Windows Processing for Deblocking Filter in H.264/AVC

Chung-Ming Chen and Chung-Ho Chen
Department of Electrical Engineering
National Cheng Kung University
Tainan City, Taiwan
cmchen@ee.ncku.edu.tw
chchen@mail.ncku.edu.tw

Jian-Ping Zeng, Wan-Chug Hsu and Chao-Tang Yu
Department of Electronics Engineering
Southern Taiwan University of Technology, Taiwan
m9430107@webmail.stut.edu.tw
shimmeryhsu@gmail.com
ctyu@mail.stut.edu.tw

***Abstract* –In this paper, we propose a window-based architecture with parallel filtering engine to accelerate the adaptive deblocking filter in H.264/AVC video coding standard. In order to improve overall system performance, we use a configurable 44x32-bit FIFO memory with a novel processing order to simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge. As a result, the performance of our configurable window-based architecture with parallel engine significantly outperforms the previous designs in terms of memory references and processing cycles.**

I. 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, 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 decoder, as well as their features, are shown in Fig. 1. It is rich with diverse coding methods including the 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]. 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.

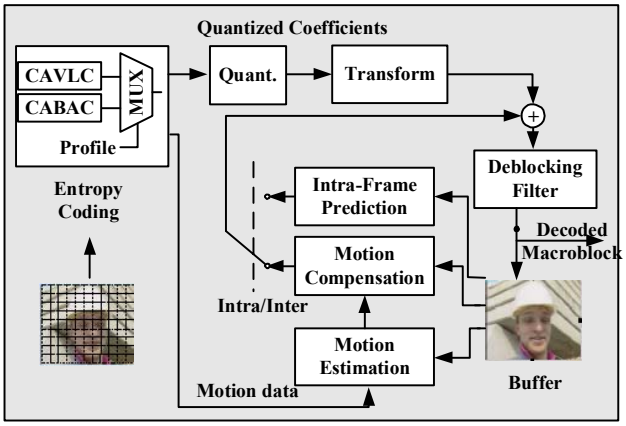


Fig. 1. Block diagram of H.264/AVC decoder

The block-based structure of the H.264/AVC architecture produces artifacts known as blocking artifacts. These blocking artifacts can occur from both the quantization of the transform coefficients and the block-based motion compensation. In order to reduce the blocking artifacts, the overlapped block motion compensation (OBMC) [16] is adopted into the H.263 standard. Unlike the OBMC in H.263, H.264/AVC uses an adaptive deblocking filter 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].

As our experiment result indicates, the operation of the deblocking filter is the most time consuming part of H.264/AVC video decoder. 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 small block size is employed for residual coding in the H.264/AVC video coding algorithm. With the 4x4 blocks and a typical filter length of two samples in each direction, each sample in a picture must be transferred from and to internal memory four times; either to be modified or to determine if the neighbouring samples will be modified. In order to reduce the number of memory references and improve the overall system performance, we propose a window-based processing architecture with parallel filtering engine, which can simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge. The proposed architecture is called "PWin."

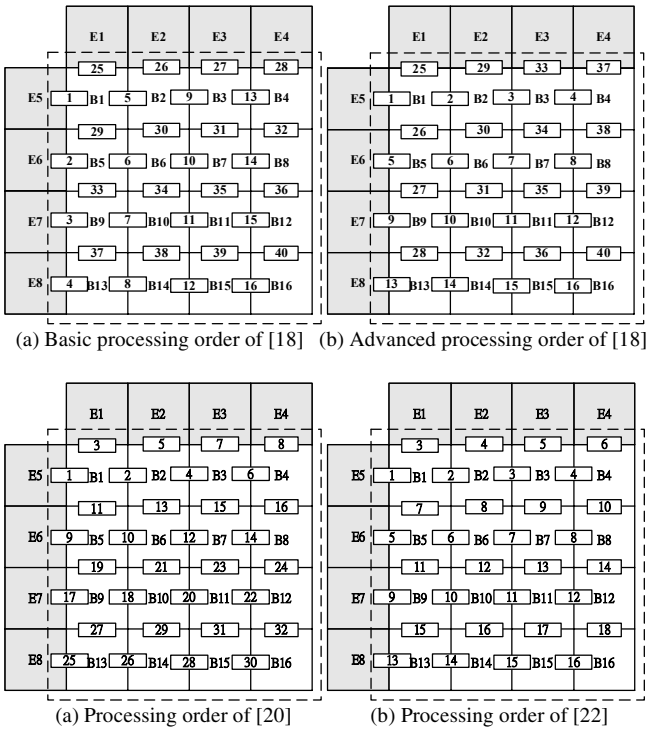
The organization of this paper is as follows. In Section II, the algorithm of the deblocking filter is explained. Section III illustrates the block diagram of our proposed architecture using the window processing approach. Section IV shows the simulation results. Finally, the conclusion is presented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

By the recommendation of H.264/AVC standard [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. According to this rule, there are four types of processing orders which are proposed by [18], [20] and [22] as shown in Fig 2 (a), (b), (c), and (d). It is obvious that adaptive deblocking filter shall be applied to all 4x4 block edges of a picture, except for the edges at the boundary of the picture and most of the 4x4 blocks need to be filtered four times with the adjacent blocks (left, right, top, and bottom). In order to improve the memory performance, we propose a window-based VLSI architecture with a novel processing order to reduce the number of memory references of each 4x4-block to one as shown in Section III.



(a) Basic processing order of [18] (b) Advanced processing order of [18]

(a) Processing order of [20] (b) Processing order of [22]

Fig. 2. Processing order of various architectures

B. Sample Processing Level

The complexity of the H.264/AVC deblocking filter is mainly based on the high adaptive filtering, which requires several conditional processing on each block edges and sample levels. 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 in a typical situation where the filter would be turned on. The filtering of p0 and q0 only takes place if the following content activity check operations are satisfied:

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

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

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

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

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

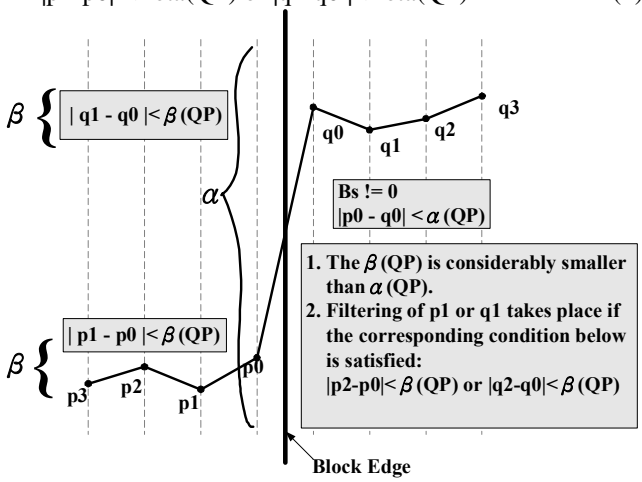


Fig. 3. Principle of deblocking filter

The basic idea is that if a relatively large absolute difference between samples near a block edge is less than the threshold value of Alpha and Beta, it is quite likely to be a blocking artifact and should therefore be filtered. However, if the value of that difference is so large that it can no longer be explained by the quantization and compensation used in the encoding, the edge is more likely to reflect the actual behaviour of the source picture and should not be smoothed over.

III. PROPOSED ARCHITECTURE

The key features of our proposed architecture include edge filtering units, a configurable window-based architecture with parallel processing engine (the horizontal and vertical filtering unit), and a novel processing order. The architecture and processing approach are described in the following section.

A. Edge Filtering Unit

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 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 the two adjacent basic 4x4 blocks as shown in Fig. 4. A detailed description of the edge filtering unit (EFU) can be found in [21].

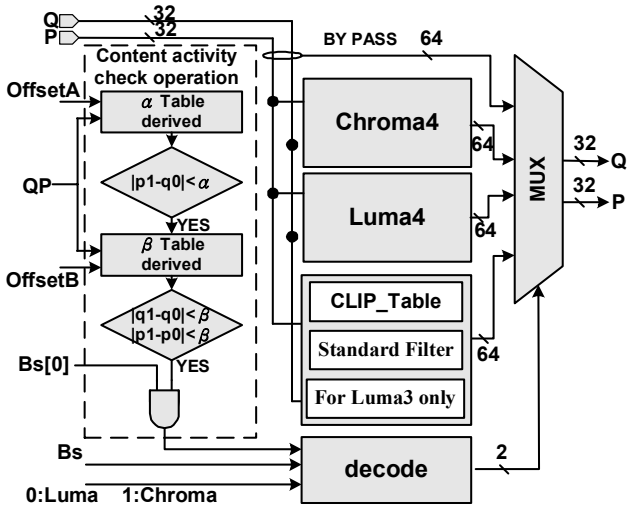


Fig. 4. Edge filter unit

B. Configurable Window-based Architecture

In order to reduce the number of memory references and improve the overall system performance, we propose a window-based processing architecture with parallel filtering engine, 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 "PWin."

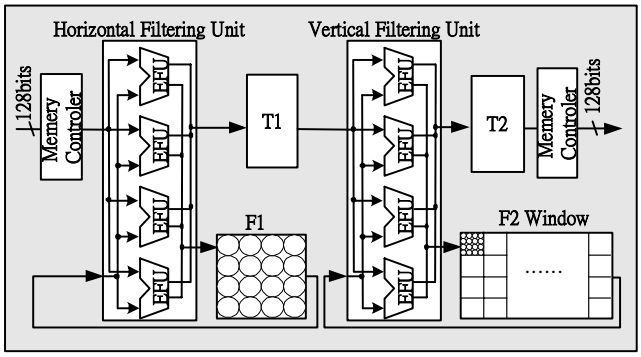


Fig. 5. Configurable window-based architecture

There are three major sub-functions in our window-based architecture. The first component is the memory buffers which are two FIFO memories in our proposed architecture. The first one is F1 which contains four entries, each with four processed samples as shown in Fig. 5. The other one is the configurable processing window, F2, which is able to be configured to any size of windows as shown in Table I that presents the performance of various window size. According to various image resolutions, the configurable window-based architecture can be configured to any size of windows by setting the window register as shown in Table II. The processing approach is described in next subsection. The second sub-function in our proposed architecture is the transposing operation. The transposing unit T1 and T2 latch the 4x4 block sample values which are transposed from FIFO F1 and the end of FIFO F2 respectively. In order to speedup the overall system performance, the final important sub-functions are the parallel horizontal and parallel vertical filtering units, each containing 4 EFU processing engine which can process a 4x4 basic block at a time.

TABLE I
THE NUMBER OF BLOCK CYCLES FOR A QCIF FRAME

Config	For one Luma Cluster	For one Chroma Cluster
WIN2	5+2x36=77	5+2x18=41
WIN3	6+3x36=114	6+3x18=60
WIN4	7+4x36=151	7+4x18=79
WIN5	8+5x36=188	8+5x18=98
WIN22	25+22x36=817	25+22x18=421
WIN44	47+44x36=1631	Non

TABLE II
THE NUMBER OF BLOCK CYCLES FOR VARIOUS RESOLUTIONS

Resolution		Win Size	Luma	Two Chroma	Total
QCIF	176x144	44&22	1631	842	2473
CIF	352x288	88&44	6427	3262	9689
VGA	640x480	160&80	19363	9766	29129
Video conf.	1280x720	320&160	58923	29126	88049
HDTV	1920x1080	480&240	130083	65286	195369

C. Parallel Filtering Engine

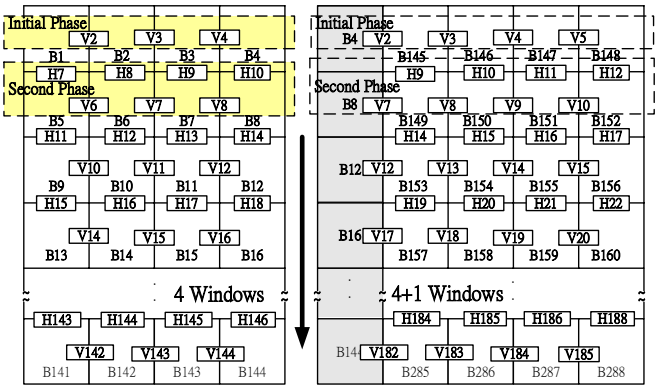
In order to speedup the processing time, the edge filtering unit is extended into a parallel engine. As shown in Fig. 5, both the horizontal and vertical filter units are extended into 4 EFU units so that the memory bus needs a 128-bit width to speedup the data transmission. Therefore, the performance can achieve 4 times when compared to the previous design [24].

D. Basic Window Processing Approach

In this section, we use Luma blocks of a QCIF frame to describe the basic window processing approach. For the first cluster (4 columns, 4x36 blocks), as shown in Fig. 6 (a) and Table III, each phase needs four cycles to process a window data. During the initialization phase (the first phase), it takes four cycles to load blocks B1, B2, B3 and B4, and perform horizontal filtering of vertical edges V2 and V3, and V4 sequentially. After initialization, in the second phase, block B5 is loaded from the internal memory to F1 FIFO at the fifth cycle, and then filtered with block B6 at the sixth cycle. At the seventh cycle, the proposed architecture PWin 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 block B1 to the internal memory at the eighth cycle. Then edges V8, H8, and the rest follow in the same way. As a result, it takes seven cycles to process the first block B1. After that, B2 follows and so on. Therefore, the number of total processing time for the first cluster is 7+144=151 cycles.

For the second cluster (5 columns, 5x36 blocks), as shown in Fig. 6 (b) and Table IV, each phase needs 5 cycles to process a window data and the window size is reconfigured

to five. As a result, the number of the total processing time for the second cluster is 8+180=188 cycles.



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

Fig. 6. Basic window processing

TABLE III
DATA FLOW OF 4 WINDOWS (W1, W2, W3, AND W4)

State	Cycle									
	1	2	3	4	5	6	7	8	9	10
F1	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
T1		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
T2							B1	B2	B3	B4
MEM								B1	B2	B3

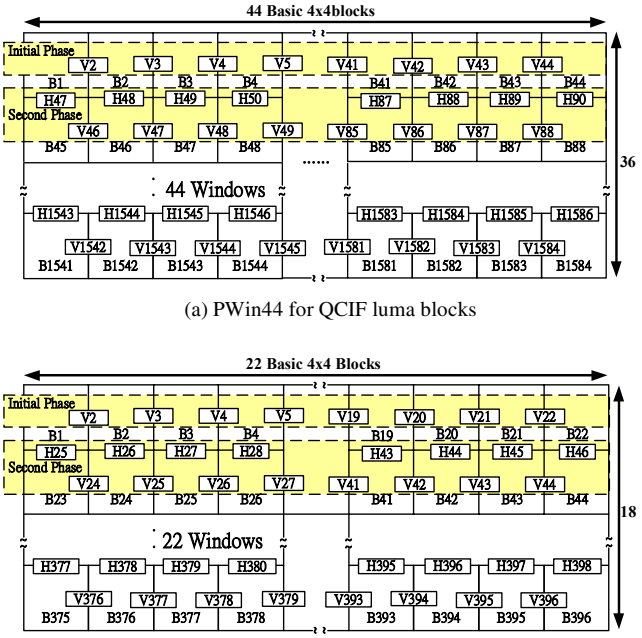
TABLE IV
DATA FLOW OF 4+1 WINDOWS (W1, W2, W3, W4, AND W5)

State	Cycle									
	1	2	3	4	5	6	7	8	9	10
F1	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
T1		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
T2								B1	B2	B5
MEM									B1	B2

E. Advanced Window Processing Approach

For luma blocks (44x36=1584 blocks) of a QCIF frame using the advanced window processing, the window size is configured to 44. As show in Fig. 7 (a), it requires 44 cycles to load and process the first phase, including the initial phase. At the 47th cycle, the block B1 just finished all adjacent filtering and then the block B1 is written to the internal memory in next cycle (the 48th cycle). After that, B2 follows and so on. As a result, the number of total processing time for all luma blocks using the PWin44 architecture is 47+44x36 = 1631.

There are two QCIF chroma blocks (22x18x2 blocks) to be processed. The processing approach is the same as the luma blocks but the window size is configured to 22. The processing order is presented in Fig. 7 (b). The initial processing cycles for each chroma block using the PWin22 is 25. At 26th cycle, the first block B1 has finished two times filtering and is written to the internal memory. After that, B2 follows and so on. As a result, the number of total processing time for two chromas using the PWin22 architecture is (25+22x18)x2=842 cycles.



(a) PWin44 for QCIF luma blocks

(b) PWin22 for QCIF chroma blocks

Fig. 7 Processing a QCIF frame using the PWin44 and PWin22 architecture

IV. RESULT

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

TABLE V
THE PERFORMANCE COMPARISON

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

A. Memory Performance

Using our proposed configurable window-based processing architecture, the memory performance is improved by 6 times, when compared to the software

implementation. Table VI shows the comparison of various architectures. Our window-based processing architecture can reduce the memory access times per macroblock from 592 to 66 when compared to the previous designs in [18], [22], and [25].

TABLE VI
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
PWin	Dual-port SRAM and Using 44 window sizes	126

B. System Performance

As shown in the previous section, using 44 and 22 window size for luma and chroma respectively, the total filtering for a QCIF frame takes 1631 and 421 x 2 = 842 cycles respectively. As a result, the number of total filtering cycles for a QCIF frame is 2473. Our filtering scheme takes less number of cycles when compared to 294x99=29106, 286x99=28314, 240x99=23760, and 192x99 = 19008 cycles of the architecture described in [18], [20], [22], and [25] respectively. Table VII 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 VII
PROCESSING CYCLE FOR LUMA BLOCKS OF A QCIF FRAME

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	294x99
Sheng [20]	2-D Deblocking Filter	286x99
Chen [22]	Dual-port SRAM or Two Single port SRAM	240x99
Li [25]	5120 bits Dual-Port SRAM	192x99
Window	Dual-port SRAM and Using 44 window sizes	9892
PWin	Dual-port SRAM(128bits data bus)	2473

C. Implementation

We implemented the window-based architecture by Verilog HDL and synthesized the design using TSMC 0.18um Artisan CMOS cell library with Synopsys Design Compiler by setting the critical path constraint to 5 ns (200MHz). The hardware comparison of the various architectures is shown in Table VIII.

TABLE VIII
THE HARDWARE COMPARISON OF VARIOUS 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
PWin	44 windows with 8 EFO	52.55K

D. Verification

In order to test our window-based architecture, we modified JM9.2 to fit our test platform and implemented our proposed architecture on an FPGA as shown in Fig. 8. Data file of the reconstructed pixels before filtering is saved in YUV non-filtered data RAM which is driven from software model and then sent to deblocking filter architecture under control. The filtered result of YUV data is compared with the filtered result of the software module as shown in Fig. 8.

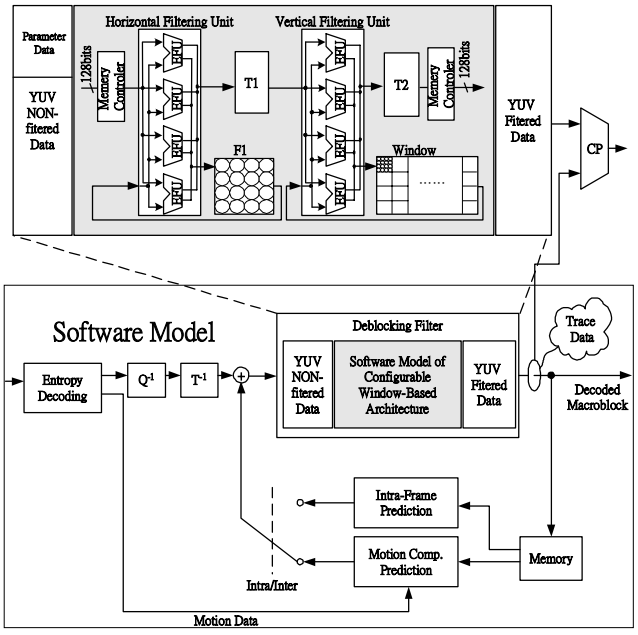


Fig. 8. The block diagram of verification

V. CONCLUSION

In this paper, we propose a window-based VLSI architecture with parallel processing engine to accelerate the operations of deblocking filter in H.264/AVC video coding. There are three major ideas. The first idea is to reduce the number of condition branch operations by implementing content activity check, the table-derived, and edge filtering operations into the edge filter unit. The second idea is to

reduce the number of memory references using a novel processing order to simultaneously process horizontal and vertical filtering. The last is to parallelize the Edge Filtering Unit to speed up the filtering performance. As a result, the PWin 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 telephone, video conferencing, video streaming, digital video authoring, and many others.

VI. ACKNOWLEDGMENT

The work in this paper is in part supported by the National Science Council, Taiwan R.O.C., under NSC 94-2220-E-006-004.

VII. 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, 2003, pp.614-619.

[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, July 2003, pp.598–603.

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

[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, Feb. 1999, pp.70–84.

[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, July 2003, pp.637–644.

[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, July 2003, pp.688–703.

[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, July 2003, pp.577–587.

[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, July 2003, pp.620–636.

[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*, 2004, pp.7-28.

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

[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, 2003, pp.715–727.

[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*, 2003, pp.693-696.

[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, 2004, pp.292-296.

[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, Oct 2004, pp.665-668.

[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,* Oct. 2005, pp.118-122.

[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*, Oct. 2005, pp.188-191.

[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*, Dec. 2005, pp.361-366.

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