Register Array Structure for Effective Edge Filtering Operation of Deblocking Filter

Jongwoo Bae^1, Neungsoo Park^{2*}, and Seong-Won Lee^3

^{1} System LSI Division, Samsung Electronics, Co. Ltd., Suwon, Korea
jwp.bae@samsung.com
^{2} Dept. of Computer Engineering, Konkuk University, Korea
neungsoo@konkuk.ac.kr
^{3} Dept. of Computer Engineering, Kwangwoon University, Korea
swlee@kw.ac.kr

Abstract. In this paper we propose a novel deblocking filter architecture using register array structure for standard video codec hardware. The proposed register array consists of multiple sub-macroblocks for a single macroblock and several sub-macroblock registers for the up and left neighboring macroblocks. The operation procedure of the register array is also presented. The proposed register array achieves fast operating speed and small circuit size at the same time.

1 Introduction

Most modern image processing systems use compressed image data that are generated by standard video codec. Widely used standard codecs are MPEG-1, MPEG-2, and MPEG-4 of International Telecommunication Union (ITU) and Motion Picture Experts Group (MPEG) recommendations. Recently new standard of MPEG-4 AVC (H.264) [1] has great attention because of its higher compression ratio. Research and standardization of H.264 are actively performed worldwide to achieve even higher compression ratio [2,3].

A video decoder system consists of a processor, multiple hardware modules to decode the compressed image data, memory, and peripherals such as DMA and PCI. The decoding hardware modules are Parser, Entropy Decoder, Inverse Transformer, Predictor, and Deblocking Filter. The compressed image data are restored to the original image sequence by sequential processing of the hardware modules.

During the compression process, image data are divided into "macroblocks" that are the basic unit of compression. Due to the individual processing of each macroblock, the restored image data has blocking effect which indicates the distortion of data discontinuity at the macroblock boundary. Since the blocking effect is appeared as a lattice in which the size of a square matches the size of a macroblock, the distortion is highly perceptual and seriously degrades subjective image quality. Deblocking is a process to reduce the blocking effect and deblocking filter [4] is the hardware module

* Corresponding author

that performs deblocking. The detail algorithm of deblocking is presented in the H.264 documentation [1].

In this paper, we present register array structure of deblocking filter to reduce calculation time of filtering operations. The proposed register array structure effectively reduces vertical filtering operation time. The efficient operation procedure for the proposed register array structure is also proposed.

Section 2 describes general operation of deblocking filter. The structure and operation of the proposed deblocking filter are presented in Section 3. In the Section 4 we analyze the performance of the proposed deblocking filter. Finally, we conclude the paper in Section 5.

2 Background Information

Deblocking filter operates on the uncompressed image data that is the largest data in the H.264 decoder system. Thus many researchers have studied efficient operation of the deblocking filter. Sima et al. proposed the pipelined deblocking filter [5]. In the pipelined deblocking filter, steps of the deblocking filtering are pipelined to improve processing performance. Li et al. propose the parallelized deblocking filter [6]. In addition to pipelining of the deblocking filter, they rearrange processing elements to improve the parallelism of the deblocking filter. Huang et al. proposed a small register array structure [7]. However, the Huang's register array does not have enough performance to process full size HDTV.

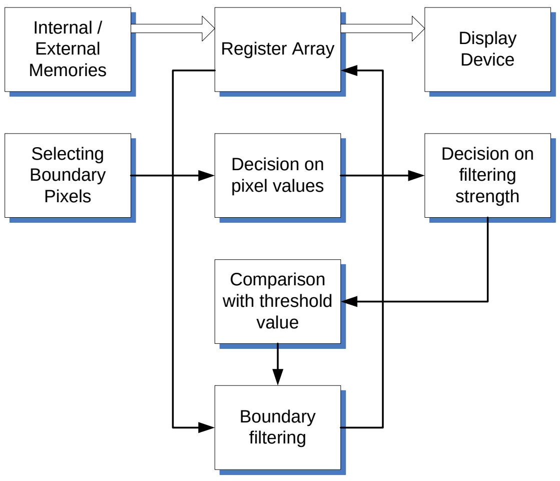

Internal / External Memories
Register Array
Display Device
Selecting Boundary Pixels
Decision on pixel values
Decision on filtering strength
Comparison with threshold value
Boundary filtering

Fig. 1. Block diagram of deblocking filter operation

Fig. 1 shows general operations of a deblocking filter. In the figure the deblocking filter first selects boundaries to perform filtering, then reads corresponding pixel data from memory to a register array. Filtering strength is decided to prevent excessive filtering on real edges. Threshold value is compared to check if the filtering operation is performed or not. Once filtering operation is decided to perform, pixels in the register array is processed with the deblocking filter and output to the display device.

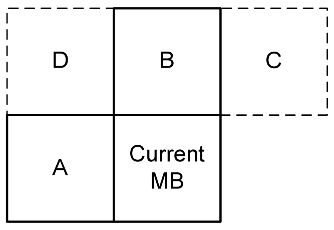


Fig. 2. A macroblock (MB) and its neighboring macroblocks for the deblocking filter.

Since the individual compression of macroblocks cause blocking effect, the deblocking filter also individually operates on macroblocks. Fig. 2 shows that a macroblock to be processed and its neighboring macroblocks (A and B) that is necessary to the deblocking filgering.

Filtering operation on a single macroblock respectively performs on a Luminance component and two Chrominance components. A macroblock consists of 16×16 pixels. The 16×16 of a macroblock is divided into 16 sub-macroblocks of 4×4 pixels. The deblocking filtering also removes blocking effect at the boundaries of the sub-macroblocks.

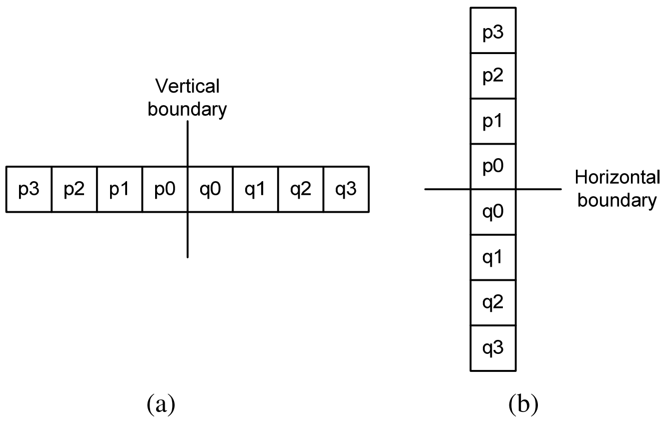
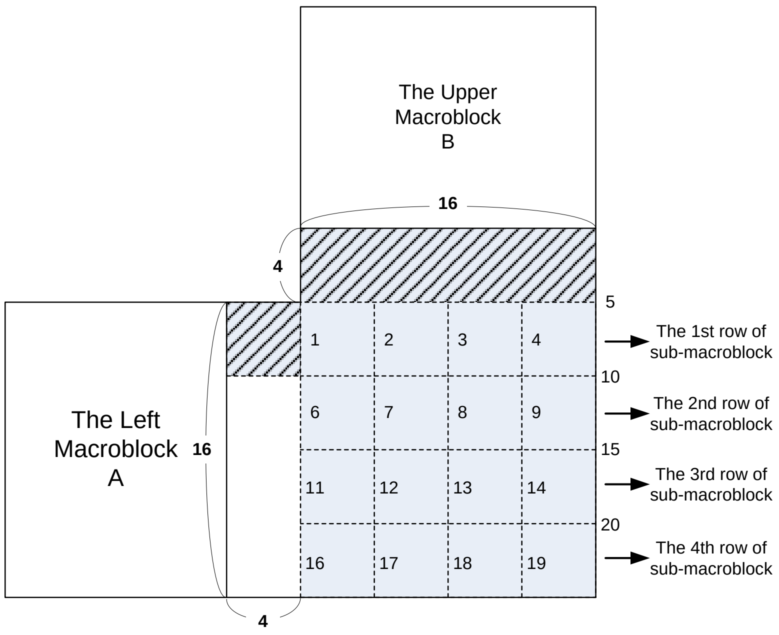


Fig. 3. Pixels used in deblocking filtering (a) Pixels in vertical filtering (b) Pixels in horizontal filtering

Fig. 3(a) shows pixels used in vertical deblocking filter and Fig. 3(b) shows pixels in horizontal deblocking filter. As shown in Fig. 3, 8 consecutive pixels are necessary to the deblocking filter. Thus, the conventional deblocking filtering, especially vertical filtering, using 8 vertical consecutive pixels for a single filtering operation is accompanied 8 memory accesses. As to a single macroblock, the total required memory access cycle is 768. This long latency cause serious problems to real-time processing of high resolution videos such as HDTV programs.

3 Proposed Deblocking Filter

In this paper, register array is composed of 3 parts: the main register, the left register, the upper register. For the filtering operation, the current macroblock is divided into several sub-macroblocks of the same size. Each sub-macroblock are stored into and read from the main register in order. The left boundary data of the sub-macroblock stored in the main register are stored into the left register. The upper boundary data of the sub-macroblock stored in the main register are stored into the upper register. In the upper register, the initial data for the current macroblock comes from the upper macroblock, and then the data coming out from the main register after the sub-macroblock computation is stored.



The Upper Macroblock B
16
4
5
1
2
3
4
The 1st row of sub-macroblock
10
The Left Macroblock A
16
6
7
8
9
The 2nd row of sub-macroblock
15
11
12
13
14
The 3rd row of sub-macroblock
20
16
17
18
19
The 4th row of sub-macroblock
4

Fig. 4. The order of filtering operation for a macroblock

3.1 Implementation of Deblocking Filtering

Fig. 4 shows the order of filtering operation for a macroblock. The dotted line is the horizontal and vertical boundary for filtering operations. As shown in Fig. 4, the current macroblock is divided into 4 rows of sub-macroblocks. The size of a row is 16×4. For the 1st row of sub-macroblocks, horizontal filtering operation is performed sequentially, and then the vertical filtering operation is performed. These filtering operations are applied to the 2nd, 3rd, and 4th row of sub-macroblocks.

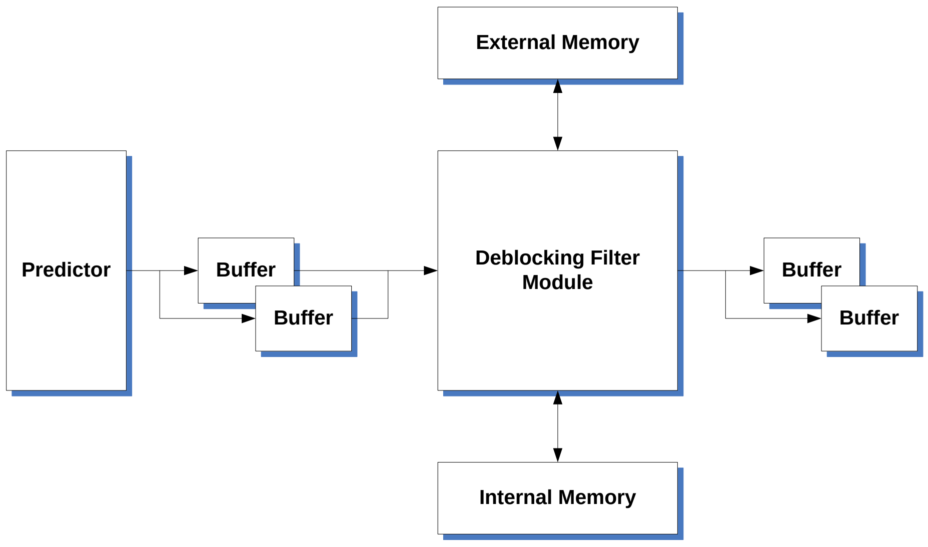


Fig. 5. The block diagram of the deblocking filter module for filtering operation

As shown in Fig. 5, the prediction results are stored into the dual buffer. The luminance and chrominance component of the current macroblock read from the dual buffers are stored into internal register arrays of the deblocking filter module. The left macroblock A and the upper macroblock B adjacent to the current macroblock as shown in Fig. 4 are read from the external memory, stored into the internal memory, and then re-stored into the register arrays.

As discussed in Fig. 4, the current macroblock is divided into 4 rows of sub-macroblocks and then filter operations are performed. The data of the 1st row of the sub-macroblocks are stored into register arrays. From the upper macroblock B, the 16×4 data adjacent to the 1st row of sub-macroblocks are read from the external memory, stored into the internal memory, and then re-stored into register arrays. From the left macroblock A, the 4×16 data adjacent to the current macroblock are stored into the internal memory and then the only upper 4×4 data block are re-stored into register arrays. Therefore, the 4×16 data from the macroblock A and the 16×4 data from the macroblock B are stored into the internal memory. The deblocking filter module performs the filter operation with the data in register arrays and then stores the result into the dual buffer.

3.2 Implementation of the Proposed Register Array

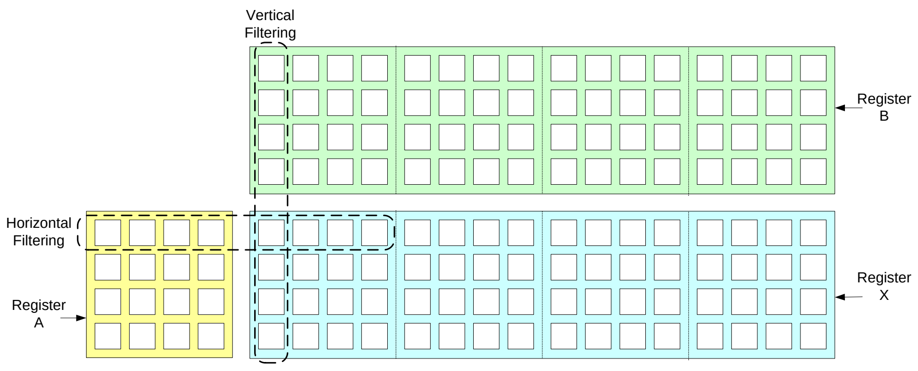


Fig. 6. Horizontal and vertical filtering operation with register arrays

As shown in Fig. 6, the register arrays is composed of 16×4 Register **X**, 4×4 Register **A**, and 4×16 Register **B**. The Register X is decomposed of 4 4×4 sub-macroblocks. It stores the 1st row of sub-macroblocks of the current macroblock which are read from the dual buffer storing the prediction results. Register A stores the upper 4×4 sub-macroblock of the left macroblock in the internal memory. Register B as same as Register X is decomposed to 4 4×4 sub-macroblocks and stores 16×4 data of the upper macroblock in the internal memory.

If each register stores the data for the filtering operation, horizontal filtering operations for the current row of sub-macroblocks in the Register X are performed with the data in Register A. And then, vertical filtering operations are performed with data in Register B.

At first, the horizontal filter operation with the sub-macroblock in Register X takes the data in Register A and the 4 leftmost data in Register X. It performs on the vertical boundary data between 4-pixel data.

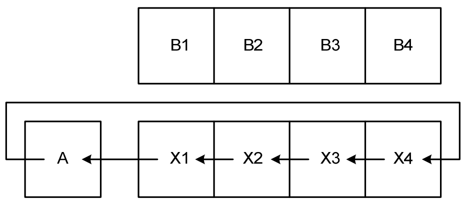


Fig. 7. Shift operation of the register array for horizontal filtering

As shown in Fig. 7, if the filtering operation with the sub-macroblock X1 is completed, the data in each register are shifted left by 4×4. So, the data in X1 sub-macroblock are moved to register A, X2 to X1, X3 to X2, X4 to X3, and then the data in Register A(A1) to X4. After that, the same procedure (filtering $\rightarrow$ shifting) is repeated until the horizontal filtering operations are completed with the data in Register

X. After 4 times shift operation, the data in Register A is moved to the X1 position. Therefore, the data is moved to the initial position by one more left shift operation.

The vertical filtering operation is performed with the row data of sub-macroblocks in Register X. This vertical operation takes the data in Register B. There is one horizontal boundary between data in Register X and Register B. The vertical filtering operation is performed to 16 pixels on this horizontal boundary.

If these horizontal and vertical filtering operations are completed with a 16×4 row of sub-macroblock, the next row of sub-macroblocks are filled into the Register X. The previous data in Register X are moved to the Register B, and the new 4×4 data in Register A are read from the internal memory. The first data in Register A and Register B are sent to the external memory. This overall procedure is repeated with an entire macroblock. Therefore, the movement between the new sub-macroblock data and sub-macroblock data after filtering operation and output from Register A and B are concurrently performed.

3.3 Implementation of Register I/O

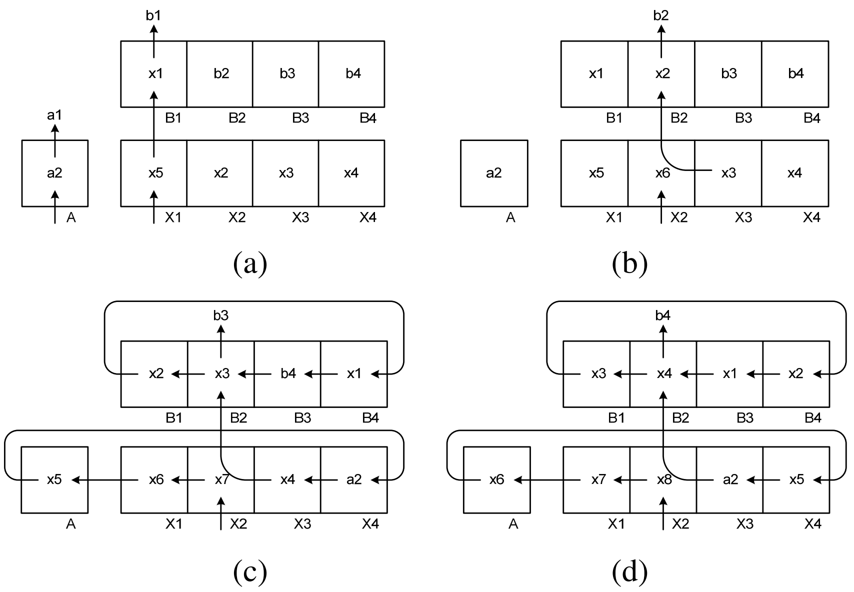


Fig. 8. The I/O operation of register array

Fig.8 shows the I/O operation of the register array. As shown in Fig. 8(a), the left-most sub-macroblock in the 2nd row are stored into register X1 and the next sub-macroblock (a2) of macroblock A in the internal memory is stored into Register A. The previous data (x1) in register X1 is shifted into register B1 and the previous data (b1) in register B1 is sent out the external memory.

As shown in Fig. 8(b), the next sub-macroblock (x6) in the 2nd row are inputted into register X2. The previous data (x2) in register X1 are shifted into register B2 and the previous data (b2) in the register B2 are sent out. At the same time, the horizontal filtering operations are performed on data (a2 and x5) in register A and X1. After filtering operation for register X1, data in each Register A and X are shifted left by 4×4 as shown in Fig. 8(c). The data in Register B is also shifted by 4×4. After the shift operation, the horizontal filtering operation is performed on the x6 and x5 in Register X1 and A, respectively. Concurrently, the 3rd sub-macroblock in the 2nd row is stored into the register X2 and the previous 3rd sub-macroblock (x3) in the 1st row is shifted into register B2. The previous data (b3) in register B2 is sent out.

After fourth sub-macroblock data (x8) of the 2nd row of sub-macroblocks is loaded into register X2, the rest of filtering operation is performed as shown in Fig. 8(d). Once horizontal deblocking filtering of a row of sub-macroblocks is completed, data is shifted to go back their original position. Then vertical deblocking filtering is performed on the data of X-register.

The procedures are repeated with the other rows of sub-macroblocks until processing of the entire macroblock is finished. It should be noticed that data is loaded into X1-register at the initial state. For all other case, data is loaded into X2-register.

4 Performance Analysis

The proposed deblocking filtering can process a single macroblock as followings: First data loading into A-register and X1-register takes 4 cycles. Since four lines of horizontal filtering takes 8 cycles (2 cycles per single position), the total cycle to perform horizontal deblocking filtering of a row of sub-macroblocks is 36. In order to perform vertical deblocking filtering the original location of data should be restored. The shift for restoring data locations takes 1 cycle. There are 16 positions to perform the vertical deblocking filtering. Thus vertical filtering for a row of sub-macroblock takes 32 cycles. With 14 overhead cycles, total cycles to perform deblocking filtering on a single macroblock is 14+4×(37+32)=306. Adding cycles for chrominance components (306×0.5=153) makes the total cycle 459 that is sufficiently short time for processing high resolution sequences.

5 Conclusion

In this paper we propose a high performance register array structure for deblocking filter architecture. The implementation cost of the proposed structure is relatively small compared with the conventional deblocking filter architectures. Data loading, filtering, data movement and data output are concurrently processed to achieve high performance of the deblocking filter architecture. In order to handle the proposed register array structure efficiently, the operation procedure of the proposed structure is also presented.

Acknowledgement

This work is partly supported by Samsung Electronics Co. Ltd. under Blue Architecture Project and partly supported by the Ministry of Information and Communication, Korea, under the ITRC program supervised by the IITA.

References

1. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T) Rec. H.264/ISO/IEC 14 496-10 AVC. Document JVTG050. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG (2003)
2. Wiegand, T., Sullivan, G., Bjntegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 560-576
3. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., Wedi, T.: Video coding with H.264/AVC: Tools, performance, and complexity. IEEE Mag. Circuits and Syst. 4 (2004) 7-28
4. P. List, A. Joch, J.L.G.B., Karczewicz, M.: Adaptive deblocking filter. IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 614-619
5. M. Sima, Y.Z., Zhang, W.: An efficient architecture for adaptive deblocking filter of h.264/AVC video coding. IEEE Trans. Consum. Electron. 50 (2004) 292-296
6. Li, L., Goto, S., Ikenaga, T.: A highly parallel architecture for deblocking filter in H.264/AVC. IEICE Trans. Inf. & Syst. E88-D (2005) 1623-1629
7. Huang, Y.W., Chen, T.W., Hsieh, B.Y., Wang, T.C., Chang, T.H., Chen, L.G.: Architecture design for deblocking filter in H.264/JVT/AVC. In: IEEE Proc. Int. Conf. Multimedia and Expo 2003. Volume 1. (2003) 693-696