [18], [19], [20], [21], [22], [23], [24] and [25] dedicated hardware was developed for acceleration. However, the deblocking filter described in the H.264/AVC standard is highly adaptive. Several parameters and thresholds, as well as the content 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 architecture. In order to reduce the conditional branch operations, we include the content activity check operations, table-derived operations, and filtering operations into edge filtering unit to accelerate the deblocking filtering of H.264/AVC video coding. In addition, we propose a window processing architecture to improve memory performance by 4 times when compared to the software implementation [1]. The proposed architecture is called "Win". It uses a novel processing order within a macroblock to simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge. Hence, our architecture is able to significantly improve the system performance.

The organization of this paper is as follows. In section II, the algorithm of the deblocking filter is explained. Section III illustrates the block diagram of our proposed architecture using window processing approach. Section IV shows the simulation results. Finally, conclusion is presented in Section V.

II. Algorithm of Deblocking Filter

In this section, we briefly describe the algorithm of deblocking filter in H.264/AVC from processing order to sample processing level. A detailed description of the adaptive deblocking filter can be found in [7].

A. Processing Order

As H.264/AVC standard recommendation [1], 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 Fig. 2.

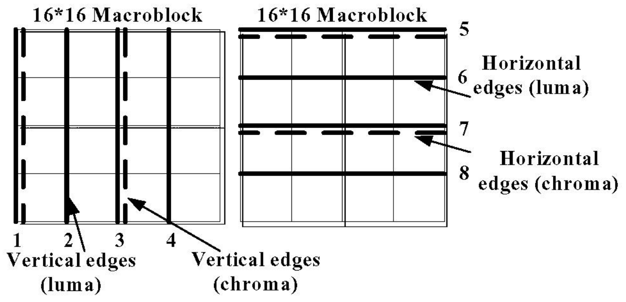


Fig. 2. Processing method of standard

B. Sample Processing Level

On the sample processing level, the quantization parameter, threshold value of Alpha and Beta, and content of picture itself can turn on or turn off the filtering for each individual set of sample. For example, Fig.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 Boundary Strength (Bs), dependent threshold Alpha(QP) and Beta(QP), and content of picture itself. Thus the filtering of p0 and q0 only takes place if the following content activity check operations are satisfied:

$$Bs != 0 \quad (1)$$
$$|p0 - q0| < Alpha(QP) \quad (2)$$
$$|p1 - p0| < Beta(QP) \text{ and } |q1 - q0| < Beta(QP) \quad (3)$$

Correspondingly, the filtering of p1 or q1 takes place if the 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.

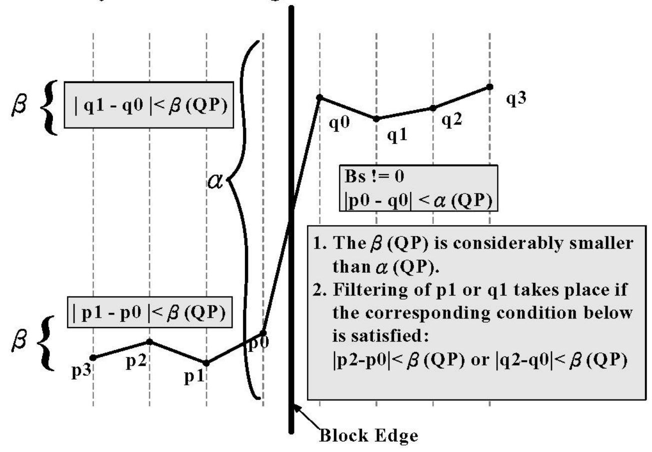

β
| q1 - q0 |< β (QP)
α
q0
q1
q2
q3
Bs != 0
|p0 - q0| < α (QP)
1. The β (QP) is considerably smaller than α (QP).
2. Filtering of p1 or q1 takes place if the corresponding condition below is satisfied:
|p2-p0|< β (QP) or |q2-q0|< β (QP)
β
| p1 - p0 |< β (QP)
p3
p2
p1
p0
Block Edge

Fig. 3. Principle of deblocking filter

III. Propose Architecture

The key features of our proposed architecture can be divided into two major components, including the edge filtering unit and window processing engine that employs a novel processing order to simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge.

A. Edge Filtering Unit

The complexity of the H.264/AVC Deblocking Filter is mainly based on two reasons. The first reason is the high adaptive filtering, 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 operation are known to be very time consuming.

Therefore, we propose an efficient VLSI architecture that includes content activity check operations, the table-derived operations, and filtering operations into the edge filter unit to accelerate the horizontal and vertical filtering on the boundary of two adjacent basic 4x4 blocks as shown in Fig. 4. A detailed description of the edge filtering unit can be found in [21].

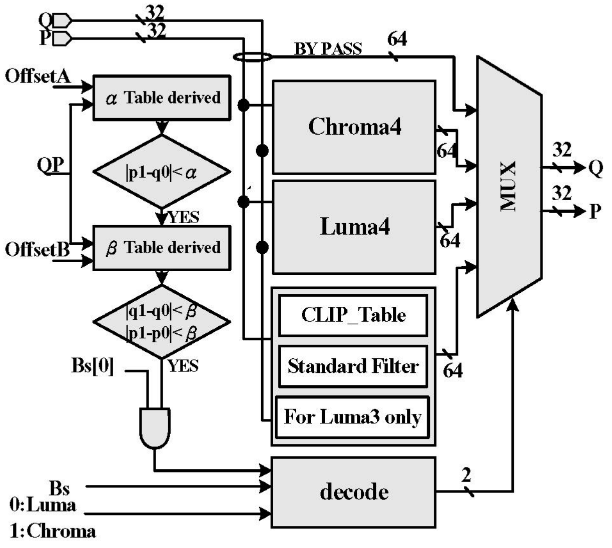


Fig. 4. Edge filtering unit

B. *Window Processing 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 transferred from and to internal 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 improve the overall system performance, we proposed a window processing architecture, which can simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge as shown in Fig. 5. The proposed architecture is called "Win".

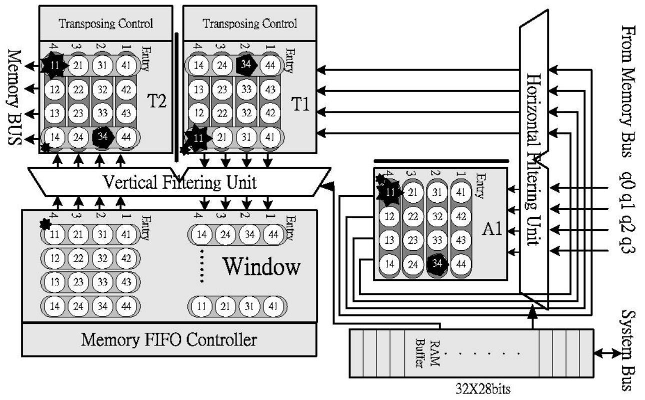


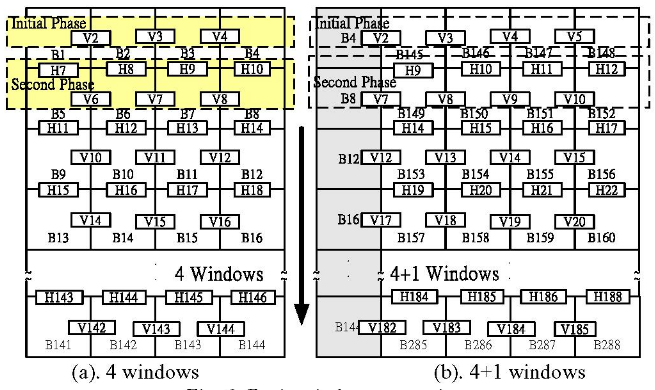
Fig. 5. Proposed architecture

There are three major sub-functions in our proposed architecture. The first component is the memory FIFO. There are two FIFO memories in our proposed architecture. The first one is A1 which contains 4 entries, each with 4 processed samples as shown in Fig. 5. The other is processing window which contains 44 block entries as shown in Fig. 5. The second sub-function in our proposed architecture is the transposing operation as shown in Fig. 5. The T1 and T2 latch the 4x4 block sample values that are transposed from A1 and windows respectively. The final important functions are the horizontal and vertical filter units which are described in the previous subsection.

C. *Window Processing*

In this section, we describe a window base approach to process deblocking filter in H.264/AVC. For the first cluster (4 columns, 4x36 blocks), as shown in Fig. 6 (a) and Table 1, each phase needs 4 block cycles to process a window data. During the initialization phase (the first phase), it takes 4 block cycles to load block B1, and perform horizontal filtering of vertical edge V2 and V3, and V4 sequentially. After initialization, in the second phase, the block B5 is loaded from the internal memory to F1 FIFO at the fifth block cycle, and then filtered with the block B6 at the sixth block cycle. At the seventh block cycle, the proposed architecture Win can simultaneously process the horizontal filtering of vertical edge V7 (the boundary of block B6 and B7) and the vertical filtering of horizontal edge H7 (the boundary of block B1 and B5), and then write the block B1 to the internal memory at the eighth block cycle. Then V8, H8 follows, so on and so forth. Therefore, it takes 7 block cycles (28 clock cycles) to process the first block B1. Then B2 follows, so on and so forth. Therefore, the number of total processing time for the first cluster is 7+143=150 block cycles (600 clock cycles).

For the second cluster (5 columns, 5x36 blocks), as shown in Fig. 6 (b) and Table 2, each phase needs 5 block cycles to process a window data and the window size is reconfigured as 5. Therefore, the number of total processing time for the second cluster is 8+179=187 block cycles (748 clock cycles). Table 3 shows variety window sizes of Win architecture.



(a). 4 windows (b). 4+1 windows

Fig. 6. Basic window processing.

Table 1. Data flow of 4 windows (w1, w2, w3, and w4)

State	Block Cycle									
	1	2	3	4	5	6	7	8	9	10
F1	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
T2		B1	B2	B3	B4	B5	B6	B7	B8	B9
W1			B1	B2	B3	B4	B5	B6	B7	B8
W2				B1	B2	B3	B4	B5	B6	B7
W3					B1	B2	B3	B4	B5	B6
W4						B1	B2	B3	B4	B5
T8							B1	B2	B3	B4
MEM								B1	B2	B3

Table 2. Data flow of 4+1 windows (w1, w2, w3, w4, and w5)

State	Block Cycle									
	1	2	3	4	5	6	7	8	9	10
F1	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
T2		B1	B2	B3	B4	B5	B6	B7	B8	B9
W1			B1	B2	B3	B4	B5	B6	B7	B8
W2				B1	B2	B3	B4	B5	B6	B7
W3					B1	B2	B3	B4	B5	B6
W4						B1	B2	B3	B4	B5
W5							B1	B2	B3	B4
T8								B1	B2	B5
MEM									B1	B2

Table 3. The performance comparison of variety window size

Win Size	Num of 4x4 block	Num of block cycles
2+1 Win	3	2449
4+1 Win	5	2020
8+1 Win	9	1825
16+1 Win	17	1707
32+1 Win	33	1668
44 Win	44	1630

IV. RESULT

The simulation results are shown in Table 4. The architecture of Win as a co-processor can accelerate H.264/AVC decoder system. Moreover, the number of total memory references for load and store is reduced by 34% and 36% respectively.

Table 4. The performance comparison

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

A. *Memory Performance*

In our proposed window architecture using novel processing order, the memory performance is improved by 4 times, when compared to software implementation. Table 5 shows the comparison of various architectures. The memory access times of our window architecture using novel processing can reduce by 592 to 16, when compared to the previous designs in [18], [22], and [25].

Table 5. Memory reference per macroblock

Author	Architecture	MEM
JM9.2 [1]	Software Implementation	768
Huang [18]	Basic+Single-port SRAM	768
Huang [18]	Advance+Dual-port SRAM	384
Huang [18]	Basic+Two-port SRAM	768
Huang [18]	Dual Arrays+Two-port SRAM	384
Chen [22]	Dual-port SRAM or Two Single port SRAM	192
Li [25]	5120 bits Dual-Port SRAM	192
Win	Dual-port SRAM and Using 44 window sizes	176

B. *System Performance*

As described in previous section, using 44 and 22 window size for luma and chroma respectively, the total filtering for a QCIF frame takes 1630x4 = 6520 and (420x4) x 2 = 3360 cycles respectively. As a result, the total filtering takes 9880 cycles for a QCIF frame. Our filtering scheme takes less number of cycles when compared to 240x99=23760, 286x99=28314, 240x99=23760, and 192x99 = 19008 cycles of the architecture described in [18], [20], [22], and [25]. Table 6 shows the performance comparison of various architectures. The cycle counts of memory reference between the external and internal memory are not calculated for a fair comparison.

Table 6. Processing cycles for QCIF

Author	Architecture	Cycles
Huang [18]	Basic+Single-port SRAM	504x99
Huang [18]	Advance+Dual-port SRAM	440x99
Huang [18]	Basic+Two-port SRAM	408x99
Huang [18]	Dual Arrays+Two-port SRAM	240x99
Sheng [18]	2-D Deblocking Filter	286x99
Chen [22]	Dual-port SRAM or Two Single port SRAM	240x99
Li [25]	5120 bits Dual-Port SRAM	192x99
Win	Dual-port SRAM and Using 44 window sizes	9880

C. *Implemantation*

We implemented the window 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 10 ns (100MHz). The hardware comparison of variety architecture is shown in Table 7.

Table 7. The hardware comparison of variety architecture

Author	Architecture	Gate Count
Huang[18]	Basic+Single-port SRAM	18.91K
Huang[18]	Advance+Dual-port SRAM	20.66K
Li[25]	5120 bits Dual-Port	9.57K
Chen [21]	Edge Filter Unit	5.66K
Chen [22]	Dual-port SRAM	22K
Win	44 window sizes	14.75K

V. Conclusion

In this paper, we study and analyze the memory reference of H.264/AVC baseline decoder based on SimpleScalar-ARM architecture. The result shows that the memory reference is known to be very time consuming in this new video coding standard. In order to reduce the memory reference and thus improve overall system performance, we propose window-based VLSI architecture to accelerate the operations of deblocking filter for H.264/AVC video coding. The major idea is to reduce the number of memory references through the simultaneous processing architecture Win using novel processing order. As a result, the processing capability of the proposed architecture is very appropriate for real-time deblocking of high-definition television (HDTV, 1920x1080 pixels/frame, 60 frames/s video signals) video operation at 60MHz. The proposed architecture Win only requires a simple bus interface for the integration into video SoC platforms that support a wide range of applications such as video telephone, video conferencing, video streaming, digital video authoring, and many others.

Acknowledgment

The work in this paper is in part supported by the National Science Council, Taiwan R.O.C., under NSC 93-2220-E-006-004.

References

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

[2] ITU-T Recommendation H.261, "Video codec for Audiovisual Services at p X 64 kbit/s," March 1993.

[3] ISO/IEC 11172: "Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s," Geneva, 1993.

[4] ISO/IEC 13818–2: Generic coding of moving pictures and associated audio information—Part 2: Video also ITU-T Recommendation H.262, 1994.

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

[6] ISO/IEC 14496–2: Information technology—coding of audiovisual objects—part 2: visual, Geneva, 2000.

[7] 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, pp.614-619, 2003.

[8] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp.598–603, July 2003.

[9] N. Ahmed, T. Natarajan, and R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp.90–93, Jan. 1974.

[10] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction for video coding," *IEEE Transactions on Circuits and Systems for. Video Technology*, vol. 9, pp.70–84, Feb. 1999.

[11] M. Karczewicz and R. Kurçeren, "The SP and SI frames design for H.264/AVC," *IEEE Transactions on Circuits and Systems*, vol. 13, no. 7, pp.637–644, July 2003.

[12] T. Wiegand, H. Schwarz, A. Joch, and F. Kossentini, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp.688–703, July 2003.

[13] T. Wedi and H.G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp.577–587, July 2003.

[14] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp.620–636, July 2003.

[15] 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*, pp.7-28, 2004.

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

[17] 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, pp.715–727, 2003.

[18] 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, pp.693-696, 2003.

[19] 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, pp.292-296, 2004.

[20] Bin Sheng, Wen Gao and Di Wu, "An Implemented Architecture of Deblocking Filter for H.264/AVC," IEEE International Conference on Image Processing (ICIP'04), Vol.1, 24-27, pp.665-668, Oct 2004.

[21] Chung-Ming Chen and Chung-Ho Chen, "An Efficient VLSI Architecture of Edge Filtering in H.264/AVC, IASTED International Conf. on Circuits," Signals, and Systems, pp.118-122, Oct. 2005.

[22] Chung-Ming Chen and Chung-Ho Chen, "An Efficent Architecture for Deblocking Filter in H.264/AVC Video Coding," IASTED International Conf. on Computer Graphics and Imaging , August. 2005.

[23] Chung-Ming Chen and Chung-Ho Chen, "Parallel Processing for Deblocking Filter in H.264/AVC," IASTED International Conf. on Communications, Internet, and Information Technology, pp.188-191, Oct. 2005.

[24] Chung-Ming Chen and Chung-Ho Chen, "A Memory Efficient VLSI Architecture for Deblocking Filter in H.264 Using Vertical Processing Order," IEEE International Conf. on Intelligent Sensors, Sensor Networks & Information Processing, pp.361-366, Dec. 2005.

[25] Lingfeng Li, Satoshi Goto, Takeshi Ikenaga, " A Highly Parallel Architecture for Deblocking Filter in H.264/AVC," IEICE Transactions on Information and Systems, pp.1623-1629, Vol.E88-D No.7, July 2005.

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