International Conference on Intelligent Control and Information Processing
August 13-15, 2010 - Dalian, China

Scalable Parallel Processing for H.264 Encoding Application to Multi/Many-core Processor

Yukihiro Takeuchi, Yohei Nakata, Hiroshi Kawaguchi, and Masahiko Yoshimoto

***Abstract*—Although valuable, the high-quality video compression format H.264/AVC workload complicates real-time encoding. This paper describes scalable parallel processing for H.264/AVC. Macroblock (MB)-level decomposition is more scalable than conventional methods for increasing the number of multiple threads. Moreover, it presents memory bandwidth advantages. This parallel algorithm can be improved using a motion estimation algorithm that distributes the workload among threads. Complementary recursive cross search (CRCS) is used to achieve efficient video encoding using MB-level decomposition. With and without B-frames for HDTV, MB-level decomposition with CRCS can respectively increase the frame rate of the conventional method by 2.4 and 4.6 times. Furthermore, the method suppresses memory accesses despite higher processing efficiency. Results show that MB-level decomposition with CRCS is suitable for computing in the many-core processor era.**

I. Introduction

In H.264/AVC (H.264) encoding, more than ten times the workload of conventional MPEG2 is necessary for higher picture quality and a lower bitrate [1]. To process the high workload of H.264 video coding, a multicore processor is used to exploit multiple threads simultaneously. Using a multicore processor, H.264 can be encoded efficiently because it can be encoded with multiple threads. Therefore, a data-level decomposition algorithm is easily used for load-balancing for parallel processing in multicore processors [2]. For this reason, we specifically examine data-level decomposition.

One data decomposition granularity of H.264 is frame-level decomposition. Encoding with frame-level decomposition, however, causes saturation of the processing speed and increases the number of memory accesses. Another data decomposition granularity of H.264 is macroblock (MB)-level decomposition [3]. In this decomposition granularity, the basic unit assigned with each thread is at the MB level. It is scalable for processing speed despite the more numerous multiple threads. Another advantage is that it uses little cache memory capacity. In this decomposition granularity, a thread must be synchronized with other threads if the workload of each thread is unbalanced and overhead is increased. In H.264, the motion estimation (ME) workload dominates the overall encoding workload. The ME workload varies according to the sequence. Therefore, a well-balanced ME algorithm is required.

Manuscript received May 1, 2010.

Y. Takeuchi, Y. Nakata and H. Kawaguchi are the Graduate School of System Informatics, Kobe University, Kobe, Hyogo 657-8501, Japan (e-mail:takeuchi_y@cs28.cs.kobe-u.ac.jp).

M. Yoshimoto is the Graduate School of System Informatics, Kobe University, Kobe, Hyogo 657-8501, Japan, and JST, CREST.

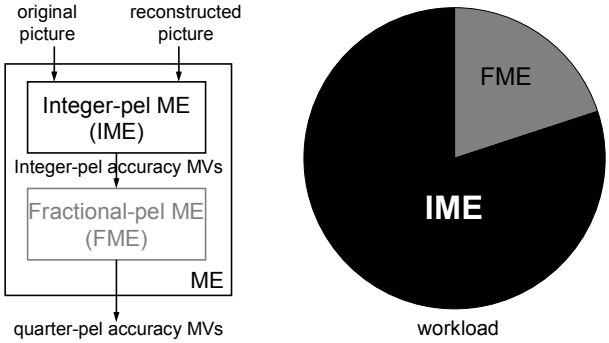


Fig. 1. Workload required for ME.

Fig. 1 presents a breakdown of the workload in a conventional ME. Integer-pel ME (IME) outputs integer-pel accuracy motion vectors (MVs). Using them, Fractional-pel ME (FME) calculates the quarter-pel accuracy MVs. According to the workload analysis for baseline profile encoding [4], IME occupies about 80% in ME. If the IME workload is balanced, the synchronization overhead of each thread is suppressed; then H.264 with MB-level decomposition can be processed efficiently.

Complementary recursive cross search (CRCS) was proposed in an earlier report [5]. This ME algorithm is well balanced in each MB workload. Therefore, the workload of each thread using CRCS becomes well balanced. For that reason, CRCS is suitable for MB-level decomposition. This paper presents a proposal of MB-level decomposition with CRCS that is more scalable and which entails fewer memory accesses than conventional methods.

As described in this paper, the parallelizing algorithm for H.264 is presented in Section II. Details of the CRCS algorithm are explained in Section III. Section IV presents the evaluation results. Section V concludes this paper.

II. Parallelization approach

Many parallelizing approaches are useful for H.264. Well-known approaches are task-level decomposition and data-level decomposition. However, in the task-level approach for H.264, the workload of each function is variable, making the ME function workload much higher than those of other functions. It is difficult to achieve a balance among many threads. Furthermore, decomposing a function to improve parallelism efficiency is difficult [2]. Therefore, we

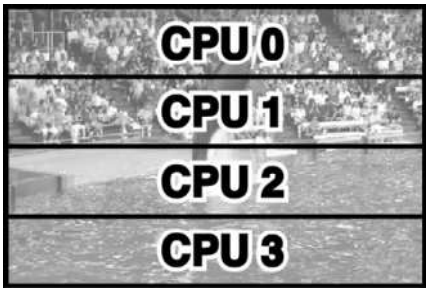


Fig. 2. Slice-level decomposition.

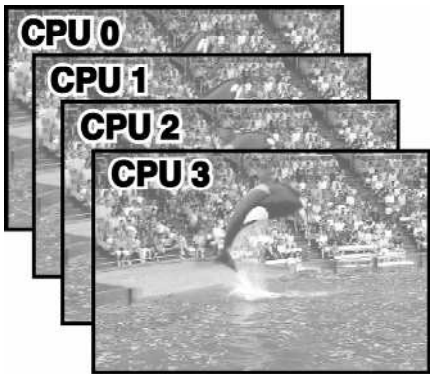


Fig. 3. Frame-level decomposition.

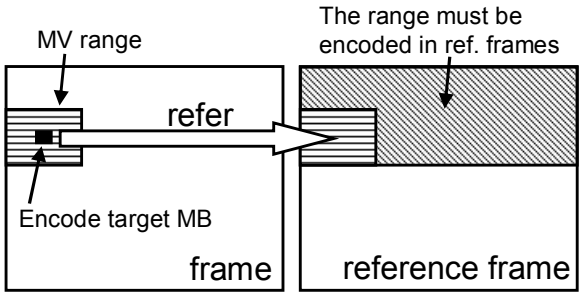
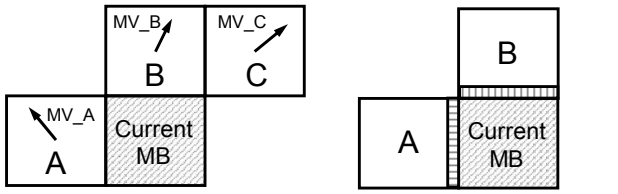


Fig. 4. Required range for synchronization of encoding.

chose data-level decomposition. The parallelizing approach of data-level decomposition for H.264 uses data independence of various granularities such as those used for groups of pictures, frames, slices, and MBs. These granularities are assigned to multiple threads and features efficiency of video encoding.

A. Slice-level decomposition

Slice-level decomposition is an encoding method by which one frame is divided into several slices. Then the slices are processed in parallel. Fig. 2 portrays a frame that is divided into four slices. Each slice is assigned one processor core. Because each slice is independent of other slices, they can be processed in parallel. Under slice-level decomposition, each thread executes the same kind of encoding process. Consequently, highly parallel performance can be achieved. This method, however, presents coding inefficiencies because the entropy coding table is divided into slice levels and MB information can not be referred on slice boundaries. Therefore, the bit rate of encoded results is increased if the slices from



(a) Intra- and MV dependencies (b) Deblocking filter dependencies

Fig. 5. Data dependencies of MB.

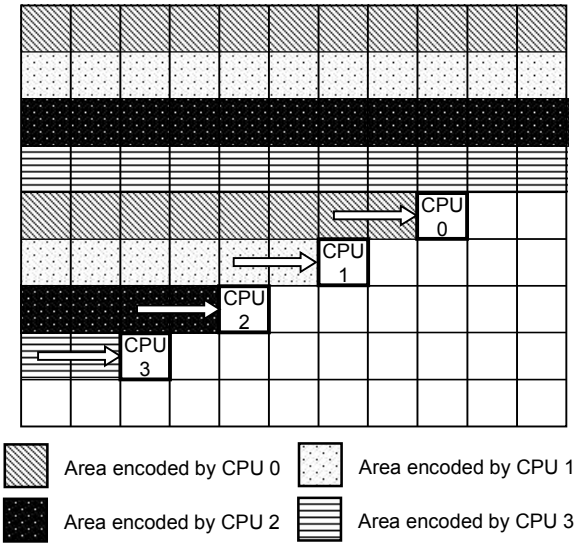


Fig. 6. Macroblock-level decomposition.

one frame are increased [6]. Furthermore, synchronization overhead increases because all threads must complete their tasks before encoding of the subsequent frame.

B. Frame-level decomposition

Under frame-level decomposition, one frame is assigned to one thread. Fig. 3 depicts four frames assigned to four processor cores. This figure presents an example in which one processor core executes encoding of one frame. This method is exploited in x264 [7].

In this method, when more than a certain number of MBs of reference frames targeted by the next frame are encoded, the next frame can be started (Fig. 4). The next frame can encode one MB line if the encoding of MB lines of reference frames—including the MV range referenced by the next frame—is completed. Therefore, if the MV range is increased, then the synchronization overhead of each thread increases.

If the reference frame number is increased, then encoding slows because the waiting time of encoding of reference frames is increased. This method has another feature: the coding efficiency is not degraded because the frame is not divided. Therefore, the bitrate is not increased if the number of threads executing encoding frames is increased. Moreover, using this method, memory usage and memory bandwidth increase when parallel processing tasks increase. Reference frames referred by each thread generally differ because each thread encodes a different frame individually. Many reference

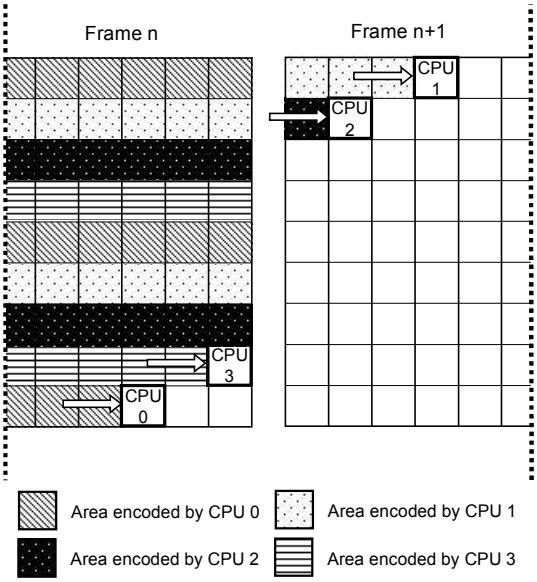


Fig. 7. Inter-Frame transition of threads.

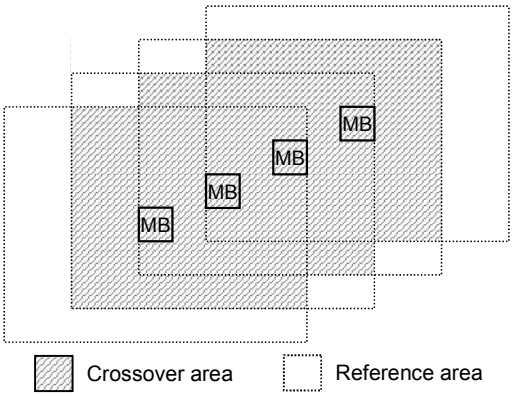


Fig. 8. Reference frames used in encoding.

frames are therefore loaded into cache memory, thereby increasing the memory bandwidth.

C. Macroblock-level decomposition

In H.264, there are several dependencies on adjacent MBs [8]. Fig. 5 presents the three kinds of data dependencies of MB. Left, upper-left, upper, and upper-right MBs are dependent because they are used in intra-prediction and MV prediction. Other dependencies are attributable to the deblocking filter (Fig. 5(b)). Therefore filtering is processed using the bottom row of the upper MB and the right row of the left MB; the upper MB and the left MB are dependent. For reasons listed above, MBs must wait until adjacent MBs are completely encoded.

Macroblock-level decomposition is the method by which one MB is assigned to one thread [3]. Using this method, the MB processing order is special, however, because it must adhere to the restriction explained above. Fig. 6 shows the data processing order of this method with four processor cores. First, encoding is started from the top-left MB. If the upper-right MB in the upper MB line is finished encoding,

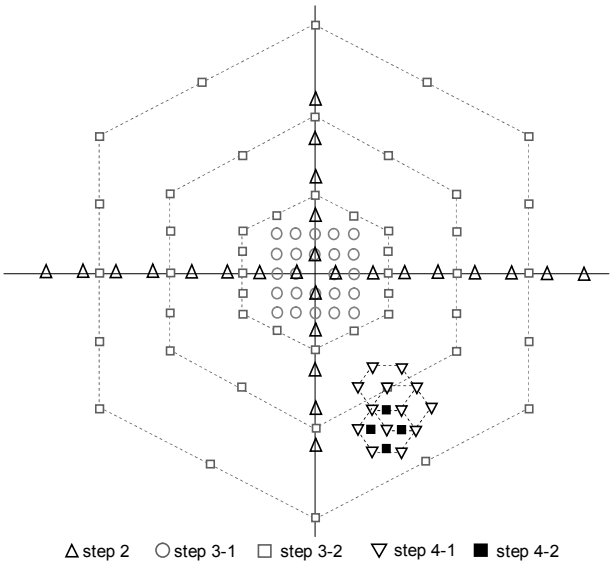


Fig. 9. Search process of the UMHS algorithm.

then encoding of next MB line can be processed. In other words, encoding of the current MB must wait until the upper-right MB encoding is finished. If all MBs in one MB line are finished, then the next unprocessed MB line is started encoding. If the unprocessed MB line does not exist in the current encoding frame, then encoding of the next frame is started as shown in Fig. 7.

In this algorithm, the amount of memory transferred between the processor and the main memory is low. Fig. 8 shows the assigned reference frames by four threads for MB-level decomposition. Most of the reference frames of each thread are overlapped. Therefore, if the cache-memory is shared by processor cores, then the cache usage rate and memory transfer rate are low because reference frames are overlapped. Therefore, as described above, using this algorithm increases the likelihood for the embedded system that cache-memory and memory-bandwidth are restricted.

III. Motion Estimation Algorithm

In MB-level decomposition, the following MB encoding must wait if the leading MB encoding is late: that delay is transmitted to all threads. Therefore, to achieve high processing efficiency, each thread must process well-balanced encoding. The dominant encoding process of H.264 is motion estimation, of which IME is the main one. Consequently, MB-level decomposition requires a well-balanced IME.

A. Unsymmetrical-cross multi-hexagon grid search (UMHS)

A well-known conventional IME algorithm is UMHS [9]; it includes the four sequentially executed steps described below. (1) Initial search point decision: This decides the initial search points for step 2 and step 3. Candidates that indicate the initial search point are the following: Median prediction vector (median vector of the left MV, the upper MV, the upper-right MV), 0 vector (indicates (0, 0) position), upper-layer prediction vector (MV predicted by other block mode in the

same MB), neighboring reference – picture prediction vector.
(2) Unsymmetrical-cross search: Two-pixel interval search is executed (Fig. 9, step 2). In this step, the number of search points is search_range/2 in the horizontal direction, and search_range/4 in the vertical direction.
(3-1) Narrow-range full search: This executes a full search that adjusts to ±2×±2 pixels (Fig. 9, step 3-1).
(3-2) Uneven Multi-Hexagon-grid search: This hexagonal search is executed as shown in Fig. 9, step 3-2. In this step, the search center point is tied up at first. Similar to step 2, the horizontal search points become less dense.
(4-1) Extended Hexagon-based search: This searches six pixels that surround the center point while the minimum evaluated value of six pixels is smaller than the evaluated value of the center point (Fig. 9, step 4-1). The point at which the evaluation value is the smallest among six pixels becomes the next center point.
(4-2) Diamond search: search four pixels which surround the center point while the minimum evaluated value of four pixels is smaller than the evaluated value of the center point (Fig. 9, step 4-2). The minimum evaluated point obtained in this step is the result point of the UMHS method.

In fact, UMHS is superior in terms of the search accuracy, even though it is fast and has a low average workload. As described above, UMHS comprises step 1 to step 3 and hexagonal search. Hexagonal search is an effective algorithm for image quality performance for a larger motion sequence. The UMHS has an early termination (ET) strategy: when switching from step 1 to step 2, or from step 2 to step 3-1, or from step 3-1 to step 3-2, or evaluation in step 3-2 if the conditions are met, it moves to step 4-1 or step 4-2. By ET, the average workload of UMHS is low. In a hexagonal search, the conditional branch, however, is frequent. The next search point is indeterminate until the prior search is completed; pixel reusability is low. In addition, the worst computational load is extremely large because this algorithm is targeted to reduce the average computational load. Because of these features, the UMHS workload varies greatly according to the encoding sequence. An elementary unit of H.264 is assigned with each thread. Therefore, to achieve highly parallel efficiency, it is desired that processes executed by each thread be evenly divided. As described above, the UMHS workload varies greatly. If this algorithm applies to MB-level decomposition, workloads of each thread fluctuate and parallelism efficiency decreases. Therefore, some other IME algorithm for which the workload of each thread is well balanced, e.g. the average workload is the same for the worst computational load, is desired for MB-level decomposition.

B. Complementary Recursive Cross search (CRCS)

In this subsection, we describe the recursive cross search (RCS) and CRCS [5], which are based on a gradient search method with enhanced parallelism. The RCS is presented in Fig. 10. The RCS algorithm operation is described as follows. First, the points of 62.5% within the search range in the horizontal direction are evaluated using one-directional search (1-DS) which detects MVs in continuous points on a straight line (step 1). In step 1, the search center point is determined similarly to UMHS. Secondly, the search continues over the points of 40% in the first step in the vertical direction, taking the point with minimal evaluated value in step 1 as the search center (step 2). And next, points of the same number as the step 2 in the horizontal direction whose center is result of step 2 are searched (step 3). For instance, in case of HDTV (search range is ±128 × ±128), ±80 points in the horizontal direction in step 1, ±32 points in the vertical direction in step 2, and horizontal ±32 points in step 3 are evaluated.

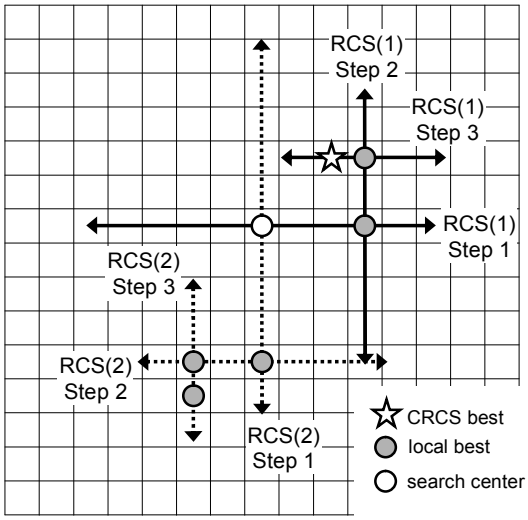
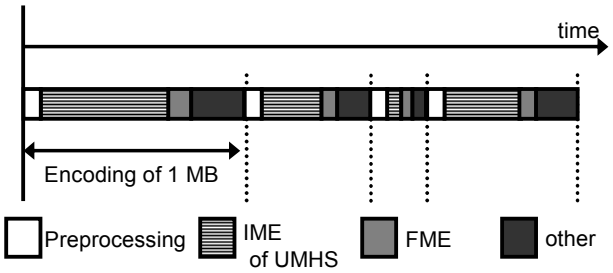


Fig. 10. Search process of the CRCS algorithm.

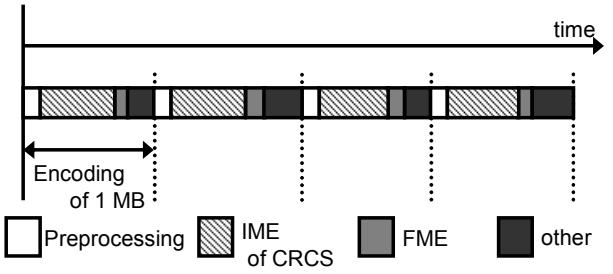
The RCS result is the point with the smallest evaluated value among all search points. After processing of two RCSs, a narrow-range full search (±4 × ±4 pixels) is executed to enhance the image quality. Whether higher picture quality is provided by RCS starting from the horizontal direction or the vertical direction depends on image sequence characteristics. Therefore, the CRCS algorithm is used to improve picture quality by complementarily using two RCSs starting with the horizontal direction search and the vertical direction search, respectively. In CRCS, two RCSs are executed with the same initial point. After the search, the vector with the smaller evaluated value of the two vectors of RCSs is chosen as a CRCS result.

The average workload of this algorithm is lower than that of UMHS. In addition, the average workload and the worst workload of this algorithm are almost identical. Therefore, the workload of this algorithm is much lower than that of UMHS because the worst workload of UMHS is large, as shown in Fig. 11. The processing time of UMHS is varied by ET (Fig. 11(a)). If ET occurs in an earlier step, the processing time is extremely short, but if ET does not occur, the processing time is long. In contrast, the CRCS processing time is constant (Fig. 11(b)).

This algorithm is adaptive for the MB-level decomposition



(a) Time chart of UMHS



(b) Time chart of CRCS

Fig. 11. Time chart of Encode of MBs.

because efficiency of the MB-level decomposition is high if the workload assigned with each thread is well balanced. In this algorithm, different from UMHS with ET, the workload of each MB is almost identical. For that reason, the waiting time of synchronization of each MB is small in the MB-level decomposition. Fig. 12 shows a time chart of the encoding of MBs for four processor cores with UMHS and CRCS. The encoding time of a MB with UMHS is occasionally early, but if encoding of other processor cores is delayed, it must wait to encode the delayed MB (Fig. 12(a)). Therefore, the entire encoding efficiency worsens. However, the encoding time of MB with CRCS is almost constant (Fig. 12(b)). Therefore, if the encoding of an MB of CRCS is slower than that of UMHS, there is little waiting overhead. The overall encoding efficiency is good.

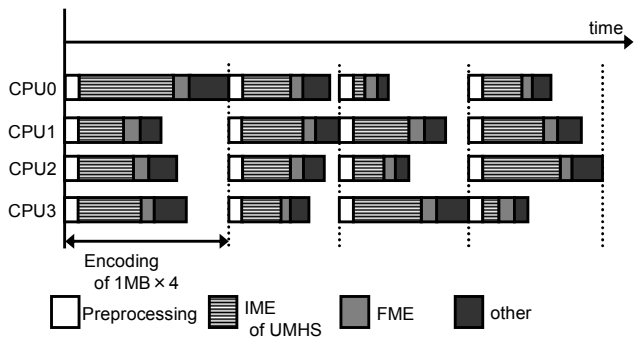
IV. Evaluation Result

To evaluate MB-level decomposition with CRCS, we conducted several evaluations. Our evaluations were performed using the SESC simulator [10]. Table I shows the evaluated configuration of the SESC.

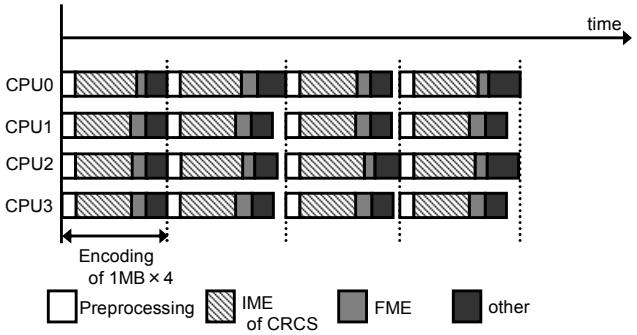
A. Motion estimation evaluation

We evaluated execution of the MB-level decomposition with CRCS and UMHS. Block sizes used for evaluation were 16×16, 16×8, 8×16, 8×8, and 4×4. This block configuration was used for the following evaluations. Table II presents evaluation conditions of the encoding sequence.

Evaluation results for SDTV are portrayed in Fig. 13. The normalized fps is normalized by the fps of UMHS executed by one thread. In MB-level decomposition, using CRCS is faster than UMHS; the memory accesses of CRCS are fewer than those of UMHS. The CRCS increased fps by 1.4 times and decreased memory accesses by 4% compared with UMHS in 16 threads.



(a) Time chart of UMHS



(b) Time chart of CRCS

Fig. 12. Time chart of Encode of MBs on four processor cores.

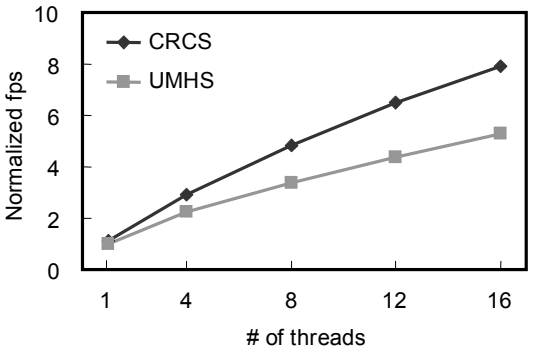
TABLE I
PROCESSOR CONFIGURATION AND SYSTEM PARAMETERS

Multi-core processor architecture	Homogeneous multicore, Out-of-order Alpha like-processor
# of cores	32
issue	4
Frequency	4 GHz
L1 instruction cache	32 KB
L1 data cache	32 KB
L2 shared cache	2048 KB
Memory band width	2.4 GB/s

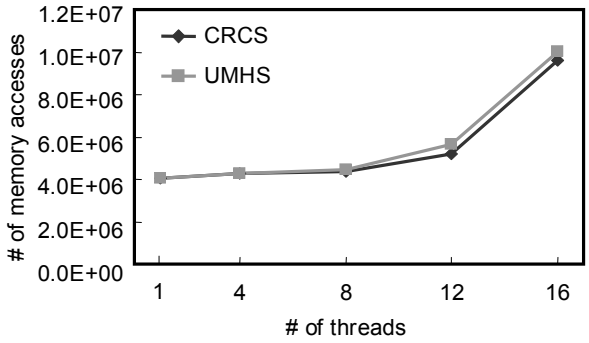
TABLE II
CONDITION FOR MOTION ESTIMATION EVALUATION

Resolution	SDTV, 720 × 480; HDTV, 1920 × 1024
Frame rate	15 fps
# of reference frames	2
(M, N)	(1, 15)
Bitrate	SDTV, 4 Mbps; HDTV, 10 Mbps
# of frames	10
Search range	SDTV, ±64 × ±64; HDTV, ±128 × ±128

Evaluation results for HDTV are shown in Fig. 14. This figure shows the same result as Fig. 13. Here, CRCS increased fps by 1.7 times and decreased memory accesses by 16% compared with UMHS in 16 threads. The search range becomes wide for HDTV; UMHS uses many data. The memory accesses of UMHS are far more numerous than those of CRCS. In MB-level decomposition for SDTV and HDTV, the PSNR of CRCS was 1% or less, on average, than UMHS.

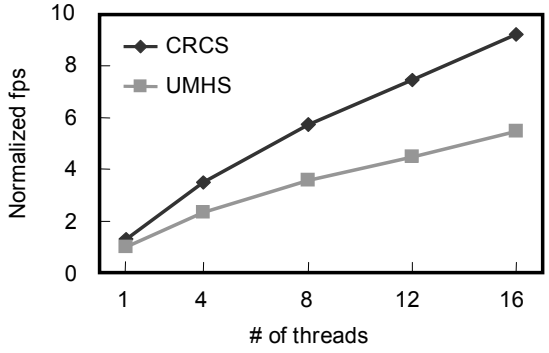


(a) Normalized fps

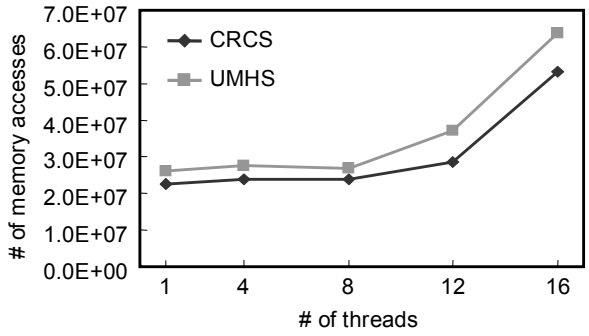


(b) Number of memory accesses

Fig. 13. Evaluation result for SDTV.



(a) Normalized fps



(b) Number of memory accesses

Fig. 14. Evaluation result for HDTV.

B. Parallelization evaluation

We evaluated the execution of frame-level decomposition with UMHS and MB-level decomposition with CRCS. Table III shows the evaluation condition of the encoding sequence.

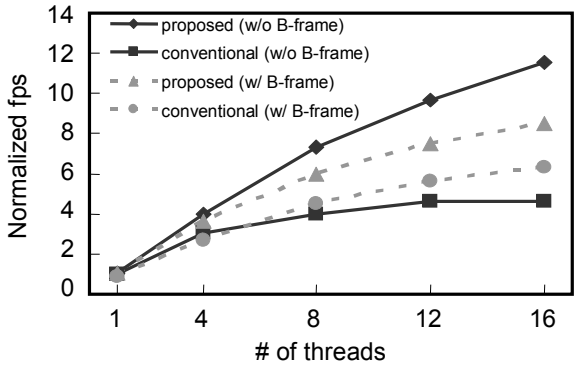
Evaluation results for SDTV without B-frame are summarized in Fig. 15. The proposed method is MB-level decomposition with CRCS; the conventional method is frame-level decomposition with UMHS. As shown in Fig. 15, these displacements are used in the following analyses. Normalized fps is normalized by fps of the conventional method executed by one thread. The proposed method increased fps more than conventional method by 2.5 times in 16 threads. The processing speed used for the conventional method is saturated as the thread number increases (Fig. 15(a)). Therefore, the number of memory accesses used for the conventional method is saturated similarly (Fig. 15(b)). This saturation results from the waiting time that reference area is constructed. The proposed method, however, is not saturated because the synchronization overhead is lower and the number of memory accesses is suppressed though higher fps. In 16 threads, the memory accesses of the proposed method, however, increase considerably because data of reference areas overflowed from the cache. The memory bandwidth of the conventional method is saturated because fps and the memory accesses are saturated. For this reason, the memory bandwidth of the proposed method is greater than that of the conventional method (Fig. 15(c)).

TABLE III
CONDITION FOR PARALLELIZATION EVALUATION

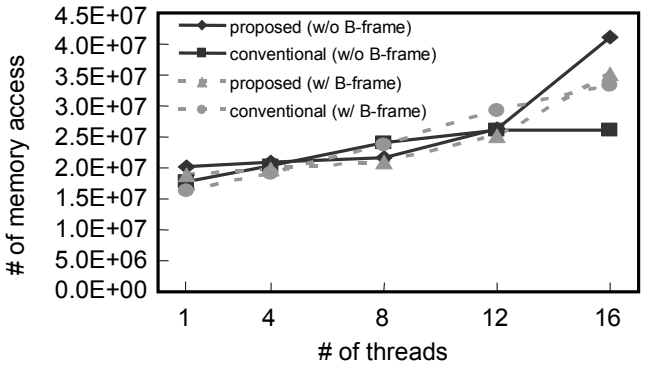
Resolution	SDTV, 720 × 480; HDTV, 1920 × 1024
Frame rate	15 fps
# of reference frames	2
(M, N)	(1, 15) and (2, 15)
Bitrate	SDTV, 4 Mbps; HDTV, 10 Mbps
# of frames	SDTV, 50; HDTV, 30
Search range	SDTV, ±64 × ±64; HDTV, ±128 × ±128

Evaluation results for SDTV with a B-frame are presented in Fig. 15. The proposed method increased fps to more than 1.3 times that provided by the conventional method in 16 threads. Different from the result without B-frame, the conventional method is not saturated in Fig. 15(a) because the processing efficiency of the conventional method increases because of the ability to encode the P-frame and B-frame in parallel efficiently. Since the processing speed is faster compared with the encoding without B-frame, however, the number of memory accesses is extended. The memory accesses under the proposed method are suppressed to the level of that without B-frame. Consequently, the memory bandwidth of the conventional method comes closer to that of the proposed method, but the bandwidth of the proposed method is greater than that of the conventional method because of the higher fps (Fig. 15(c)).

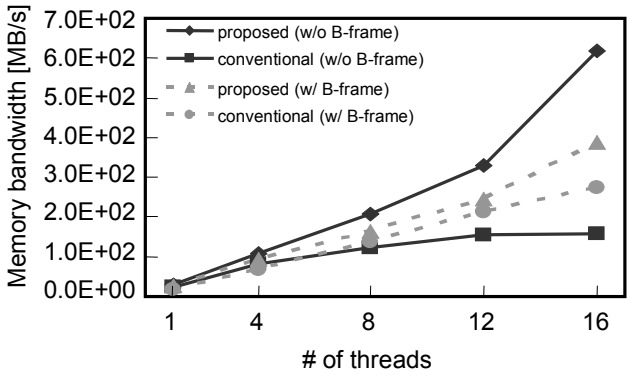
Evaluation results for HDTV without B-frame are shown in Fig. 16. Evaluation was executed using 1, 4, 8, 12, 16, and 32 threads. The normalized fps is normalized by the fps of the conventional method executed by one thread. The proposed



(a) Normalized fps

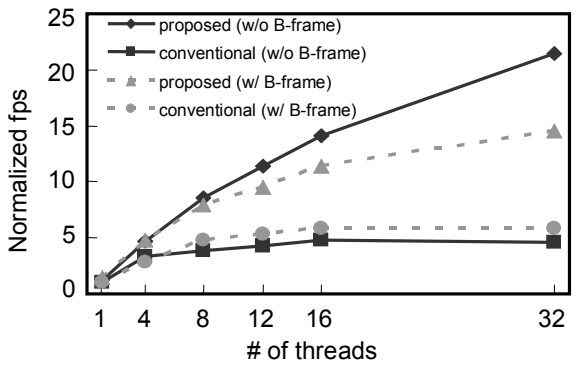


(b) Number of memory accesses

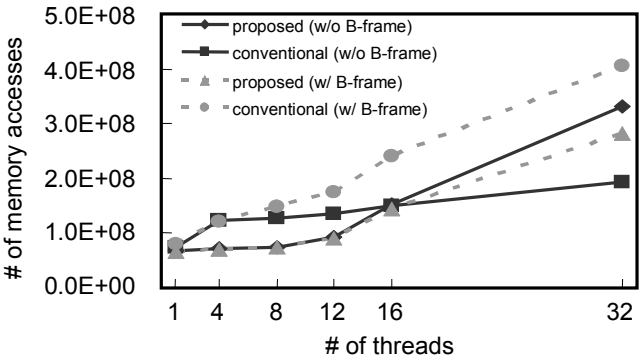


(c) Memory bandwidth

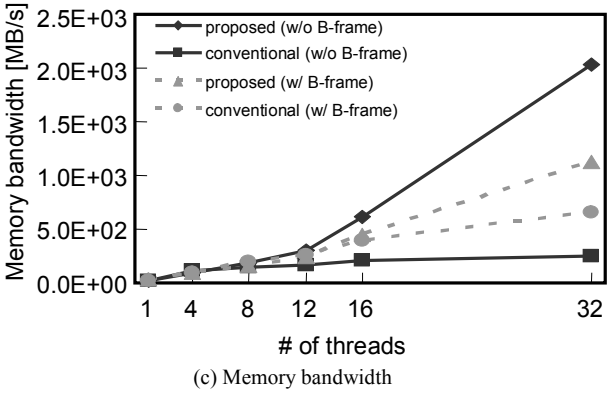
Fig. 15. Evaluation result for SDTV.



(a) Normalized fps



(b) Number of memory accesses



(c) Memory bandwidth

Fig. 16. Evaluation result for HDTV.

method increased fps to 4.6 times that obtained using the conventional method in 32 threads. As shown in Fig. 16(a), the processing speed of the conventional method for HDTV is saturated because the number of threads increases, as was true also for SDTV. Different from SDTV, the number of memory accesses under the conventional method increases considerably compared with the proposed method because the reference area broadens. In the conventional method, reference areas accessed from each thread are different, but in the proposed method, the reference areas accessed from each thread are overlapped. By this overlap of the reference areas, the memory accesses performed using the proposed method is considerably fewer than under the conventional method. In 32 threads, the number of memory accesses of the proposed method increases considerably, however, as was true also for SDTV. This increase results from data of reference areas overflowing from the cache. The memory bandwidth of the conventional method is saturated as it was with SDTV, but the memory bandwidth using the proposed method is greater than that under the conventional method (Fig. 16(c)), but the memory bandwidth of the proposed method is closer to that of the conventional method by suppression of the number of memory accesses.

Evaluation results for HDTV with a B-frame are shown in Fig. 16. The proposed method can increase fps to 2.4 times more than the conventional method in 32 threads. In Fig. 16(a), the processing speed of the conventional method is shown to be as saturated as that encoded without B-frame. The memory

accesses of the conventional method encoded with B-frame, however, are not saturated because the P-frame and B-frame are encoded simultaneously, as was true for SDTV, but the proposed method shows the same tendency as SDTV. Therefore, the memory accesses under the proposed method are fewer than those under the conventional method. In 32 threads, the memory accesses of the proposed method increase considerably, as shown also in the result of evaluation of without B-frame, but that of the conventional method in 32 threads increased similarly. The memory bandwidth of the two methods is nearly equal, but the fps of proposed method is considerably high (Fig. 16(c)).

V. CONCLUSION

The effectiveness of the MB-level decomposition with CRCS was described. The MB-level decomposition is more scalable to the thread number; CRCS is a suitable motion estimation method for MB-level decomposition because the workload of each thread is well balanced in CRCS.

We evaluated encoding of MB-level decomposition with CRCS and this method with UMHS. Results show that this method with CRCS is faster and uses less memory access than UMHS does. For instance, CRCS increased fps by 1.7 times and decreased memory accesses by 16% compared with UMHS in 16 threads for HDTV. Therefore, CRCS is effective for MB-level decomposition. Additionally, we evaluated encoding of MB-level decomposition with CRCS and frame-level decomposition with UMHS. Results show that the proposed method is more scalable than the conventional method because the processing speed of the conventional method is saturated as the thread number increases. The proposed method for HDTV without the B-frame increased fps to 4.6 times as high as that by the conventional method; for HDTV with the B-frame, the proposed method increased fps to 2.4 times that of the conventional method, despite fewer memory accesses under the proposed method. Because of its scalability and fewer memory accesses, MB-level decomposition with CRCS is suitable for use with multi-core computing of the future.

REFERENCES

[1] ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, Draft ITU-T Recommendation and Final Draft Information Standard of Joint Video Specification, 2003.

[2] Y.-K. Chen, X. Tian, S. Ge, and M. Girkar, "Towards Efficient Multi-Level Threading of H.264 Encoder on Intel Hyper-Threading Architectures," Parallel and Distributed Processing Symp., 2004. Proc.. 18th Int'l., pp.063-072.

[3] Y.-K. Chen, E.-Q. Li, X. Zhou, and S. Ge, "Implementation of H.264 Encoder and Decoder on Personal Computers," Journal of Visual Communication and Image Representation, 2006, pp. 509–532.

[4] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," IEEE Trans. Circuits Syst. Video Technol., vol.16, no.6, pp.128-129, Feb. 2005.

[5] Y. Murachi, J. Miyakoshi, M. Hamamoto, T. Iinuma, T. Ishihara, F. Yin, J. Lee, H. Kawaguchi, and M. Yoshimoto, "A Sub 100 mW H.264 MP@L4.1 Integer-Pel Motion Estimation Processor Core for MBAFF Encoding with Reconfigurable Ring-Connected Systolic Array and Segmentation-Free, Rectangle-Access Search-Window Buffer," IEEE Trans. Electron., April 2008.

[6] M. Roitzsch, "Slice-Balancing H.264 Video Encoding for Improved Scalability of Multicore Decoding," in WorkinProgress Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS), 2006.

[7] x264, http://www.videolan.org/developers/x264.html

[8] E. B. Van Der Tol, E. G. T. Jaspers, and R. H. Gelderblom, "Mapping of h.264 decoding on a multiprocessor architecture," Image and Video Communications and Processing, May 2003.

[9] ISO/IEC | ITU-T VCEG, Fast Integer Pel and Fractional Pel Motion Estimation for JVT, JVT-F017, 2002

[10] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, K. Strauss, S. Sarangi, P. Sack, and P. Montesinos, "SESC Simulator," January 2005. http://sesc.sourceforge.net.