A Memory Efficient Architecture for Deblocking Filter in H.264 Using Vertical Processing Order

Chung-Ming Chen [1], Chung-Ho Chen [2]
[1] *Department of Electrical Engineering, National Cheng Kung University*
Taiwan, R.O.C., cmchen@casmail.ee.ncku.edu.tw
[2] *Department of Electrical Engineering, National Cheng Kung University*
Taiwan, R.O.C , chchen@mail.ncku.edu.tw

Abstract

In this paper, we study and analyze the memory reference of deblocking filter in H.264/AVC baseline decoder based on SimpleScalar/ARM simulator. The simulation 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 a vertical processing order with efficient VLSI architecture which simultaneously processes the horizontal filtering of vertical edge and vertical filtering of horizontal edge. As a result, the memory performance of the proposed architecture is improved by 4.4 times when compared to software implementation. Moreover, the system performance of our proposal is 129% faster than the advanced architecture of previous proposal.

1. INTRODUCTION

The new video coding standard Recommendation H.264 of ITU-T [1] also known as ISO/IEC 14496-10 or MPEG-4 Part 10 Advanced Video Coding (AVC) [2], 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 Fig. 1. 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. At the same time, preliminary studies [4] 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.

As our experiment result indicates, the operation of the deblocking filter is the most time consuming parts 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) [5] is adopted into H.263 standard. Unlike the OBMC 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. 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]. 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 [6].

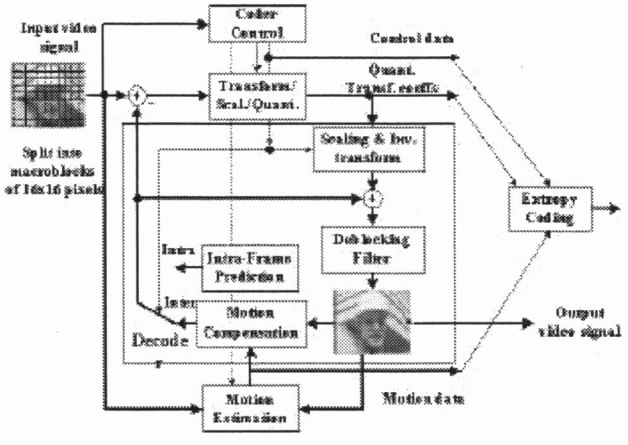

Fig. 1: Block Diagram of H.264/AVC

The filtering operations of H.264/AVC standard require more instructions to process deblocking. Due to intensive computations, in [8], [9], and [10] 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 operation, we include the content activity check, 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 an efficient VLSI architecture to improve memory performance by 4.4 times when compared to the software implementation [1]. The proposed

architecture is called "Simultaneous Processing Architecture (SPA)". It uses a novel processing order to simultaneously process the horizontal filtering of vertical edge and vertical filtering of horizontal edge and adopts vertical processing order instead of raster scan to improve overall system performance. Hence, our architecture is able to significantly reduce the power consumption in the embedded system.

The organization of this paper is as follows. In section 2, the algorithm of the deblocking filter is explained. Section 3 illustrates the block diagram of our proposed architecture using basic and vertical processing order. Section 4 shows the simulation results. Finally, conclusion is presented in Section 5.

2. ALGORITHM OF DEBLOCKING FILTER

As ITU-T Recommendation [1], 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 Fig. 2 ("V" denotes Vertical edge and "1" the first block cycle, "H" denotes Horizontal edge and "17" the seventeenth block cycle.)

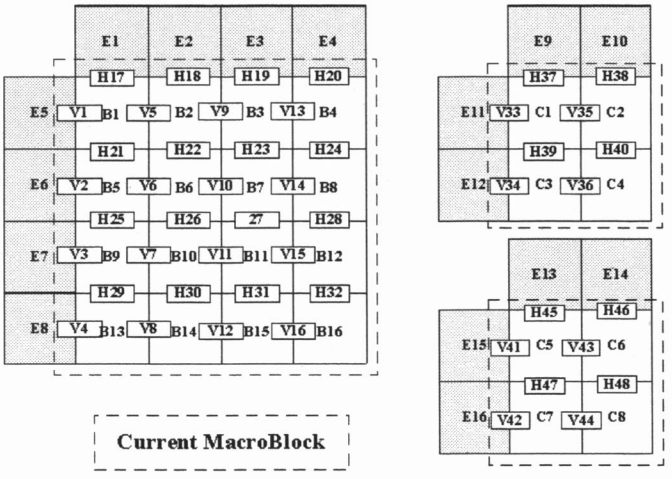

Fig. 2: Processing Order of Standard

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 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:

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

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

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

Correspondingly, 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.

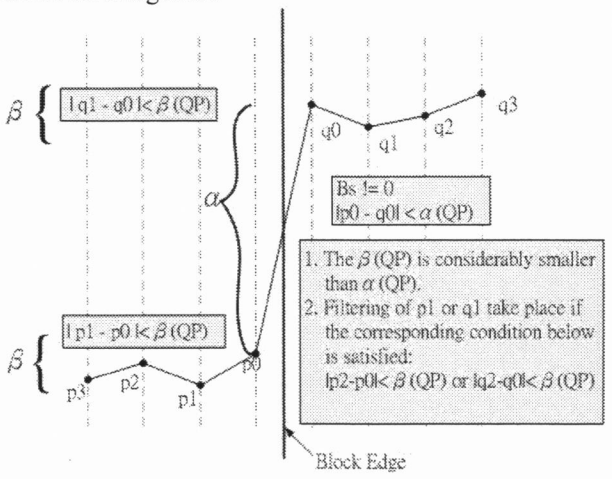

Fig. 3: Principle of Deblocking Filter

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 artefact 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 behaviour of the source picture and should not be smoothed over.

3. PROPOSED ARCHITECTURE

The key feature of our proposed architecture can be divided into two major components, including the edge filtering unit and a simultaneous 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 Filter Operation

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 proposed an efficient VLSI architecture that includes 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 Fig. 4. A detailed description of the edge filtering unit can be found in [11].

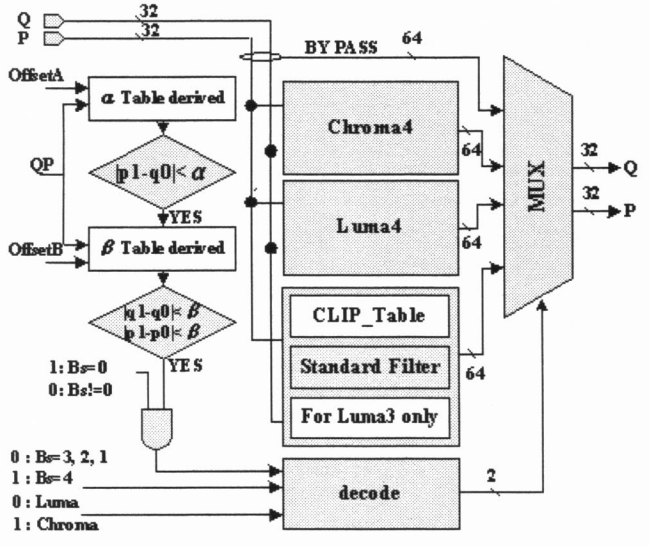

Fig. 4: Edge Filtering Unit

B. *Simultaneous 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 loaded/stored from/to memory 4 times; either to be modified or to determine if the neighbouring samples will be modified. In order to reduce the numbers of memory reference and improve the overall system performance, we proposed an efficient architecture, which can simultaneously processes the horizontal filtering of vertical edge and vertical filtering of horizontal edge as shown in Fig. 5. The proposed architecture is called "Simultaneous Processing Architecture (SPA)".

There are three major sub-functions in our proposed architecture. The first component is the Shift Operation Array. There are six forwarding shift register arrays in our proposed architecture (for example, Array1, 3, 4, 5, 6 and 7). Each array contains four entries which contains 4 processed samples. The shift direction is shown in Fig. 5. The second function in our proposed architecture is the transposing operation as shown in Fig. 5. The Array2 and Array8 latch the 4x4 block sample values that are transposed from Array 1 and Array7 respectively. The final important functions are the horizontal and vertical filter units which are described in the previous subsection.

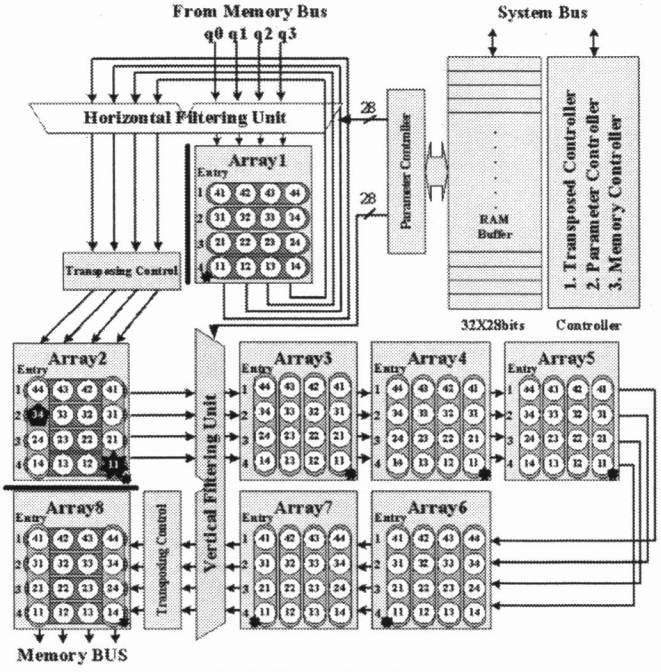

Fig. 5: Proposed Architecture

C. *A Novel Processing Order in a Macroblock*

In a 16x16 samples macroblock, our architecture utilizes a novel processing order, which allows the simultaneous processing of horizontal and vertical filtering as shown in Fig. 6. The processing order begins from V6 to V7 ("V" denotes Vertical edge and "6" represents the sixth block cycle). And then at the eighth block cycle, the vertical edge V8 and horizontal edge H8 ("H" denotes Horizontal edge and "8" represents the eighth block cycle) are simultaneously processed with the horizontal and vertical filtering unit. Then V9, H9 follows, so on and so forth.

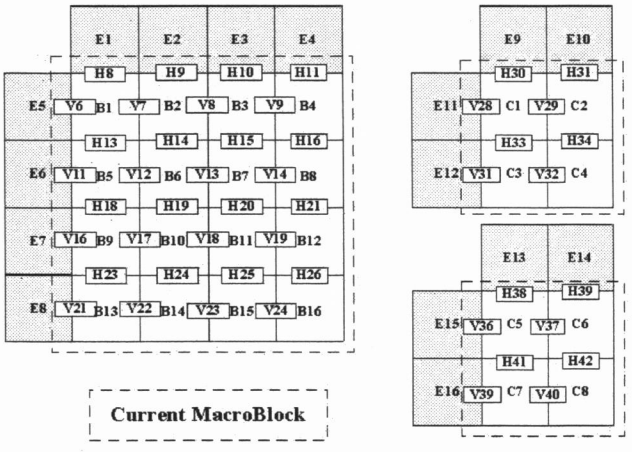

Fig. 6: Proposed Processing Order

D. *Basic Processing*

In this subsection, the basic processing using raster scan order for a picture is shown in Fig. 7. The data flow for the basic processing order to process each macroblock is presented in

Table 1. For a basic 4x4-block, it takes 13 block cycles (52 clock cycles) to process the first block B1 and the number of total processing time for each macroblock is 32 block cycles (128 clock cycles). In the first 5 block cycles (the first 20 clock cycles), the blocks of E1, E2, E3, E4, and E5 are loaded from the internal memory to SPA's Array5, Array4, Arrar3, Array2, and Array1 respectively and no filtering operations are performed. In the next 2 block cycles (the sixth and seventh block cycle), the horizontal filtering of vertical edge V6 and V7 are performed sequentially. At the eighth block cycle, the proposed architecture SPA can simultaneously process the horizontal filtering of vertical edge V8 (the boundary of block B2 and B3) and the vertical filtering of horizontal edge H8 (the boundary of block E1 and B1), and write the block E1 to the internal memory at the next block cycle (the ninth block cycle). At the thirteenth block cycle, the vertical filtering of horizontal edge H13 (the boundary of block B1 and B5) is performed. At this time, the block B1 has finished 4 times filtering with the adjacent blocks (left block E5, right block B2, top block E1, and bottom block B5). Finally, SPA writes the block B1 to the internal memory at the fourteenth block cycle. Therefore, the number of total block cycles required for a QCIF picture is 32x99=3168 (12672 clock cycles).

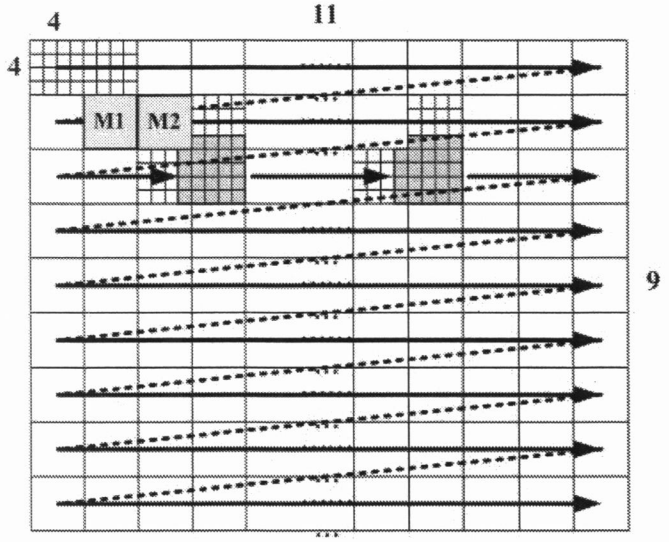

Fig. 7: Basic Processing Order

TABLE 1: DATA FLOW OF BASIC PROCESSING ORDER

State	Block Cycle							
	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

E. Vertical Processing

The vertical processing for a picture uses top-down order instead of raster scan order as shown in Fig. 8. The basic idea of the vertical processing is to buffer the initial blocks B13, B14, B15, and B16 in SPA's Array4, Array3, Array2, and Array1 respectively as shown in Fig. 9 and Table 2 When processing the next macroblock M2, the SPA with vertical processing order does not need to load these initial blocks. And thus we can save 4 block cycles when processes each macroblock, except for the macroblock at the bottom of a picture. As a result, the number of the total block cycles to process a QCIF frame is 2021 (equal to 8084 clock cycles). Table 2 shows the data flow of the vertical processing order.

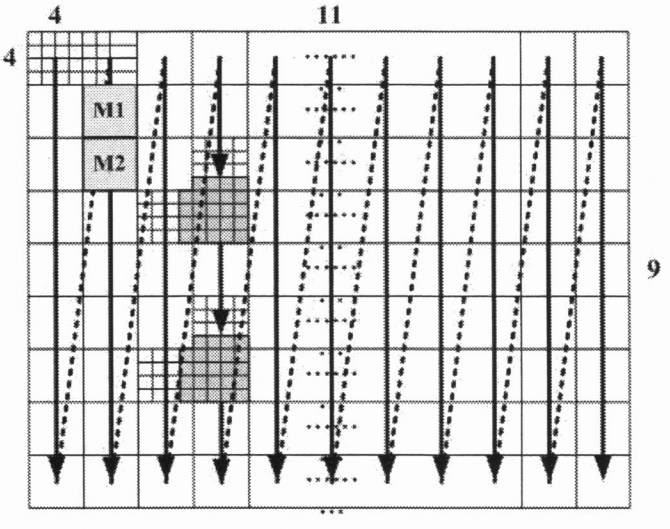

Fig. 8: Vertical Processing Order

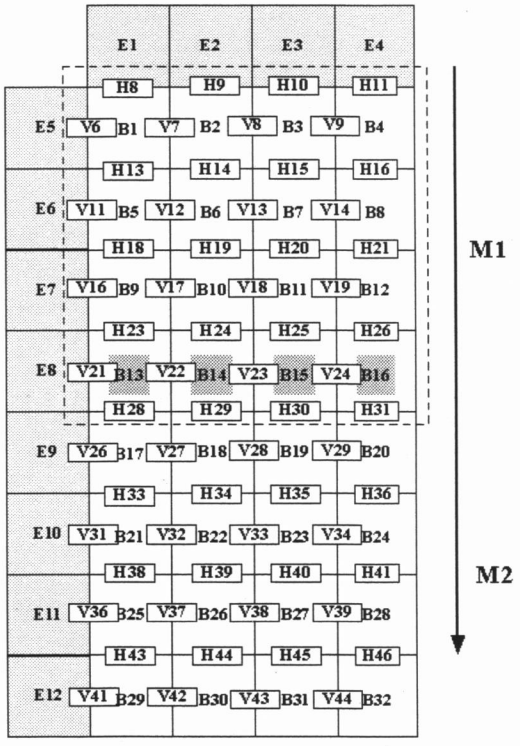

Fig. 9: Buffer Skill of Vertical Processing

TABLE 2: DATA FLOW OF VERTICAL PROCESSING ORDER

State	Block-Filtering Cycle							
	1	2	3	4	5	6	7	8
Array1	E9	B17	B18	B19	B20	E10	B21	B22
Array2	B16	E9	B17	B18	B19	B20	E10	B21
Array3	B15	B16	E9	B17	B18	B19	B20	E10
Array4	B14	B15	B16	E9	B17	B18	B19	B20
Array5	B13	B14	B15	B16	E9	B17	B18	B19
Array6		B13	B14	B15	B16	E9	B17	B18
Array7			B13	B14	B15	B16	E9	B17
Array8				B13	B14	B15	B16	E9
MEM					B13	B14	B15	B16

4. RESULT

The simulators used in this study are derived from the SimpleScalar/ARM tool set [12], a suite of functional and timing simulation tools for ARM ISA. 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

A. *Memory Performance*

As the ITU-T Recommendation [1] and previous proposal [8] and [9] suggest, an adaptive filtering shall be applied to all 4x4 block edges of a picture. Most of the 4x4 blocks need to be filtered 4 times with the adjacent blocks (left, right, top, and bottom), except for the macroblock at boundary of a picture. Therefore, the number of total memory reference for each macroblock, including read and write, is 4x4x2x16=512. For a picture in a QCIF format, the number of total memory accesses is 47392 (Assume that the memory bus width is 32 bits).

In our proposed SPA architecture using basic and vertical processing order, the memory performance is improved by 4 and 4.4 times respectively, when compared to software implementation. Table 4 shows the comparison of various architectures. Our SPA architecture using vertical processing can save 78.75% and 52.5% of the memory bandwidth respectively, when compared to the previous basic and advanced proposals in [8]. Hence, our architecture is able to significantly reduce power consumption in an embedded system.

TABLE 4: MEMORY REFERENCE PER MACROBLOCK

Author	Architecture	MEM Ref
JM9.2 [1]	Software Implementation	768
Huang [8]	Basic+Single-port SRAM	768
Huang [8]	Advance+Dual-port SRAM	384
Huang [8]	Basic+Two-port SRAM	768
Huang [8]	Dual Arrays+Two-port SRAM	384
Basic Processing	Dual-port SRAM or Two Single port SRAM	192
Vertical Processing	Dual-port SRAM or Two Single port SRAM	176

B. *System Performance*

As described in previous section, using vertical processing order, the total filtering for one luma and two chroma macroblocks takes 20x4=80 and (6x4=24)x2=48 cycles respectively. As a result, the total filtering takes 128 cycles for a luma and two chroma macroblocks. Our filtering scheme takes less number of cycles when compared to 294 and 286 cycles of the architecture described in [8] and [10]. Table 5 shows the performance comparison of various architectures. The cycle counts of loads and stores between the external and internal memory are not calculated for a fair comparison.

TABLE 5: PROCESSING CYCLES PER MACROBLOCK

Author	Architecture	Cycles/ MB
Huang [8]	Basic+Single-port SRAM	558
Huang [8]	Advance+Dual-port SRAM	494
Huang [8]	Basic+Two-port SRAM	462
Huang [8]	Dual Arrays+Two-port SRAM	294
Sheng [10]	2-D Deblocking Filter	286
Basic Processing	Dual-port SRAM or Two Single port SRAM	240
Vertical Processing	Dual-port SRAM or Two Single port SRAM	128

C. *Implementation*

We implemented the SPA 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 6 and Table 7.

TABLE 6: THE AREA OF EDGE FILTERING UNIT

Item	Function	Gate count
1.	Alpha Table derived	137
2.	Beta Table derived	87
3.	CLIP Table	66
4.	Luma4 Filtering Operation	1372
5.	Chroma4 Filtering Operation	247
6.	Standard Filtering for Luma and Chroma	811
7.	Content Activity Check	1104
Total	Edge Filter Unit	3824

TABLE 7: THE AREA OF SPA ARCHITECTURE

Item	Function	Gate count
1.	Horizontal Filtering Unit	3824
2.	Vertical Filtering Unit	3824
3.	SPA Processing Circuit	10778
Total	SPA Architecture	18426

5. CONCLUSION

In this paper, we propose an efficient 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 SPA using vertical processing order. As a result, the SPA 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.

ACKNOWLEDGEMENT

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

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*, pp. 7-28, 2004.

[5] 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.

[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, pp.614-619, 2003.

[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, 715–727, 2003.

[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, pp. 693-696, 2003.

[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, pp. 292-296, 2004.

[10] 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.

[11] Chung-Ming Chen, Chung-Ho Chen, "An Efficient VLSI Architecture of Edge Filtering in H.264/AVC", IASTED International Conf. on CSS, 2005.

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