Pipelined Macroblock Processing to Reduce Internal Buffer Size of Motion Estimation in Multimedia SoCs

Seongsoo Lee

A multimedia SoC often requires a large internal buffer, because it must store the whole search window to reduce the huge I/O bandwidth of motion estimation. However, the silicon area of the internal buffer increases tremendously as the search range becomes larger. This paper proposes a new method that greatly reduces the internal buffer size of a multimedia SoC while the computational cost, I/O bandwidth, and image quality do not change. In the proposed method, only the overlapped parts of search windows for consecutive macroblocks are stored in the internal buffer. The proposed method reduces the internal buffer size to 1/5.0 and 1/8.8 when the search range is ±64×±64 and ±128×±128, respectively.

Keywords: Multimedia, VLSI, SoC, internal buffer size reduction, motion estimation.

I. Introduction

Block-matching motion estimation [1] is widely used in many image compression standards [2], [3] because of its simplicity and high efficiency. The highest compression ratio can be obtained with the full search algorithm (FS) [1], which exhaustively matches all possible candidates in the search range. However, as the search range becomes larger, the computational cost and the I/O bandwidth greatly increase, which is a serious problem in multimedia system-on-a-chip (SoC) design. To overcome these problems, many papers have reported fast algorithms [4]-[8] that reduce the computational cost substantially. However, they do not efficiently improve the I/O bandwidth requirement. Recently, in many multimedia SoCs, researchers have tried to alleviate the I/O bandwidth problem by storing the whole search window in an internal buffer [9]-[13], but a significant part of the silicon area is occupied by the large internal buffer when the search range is large.

In this paper, we propose a pipelined macroblock processing (PMP) method that efficiently reduces the internal buffer size of multimedia SoCs without increasing the I/O bandwidth. It performs motion estimation on several consecutive macroblocks in parallel. Because the search windows of these consecutive macroblocks overlap, only their overlapped part needs to be stored in the internal buffer. This efficiently reduces the internal buffer size without increasing the I/O bandwidth and can be applied to various multimedia SoCs exploiting the motion estimation.

Manuscript received Jan. 15, 2003; revised Mar. 5, 2003.

This work was supported by the Soongsil University Research Fund.

Seongsoo Lee (phone: +82 2 820 0692, email: sslee@e.ssu.ac.kr) is with School of Electronic Engineering, Soongsil University, Seoul, Korea.

II. The I/O Bandwidth and the Internal Buffer Size of Motion Estimation in a Multimedia SoC

In multimedia applications, the search range for motion estimation should be large enough to cover the motion displacements of fast-moving objects. Figure 1 shows the peak signal-to-noise ratio (PSNR) performance of the MPEG-2 motion estimation, where the picture format is CCIR601 (720×480 pixels, 30 frames/s) and 45 frames are simulated for each sequence. From the simulation results, appropriate search ranges of P-pictures are ±64×±64, and for sequences of one P-picture followed by two B-pictures, they are ±128×±128.

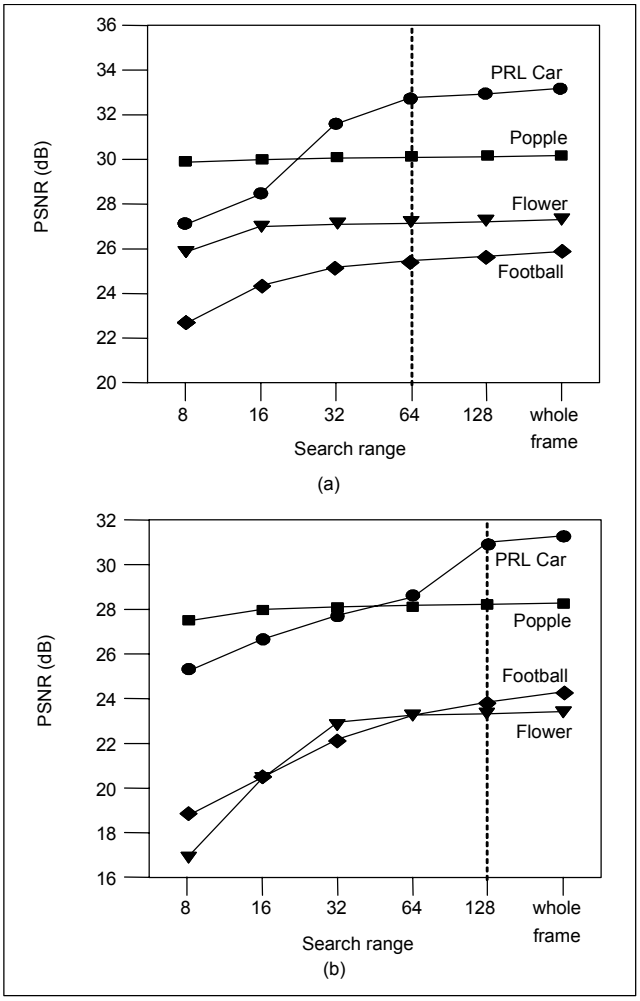

Fig. 1. PSNR performance of the MPEG-2 motion estimation for various search ranges.

Both the computational cost and the I/O bandwidth of the full search algorithm become too large for SoC implementation if the search range exceeds ±64×±64 (Fig. 2). To reduce the computational cost, various researchers have proposed fast algorithms, such as the 4:1 alternate subsampling search (4:1AS) [6] algorithm, the 4:1 search window subsampling search (4:1SS) [8] algorithm, the one-dimensional full search (1DFS) [5] algorithm, and the three-stage hierarchical block-matching algorithm (3HBMA) [7]. As Table 1 shows, these fast algorithms reduce the computational cost to 1/4–1/163, but the required I/O bandwidth is not much reduced, assuming no internal buffer is exploited to store pixel data.

The I/O bandwidth is greatly reduced if the search window

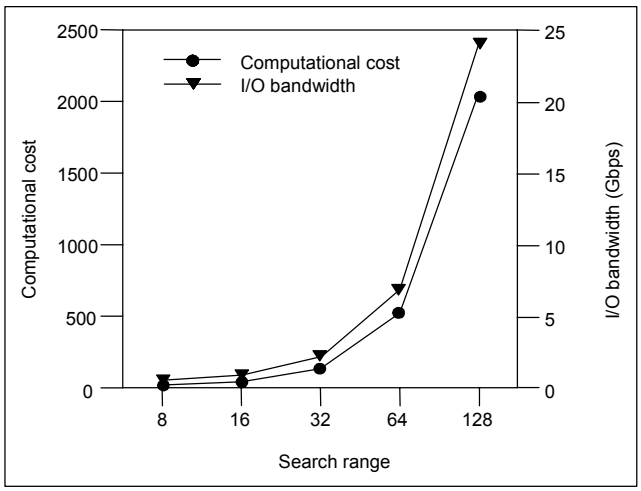

Fig. 2. Computational cost and I/O bandwidth of the MPEG-2 motion estimation for various search ranges.

Table 1. Computational cost, I/O bandwidth, and PSNR performance for various motion estimation algorithms.

Algorithm		FS	4:1AS	4:1SS	1DFS	3HBMA
±64 ×±64 search range (MPEG2 P-picture without B-pictures)	Computational cost (ops)	5.10×10^{11}	1.28×10^{11}	1.28×10^{11}	1.19×10^{10}	3.49×10^{9}
	I/O bandwidth (bps)	6.80×10^{9}	7.13×10^{9}	2.09×10^{9}	2.41×10^{9}	7.05×10^{9}
	PSNR performance (dB)	28.85	28.52	28.29	28.13	27.25
±128 ×±128 search range (MPEG2 P-picture with two B-pictures)	Computational cost (ops)	2.04×10^{12}	5.10×10^{11}	5.10×10^{11}	2.38×10^{10}	1.25×10^{10}
	I/O bandwidth (bps)	2.41×10^{10}	2.73×10^{10}	7.13×10^{9}	4.40×10^{9}	2.43×10^{10}
	PSNR performance (dB)	26.57	26.16	25.95	25.38	25.00

Table 2. I/O bandwidth and internal buffer size when the search window data are stored in an internal buffer.

Algorithm		FS,4:1AS,1DFS,3HBMA	4:1SS
±64 ×±64 search range (MPEG2 P-picture without B-pictures)	I/O bandwidth (bps)	8.29×10^8	2.70×10^8
	Internal buffer size (kbit)	188.4	50.2
±128 ×±128 search range (MPEG2 P-picture with two B-pictures)	I/O bandwidth (bps)	1.49×10^9	4.35×10^8
	Internal buffer size (kbit)	630.8	160.8

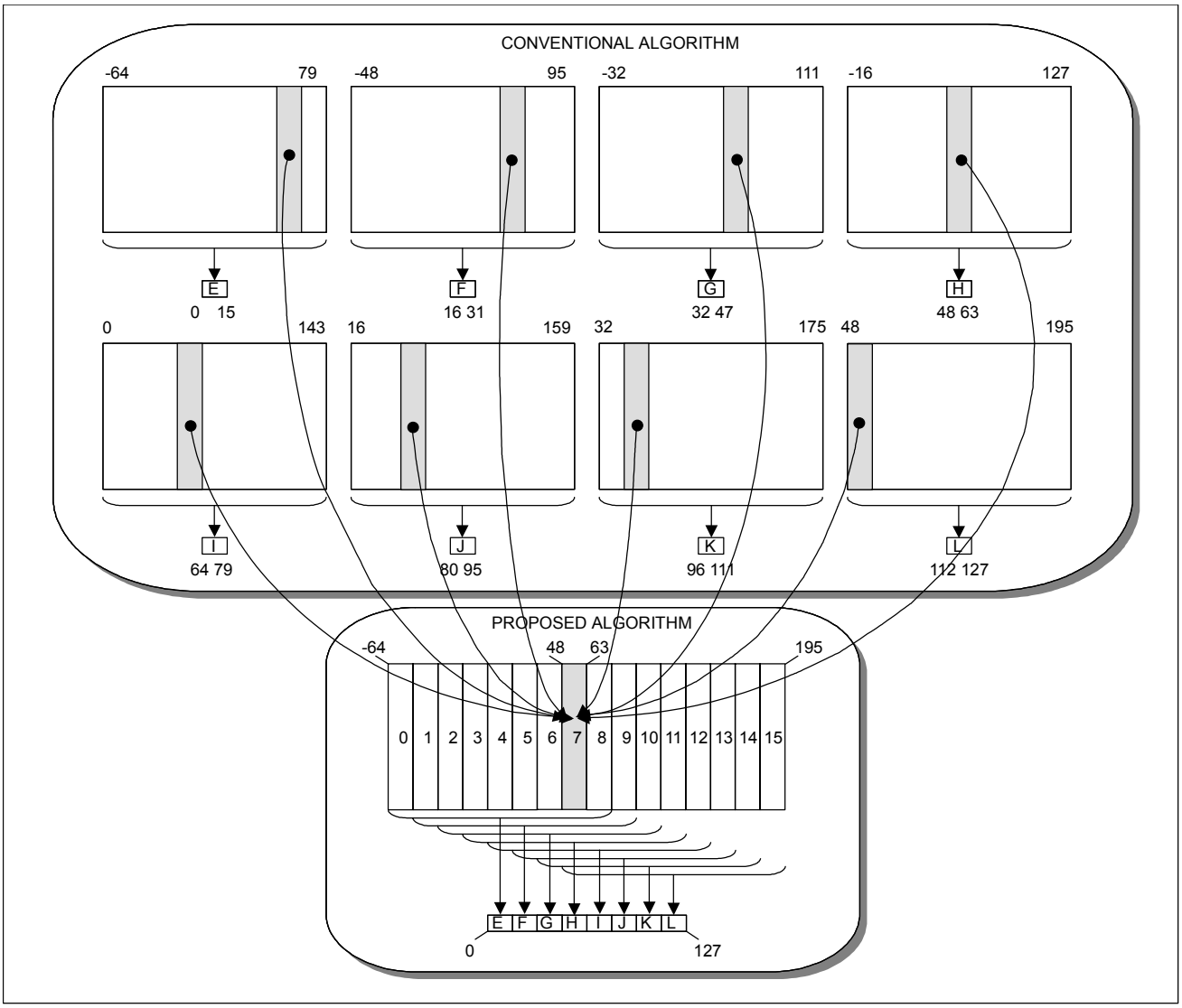

Fig. 3. Overlap of the search window when the search range is ±64 ×±64.

data are stored in an internal buffer (Table 2). However, the size of the internal buffer is so large that its silicon area becomes dominant in SoC implementation if the search range exceeds ±64×±64. Consequently, a new algorithm with a substantially reduced internal buffer size is required.

III. The Pipelined Macroblock Processing Method

In the conventional multimedia SoC design, motion estimation for each macroblock is processed sequentially on a macroblock basis. For a search range of ±M×±M pixels and a

macroblock size of N×N pixels, the current macroblock and its whole search window data of (2M+N)×(2M+N) pixels must be stored in the internal buffer for the motion estimation of the current macroblock. The next macroblock and its additional search window data of N×(2M+N) pixels must be imported into the internal buffer before starting motion estimation on it. Therefore, the required internal buffer size is $2\{N^2+(M+N)\times(2M+N)\}$ pixels.

In this paper, we propose an effective method for internal buffer reduction called pipelined macroblock processing (PMP), in which 2M/N macroblocks are processed in parallel. The internal buffer size can be reduced by using the fact that the search windows of consecutive macroblocks overlap. Figure 3 illustrates the overlap of the search window when the search range is ±64×±64 (M=64) and the macroblock size is 16×16 (N=16).

The search operation for a column of search positions is performed over 2M/N macroblocks in parallel. Because N×(2M+N) pixels of the search windows are overlapped for 2M/N macroblocks, only consecutive macroblocks of 2M×N pixels and the overlapped search windows of the N×(2M+N) pixels need to be stored in the internal buffer. Additional search window data of 2M+N pixels must be imported into the internal buffer before starting the search operation for the next column of search positions. Similarly, the next macroblock of N^2 pixels must be imported before starting the motion estimation for the next macroblock. Therefore, the total internal buffer size is reduced to (2N+1)×(2M+N) pixels. The reduction ratio of the internal buffer size is $(2N+1)\times(2M+N)/2\{N^2+(M+N)\times(2M+N)\}$, while the computational cost, the I/O bandwidth, and the PSNR degradation do not increase at all. Figure 4 shows the internal buffer reduction when the search range is ±64×±64 (M=64) and the macroblock size is 16×16 (N=16).

In the proposed method, the search operation is performed column by column, that is, the leftmost column of search positions (–M,–M) – (–M,M–1) is processed first, then the next column of search positions (–M+1,–M) – (–M+1,M–1) is processed, and so on. Therefore, the proposed method can be applied to all motion estimation algorithms in which the search operation is performed column by column. Although it is applied only to the full search algorithm in this paper, with only minor modification, it can also be applied to many fast algorithms (Table 3). Table 4 shows the reduction of internal buffer sizes when the proposed method is applied to various motion estimation algorithms. The internal buffer size for the full search algorithm is reduced to 1/5.0 and 1/8.8 of the original sizes when the search range is ±64×±64 and ±128×±128, respectively.

The proposed method can be applied to various motion

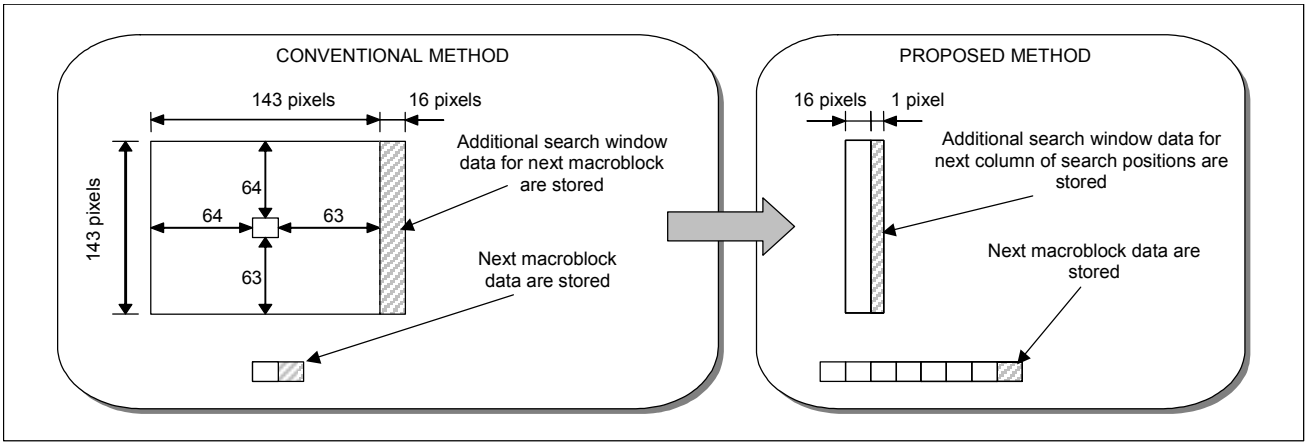

Fig. 4. Reduction of the internal buffer size when the search range is ±64 ×±64.

Table 3. Motion estimation algorithms to which the proposed method can be applied.

The proposed method can be fully applied to:	The proposed method can be partially applied to:	The proposed method cannot be applied to:
• full search [1] • 4:1 alternate subsampling search [6] • 4:1 search window subsampling search [8]	• three-stage hierarchical block-matching [7] (only first stage)	• one-dimensional full search [5] • three-step search [4]

Table 4. Reduction of the internal buffer size for various motion estimation algorithms.

Algorithm		FS, 4:1AS, 3HBMA	4:1SS	1DFS
±64 ×±64 search range (MPEG2 P-picture without B-pictures)	Conventional method (kbit)	188.4	50.2	188.4
	Proposed method (kbit)	38.0	23.6	N/A
±128 ×±128 search range (MPEG2 P-picture with two B-pictures)	Conventional method (kbit)	630.8	160.8	630.8
	Proposed method (kbit)	71.8	44.6	N/A

estimation algorithms, so the possible implementation methods depend on the basis algorithm (=original motion estimation algorithm before applying the proposed method). In general, the hardware and software overhead of the proposed method is not large, since only the dataflow control part is modified. When n processing elements are used in the basis algorithm, the proposed method also employs the same number and same type of processing elements. In most cases, the processing elements occupy the majority of the silicon area. The overhead of the dataflow control part is not negligible, but it is unlikely to increase the total silicon area too much when compared to the conventional algorithms.

In Fig. 4, the internal buffers import the pixel data for the next macroblock from frame memory while processing the current 8 macroblocks (shaded area in Fig. 4). The amount of the memory access of the proposed method is the same as the conventional algorithms, but the order of the memory access of the search window data is different. As shown in Fig. 4, the proposed method accesses the search window data in 1×143 order, i.e., (0,0)-(0,1)-(0,2)-…-(0,142), while conventional algorithms access in 16×143 order, i.e., (0,0)-(1,0)-…-(15,0)-(0,1)-(1,1)-…-(15,1)-(0,2)-…-(15,142).

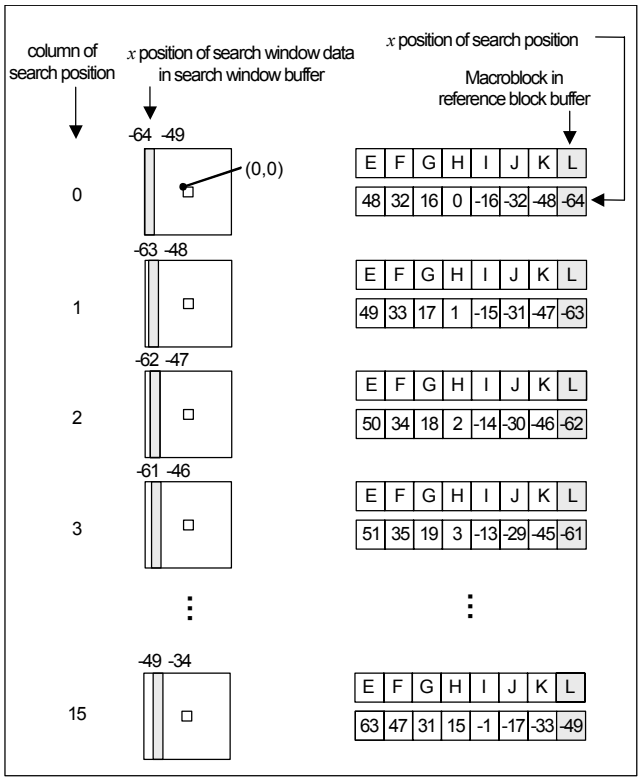

Fig. 5. Operations for the first 16 columns of search positions.

IV. The Detailed Operations of the Pipelined Macroblock Processing Method

Figure 5 illustrates how to process the first 16 columns of the search positions of macroblock L. When the column 0 of the search positions is processed, macroblocks E-L are stored in the reference block buffer, and the search window data corresponding to the column 0 of the search positions of macroblock L are stored in the search window buffer. At this time, the search positions of x = –64 of macroblock L are processed. Simultaneously, the search positions of x = 48, x = 32, x = 16, x = 0, x = –16, x = –32, and x = –48 of macroblocks E, F, G, H, I, J, and K are processed, respectively. Note that the search window data are common although the search positions of x displacement are different for macroblocks E-L.

When the column 1 of the search positions is processed, the search window data corresponding to the column 1 of the search positions of macroblock L are stored in the search window buffer. At this time, the search positions of x = –63 of macroblock L are processed. Simultaneously, the search positions of x = 49, x = 33, x = 17, x = 1, x = –15, x = –31, and x = –47 of macroblocks E, F, G, H, I, J, and K are processed, respectively.

Similarly, the columns 2 to 15 of the search positions of macroblock L are processed. When the column 15 of the search position is processed, the search window data corresponding to the column 15 of the search positions of macroblock L are stored in the search window buffer. At this time, the search positions of x = –49 of macroblock L are processed. Simultaneously, the search positions of x = 63, x = 47,

$x = 31$, $x = 15$, $x = -1$, $x = -17$, and $x = -33$ of macroblocks E, F, G, H, I, J, and K are processed, respectively.

Figure 6 illustrates how to process the macroblock change in the reference block buffer. When the column 15 of the search positions is processed, macroblock E is finished. After that, the column 16 of the search positions is processed, and macroblock M begins. At this time, macroblocks F-M are stored in the reference block buffer, and the search positions of $x = -48$ of macroblock L are processed. Simultaneously, the search positions of $x = 48$, $x = 32$, $x = 16$, $x = 0$, $x = -16$, $x = -32$, and $x = -64$ of macroblocks F, G, H, I, J, K, and M are processed, respectively.

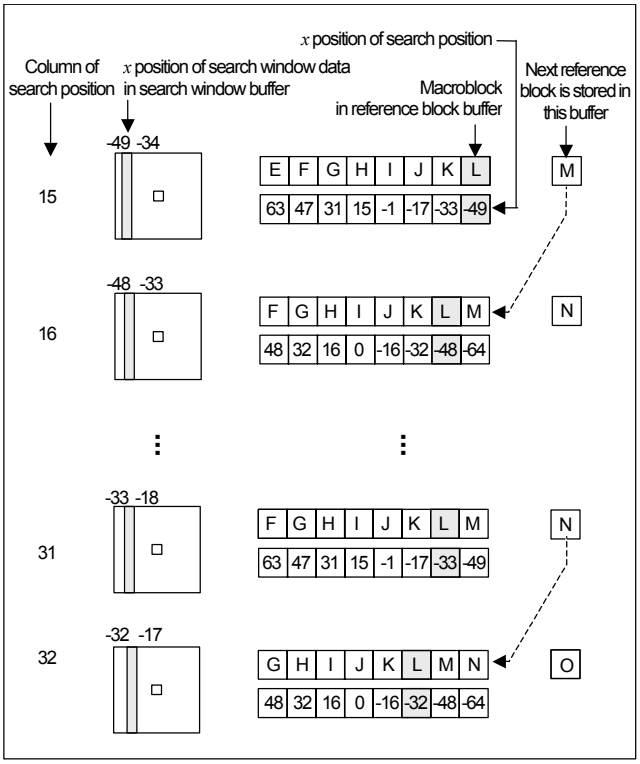

Fig. 6. Operations for the macroblock change in the reference block buffer.

Figure 7 illustrates how to process the last 16 columns of the search positions of macroblock L. When the column 127 of the search positions is processed, macroblocks L-S are stored in the reference block buffer, and the search window data corresponding to the column 127 of the search positions of macroblock L are stored in the search window buffer. At this time, search positions of $x = 63$ of macroblock L are processed. Simultaneously, the search positions of $x = 47$, $x = 31$, $x = 15$, $x = -1$, $x = -17$, $x = -33$, and $x = -49$ of macroblocks M, N, O, P, Q, R, and S are processed, respectively. After processing the column 127 of the search positions, all the search positions of macroblock L are finished, and macroblock L is eliminated from the reference block buffer.

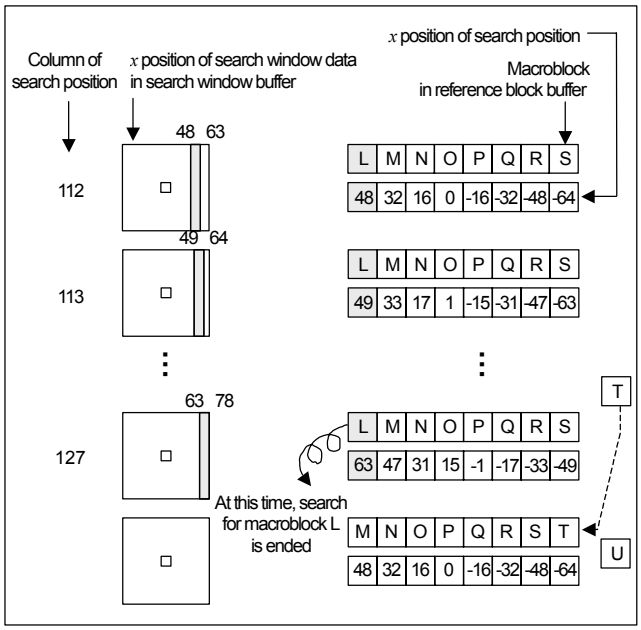

Fig. 7. Operations for the last 16 columns of search positions.

The proposed method has no additional latency at the frame boundary, since it does not perform motion estimation when the search position exceeds the frame boundary. Figure 8 illustrates how to process macroblock H at the rightmost frame boundary, where T is the time required to perform the search operations for one column of search positions. For t ranging from 65T to 79T, no motion estimation is performed, and during this interval, the new search window data for macroblock I, J, K, L, and M are transferred into the search window buffer.

V. Conclusion

In this paper, we proposed an effective method that reduces the internal buffer size of the multimedia SoC. Our method exploits the overlap of search windows of consecutive macroblocks. The proposed method stores only the overlapped parts of the search windows in the internal buffer and performs motion estimation for consecutive macroblocks in parallel. It reduces the internal buffer size of multimedia SoCs to 1/5.0 and 1/8.8 times when the search range is $\pm 64 \times \pm 64$ and $\pm 128 \times \pm 128$, respectively. It does not increase the computational cost or the I/O bandwidth. It does not degrade the PSNR performance nor does it require any additional latency at the frame boundary. The proposed method makes it feasible to design high-performance multimedia SoCs with a large search range.

References

[1] J. Jain and A. Jain, "Displacement Measurement and its

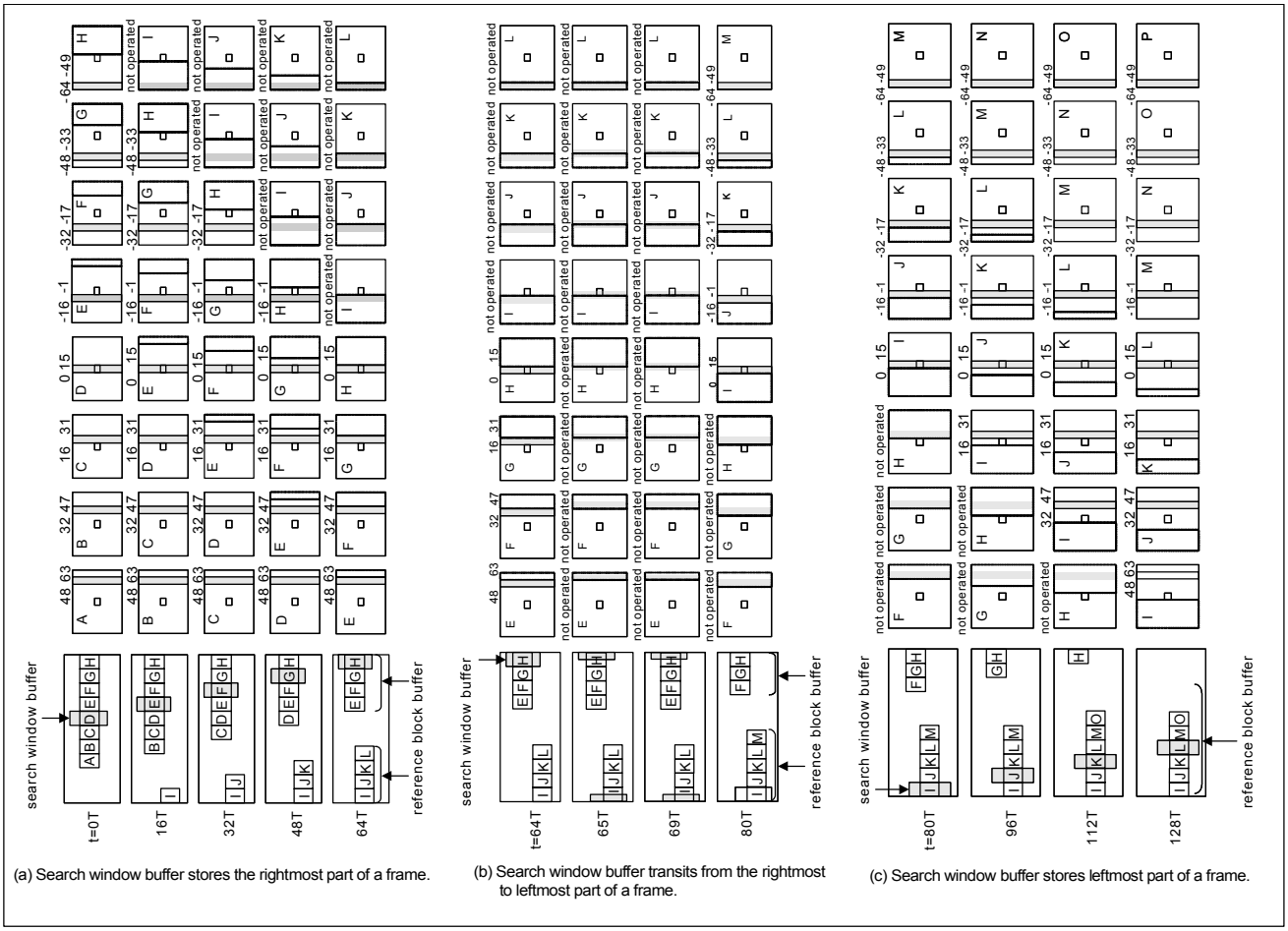

Fig. 8. Operations in the frame boundary.

Application in Interframe Image Coding," *IEEE Trans. Commun.*, vol. COM-29, no. 12, Dec. 1981, pp. 1799-1808.

[2] ISO/IEC JTC1/SC29/WG11 13818-1, *Coding of Moving Pictures and Associated Audio*, Nov. 1994.

[3] CCITT SG XV, *Recommendation H.261-Video Codec for Audiovisual Services at p*64 kbit/s*, Aug. 1990.

[4] T. Koga, "Motion Compensated Interframe Coding for Video-Conferencing," *Proc. National Telecommunication Conf.*, 1981, pp. G5.3.1-G5.3.5.

[5] M. Chen, L. Chen, and T. Chiueh, "One-Dimensional Full Search Motion Estimation Algorithm for Video Coding," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 4, no. 5, Oct. 1994, pp. 504-509.

[6] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimation of Block Motion Vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 2, Apr. 1993, pp. 148-157.

[7] M. Biering, "Displacement Estimation by Hierarchical Block Matching," *Proc. SPIE Visual Communication and Image Processing*, 1988, pp. 942-951.

[8] H. Jong, L. Chen, and T. Chiueh, "Accuracy Improvement and Cost Reduction of 3-Step Search Block Matching Algorithm for Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 1, Feb. 1994, pp. 88-90.

[9] H. Lin and B. Petryna, "A 14 GOPS Programmable Motion Estimator for H.26x Video Coding," *Proc. Int'l Solid-State Circuits Conf.*, 1996, pp. 246-247.

[10] T. Onoye et al., "Single Chip Implementation of Motion Estimator Dedicated to MPEG2 MP@HL," *IEICE Trans. Fundamentals*, vol. E79-A, no. 8, Aug. 1996, pp. 1210-1216.

[11] M. Mizuno et al., "A 1.5-W Single Chip MPEG-2 MP@ML Video Encoder with Low Power Motion Estimation and Clocking," *IEEE J. of Solid-State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1807-1816.

[12] P. Kuhn et al., "A Flexible Low-Power VLSI Architecture for MPEG-4 Motion Estimation," *Proc. SPIE Visual Communication and Image Processing*, 1999, pp. 883-894.

[13] Y. Lai, "A Memory Efficient Motion Estimator for Three Step Search Block-Matching Algorithm," *IEEE Trans. Consumer Electron.*, vol. 47, no. 3, Aug. 2001, pp. 644-651.

Seongsoo Lee received the BS, MS, and PhD degrees in electrical engineering from Seoul National University, Korea in 1991, 1993, and 1998. From 1998 to 2000, he was a Research Associate in the Institute of Industrial Science, University of Tokyo, Japan. From 1998 to 2002, he was a Research Professor in Department of Information Electronics Engineering at Ewha Womans University in Korea. Since 2002, he has been an Assistant Professor in the School of Electronics Engineering, Soongsil University, Korea. His research interests include low-power VLSI systems, low-power wireless communications, multimedia signal processing, wireless sensor network, high-speed circuits, and signal integrity.