

A Highly Efficient Parallel Algorithm for H.264 Encoder Based on Macro-Block Region Partition

Shuwei Sun, Dong Wang, and Shuming Chen

Computer School, National University of Defense Technology, Changsha, China
shuwei97@sina.com, nudtjum@163.com, smchen@nudt.edu.cn

Abstract. This paper proposes a highly efficient MBRP parallel algorithm for H.264 encoder, which is based on the analysis of data dependencies in H.264 encoder. In the algorithm, the video frames are partitioned into several MB regions, each of which consists of several adjoining columns of macro-blocks (MB), which could be encoded by one processor of a multi-processor system. While starting up the encoding process, the wave-front technique is adopted, and the processors begin encoding process orderly. In the MBRP parallel algorithm, the quantity of data that needs to be exchanged between processors is small, and the loads in different processors are balanced. The algorithm could efficiently encode the video sequence without any influence on the compression ratio. Simulation results show that the proposed MBRP parallel algorithm can achieve higher speedups compared to previous approaches, and the encoding quality is the same as JM 10.2.

1 Introduction

Compared with previous standards, H.264 developed by the JVT(Joint Video Team formed by ISO MPEG and ITU-T VCEG) achieves up to 50% improvement in bit-rate efficiency and more than 4 times of the computational complexity [1], due to many new features including quarter-pixel motion estimation (ME) with variable block sizes and multiple reference frames (up to 16), intra-prediction, integer transformation based on discrete cosine transform (DCT), alternative entropy coding mode Context-based Adaptive Variable Length Coding (CAVLC) or Context-Based Adaptive Binary Arithmetic Coding (CABAC), in-loop de-blocking filter and so on [2]. Therefore, the parallel structure and parallel algorithm are an alternative ways for real-time H.264 video application.

Multi-processor and multi-threading encoding system and parallel algorithms have been discussed in many papers [3][4][5][6]. In [3], the H.264 encoding algorithm is mapped onto multi-processor DSP chips, C3400 MDSP of CRADLE, in which, computing subsystem of 2 RISC core and 4 DSP are used to implement the baseline encoder, forming three pipeline stages, implementing a CIF resolution H.264 codec. However, as a traditional parallel mechanism, task pipelining is not appropriate for H.264 codec due to two reasons: 1) large amount of data need transferring between processors, which demand a lot for the system bandwidth; 2) functions in the H.264 codec have different computing complexity, and it is hard to map the algorithms among processors uniformly, so the final performance is always restricted by the processor with the heaviest load.

Because there are a lot of data parallelisms in the video codec, many parallel algorithms are proposed exploiting data parallelism [4][5]. Y. K. Chen, et al. give the parallel algorithms of H.264 encoder based on Intel Hyper-Threading architecture [4], they split a frame into several slices, which are processed by multiple threads, resulting speedups ranging from 3.74 to 4.53 on a system of 4 Intel Xeon processors with Hyper-Threading technique. However, this brings additional overheads on bit-rates: splitting frames into slices increases the bits for information of slice header; macro-blocks (MB) between slices do not need ME and MC, which increases the bits for the transformed coefficients.

Z. Zhao and P. Liang propose the parallel algorithms with wave-front technique based on the analysis of the data dependencies in the H.264 baseline encoder [5], data are mapped onto different processors at the granularity of frames or MB rows, and the final speedups are over 3.0 on software simulator with 4 processors. This method of data partition with the wave-front technique avoids the damage on compression ratio by splitting frames into slices. However, in the motion estimation of H.264 encoder, the search center is the predicted motion vector (PMV), which leads to the data dependencies introduced by ME are not confined by the search range and have the property of uncertainty, however, the analysis in [5] is unsatisfactory, in fact, their method confines the search center at the position of (0,0).

In this paper, we present a method of data partition based on the analysis of the data dependencies in the H.264 encoder, in which, frames are split into several MB regions, each of which includes several adjoining columns of MBs and is mapped onto a different processor, while starting up the encoding process, the wave-front technique is adopted. At last, we give a new efficient parallel algorithm for H.264 encoder based on the MB region partition, i.e. MBRP parallel algorithm.

The rest of the paper is organized as follows: Section 2 provides an overview of the data dependencies in H.264 encoder and the homologous limitative factor to exploit data parallelism. Section 3 details our data partition method based on the MB region and our MBRP parallel algorithm. Section 4 provides the simulation results and Section 5 concludes our work and gives the future work.

2 Data Dependencies in H.264

In the H.264 encoder, a MB is composed of 16×16 luma pixels and $8 \times 8 \times 2$ chroma pixels. The reference software JM 10.2 provided by JVT processes each MB in sequence [7], which results in several types of data dependencies that should be avoided in parallel algorithm.

2.1 Data Dependencies Introduced by Inter-prediction

In inter-prediction, the PMV defines the search center of ME, which comes from the motion vectors (MV) of the neighboring sub-blocks, MV_A , MV_B , MV_C , and the corresponding reference indexes [7], as shown in Fig.1 (a). Only the difference between the final optimal MV and the PMV will be encoded. Accordingly, the ME processes of the left, top, and top-right neighboring MBs should be finished before encoding the current MB. Besides, data of the reconstructed pixels in the search

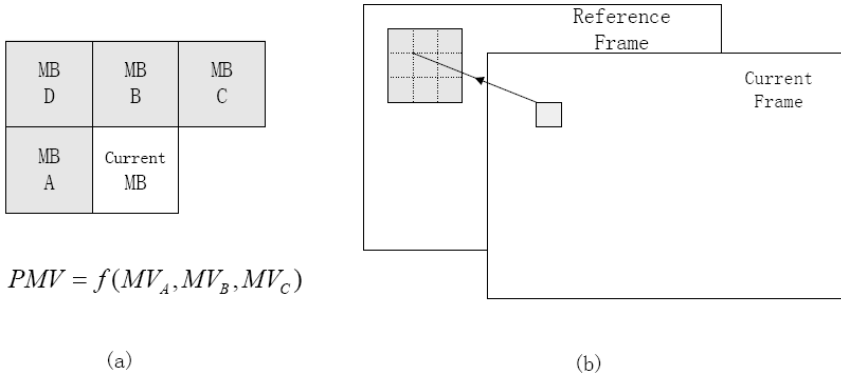


Fig. 1. Data dependencies introduced by inter-prediction

window should be available at that time, if the search range is 16, at least the reconstructed pixels of 9 MBs should be available, as shown in Fig.1 (b).

What should be noticed is, in H.264 encoder the search center of ME is the PMV, which results in that the data dependencies introduced by ME are not confined by the search range and have the property of uncertainty, especially when the RDO is used (there is no restriction for the search center), the final optimal MV may go far beyond the search range. We have collected the final MV of each MB using the JM10.2 software, and the evaluation is carried out on the standard video sequence "Foreman"(CIF, 300 frames), and the main encoder parameters are shown in Table 1. The results show that the biggest MV value is 61 pixels (in horizontal axis), which means that the final MV may be as four times long as the search range. The value may be bigger for sequence with high amount of movement. If RDO is not used, the search center is restricted to be smaller than search range; therefore the final MV could be as two times long as the search range at most.

Table 1. Encoder parameters used for evaluation

Search range	16	QP	28
ME algorithm	FS	Hadamard transform	Used
Number of reference frames	1	RDO mode decision	Used
Sequence type	IPPP	Entropy coding method	CAVLC

2.2 Data Dependencies Introduced by Intra-prediction and In-Loop Filter

In the intra-prediction, the reconstructed pixels of the neighboring MBs in the same frame should be available before encoding the current MB, as shown in Fig. 2 (a), in which the white field figures the current MB and the grey field figure the reconstructed pixels, which are distributed in the left, top-left, top and top-right neighboring MBs. What should be noticed is that these reconstructed data are not the same as those used in the inter-prediction; the latter are produced after the former are in-loop filtered.

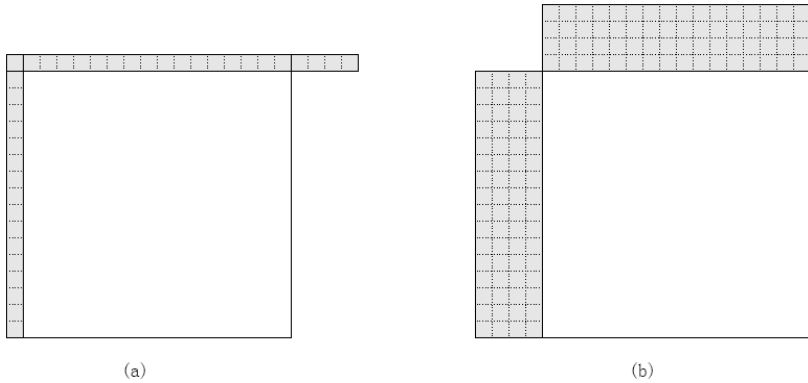


Fig. 2. Data dependencies introduced by intra-prediction and in-loop filter

In the process of in-loop filtering the current MB, the reconstructed and half-filtered data of the neighboring MB should be available, as shown in Fig. 2 (b), in which, the white field figures the current MB and the grey field figures the reconstructed and half-filtered pixels, which are distributed in the left and top neighboring MBs. And we must know that these data in the neighboring MBs are not full-filtered.

2.3 Data Dependencies Introduced by CAVLC

CAVLC is the entropy-coding tool in the baseline of H.264 encoder. In the process, the total number of non-zone transformed coefficients and the number of tailing ones are coded firstly, and there are 4 look-up tables to choose from according to the number of none-zero transformed coefficients of the left and top neighboring MB, which means the number of none-zero transformed coefficients of the left and top neighboring MB must be available before coding the current MB.

2.4 Data Dependencies Summary

As analyzed above, there are 3 types of data dependencies: 1) data dependencies between frames, which means MB could not be processed until the reconstructed and filtered pixels needed for inter-prediction are available; 2) data dependencies between MB rows in the same frame, which means MB could not be processed until the three neighboring MBs above are encoded and reconstructed; 3) data dependencies in the same MB row, which means MB could not be processed until the left neighboring MB is encoded and reconstructed. The three types of data dependencies must be avoided to exploit the data parallelisms in the H.264 encoder.

3 MBRP Parallel Algorithm

Firstly, we partition a frame into several MB regions, as shown in Fig.3, in which each little square stands for a MB, and each MB region comprises several adjoining

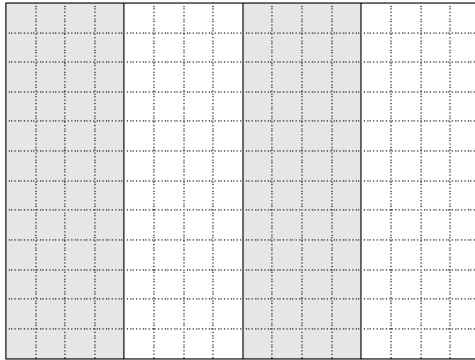


Fig. 3. Partition for MB region

columns of MBs; then, these MB regions are mapped onto different processors, and the data is exchanged appropriately according to the three types of data dependencies; at last, we propose our MBRP parallel algorithm with the wave-front technique.

3.1 The MB Region Partition

In order to achieve high performance, we must try to minimize the amount of data that needs to be exchanged between processors. In H.264 encoder, the data must be exchanged are those which are involved in the data dependencies but encoded by different processors.

Firstly, the data dependencies between frames are uncertainty because of the uncertainty of the search center in inter-prediction, and all the position in the reference frame may be the candidate to be checked in theory. Thereby, the reconstructed and filtered data of a MB region must be transferred to all the other processors as the reference data, no matter how the data is partitioned.

As for the data dependencies between MB rows in the same frame, because the three neighboring MB above are all involved, it seems that no matter how to partition the data, a processor always needs to exchange data with the processors which encode the neighboring MB regions. However, considering the data dependencies in the same MB row, if a MB region comprises only one column of MBs, the neighboring MB regions could not be processed simultaneously, which is opposite to our objective, thereby each MB region should comprise two columns of MBs at least.

On the other hand, when we implement the parallel H.264 encoder, the number of the processors in the multi-processor system is another important factor to determine the MB region partition. If the processors number is not too large (smaller than half of the number of MB columns of a frame), the number of MB columns each MB region comprises should be equal or close as much as possible, so that the load of each processor could be balanced; if the processor number is larger than half of the number of MB columns of a frame, each MB region should comprise two columns of MBs, and we should combine our MBRP algorithm with some other parallel algorithms, e. g., methods in [4] or [5], to improve the performance further.

3.2 The Data Exchanging Between Processors

Now, we analyze the necessary data exchanging between processors due to the three types of data dependencies and the data partition based on MB region. There are 5 types of data that need exchanging: 1) reconstructed and unfiltered data, which are used in the intra-prediction; 2) MVs of the MBs or sub-macro-block, which are used in the inter-prediction; 3) entropy coding information of MBs, which are used in the entropy coding for neighboring MBs; 4) reconstructed and half-filtered data, which are used in in-loop filter for the neighboring MB; 5) reconstructed and filtered data, which are used in the inter-prediction. According to the analysis of data dependencies and the description of the data partition, it is clear that the types of exchanging-data and the quantity of data that need to be exchanged are different, therefore, we must arrange the time and manner to exchange these data appropriately.

3.3 The Wave-Front Technique

After the data partition, we can map the MB regions to different processors. Considering the data dependencies in the same MB row, MBs in a MB row must be encoded in sequence, therefore we adopt the wave-front technique, starting up the encoding process of processors orderly to encode MBs in the same MB row but different regions. After adopting the wave-front technique, processors start to encode data after a short time one by one, and during the time a processor could encode a row of MBs in a MB region and transfer required reconstructed data to the next adjoining processor, thus avoiding the data dependencies in the same MB row, and processors could encode MBs in different MB rows of respective MB regions synchronously, as shown in Fig.4, squares with single diagonal denote the encoded MBs while squares with chiasm diagonal stand for the MBs being encoded by processors synchronously.

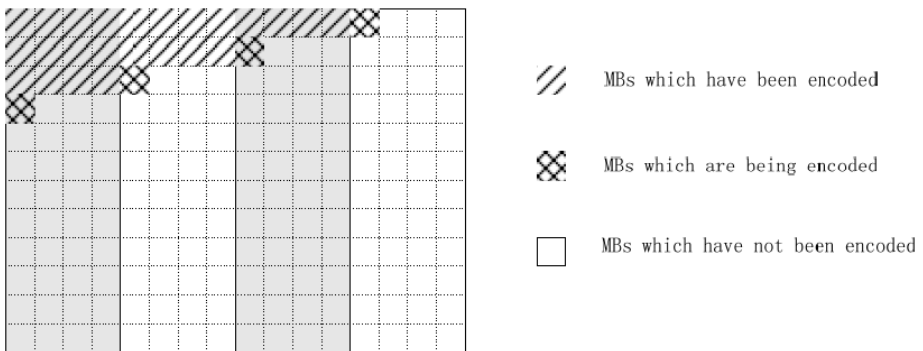


Fig. 4. Wave-front technique based on MB region partition

3.4 The Parallel Algorithm Based on the MB Region Partition

At last, we present our MBRP parallel algorithm for H.264 encoder: Firstly, we determine the MB region partition according to the encoder parameters and the processor number of the multi-processor system, insuring that each MB region

comprises equal or close number of columns of MBs. Then, we map the MB regions onto different processors to be encoded. While starting up the encoding process, we adopt the wave-front technique, and the processors begin to encode data orderly after a short time to avoid the data dependencies in the same MB row. Each processor encodes a MB region in the order of MB rows, after encoding the boundary MB (the first and the last columns of MB in MB region), required reconstructed and unfiltered data, the MVs and the number of non-zero transformed coefficients of the boundary MB are transferred to the neighboring processors (which encode the neighboring MB regions). After the in-loop filtering of the MB region, the half-filtered data are transferred to the next neighboring processor (which encodes the right neighboring MB region), and the filtered data are broadcasted to all the other processors. When the required data are available, all the processors work synchronously, encoding different MB regions of video frames respectively.

4 Simulation Results

The simulator of our MBRP parallel algorithm for H.264 encoder is developed using C language and implemented on a PC with a P4 1.7GHz processor and a 512MB memory. The simulation results are compared with those from JM10.2, which is a sequential encoding structure. In our software simulation of H.264 encoder, four processors are simulated. The main encoder parameters are shown in Table 1. The simulator collects the maximal encoding time among every 4 concurrently processed MB regions and the corresponding time spent on data exchanging. Some of the simulation results are presented in Table.2 and Table.3. For Table.2, we used "Foreman" (CIF) as video source, and 300 frames were encoded. And for Table.3, we used "Foreman" (SDTV) as video source, and 300 frames were encoded. Experiments show that the speedups higher than 3.3 are achieved and the encoding quality is the same as JM10.2.

Table 2. Simulation results for "Foreman" (CIF)

	Encoder time /frame	PSNR (Y)	PSNR (U)	PSUN (V)	Bits/frame	Speedup
JM10.2	5.5656 s	36.12	40.48	42.02	15295.87 bits	1
MBRP Algorithm	1.67045 s	36.12	40.48	42.02	15295.87 bits	3.33

Table 3. Simulation results for "Foreman" (SDTV)

	Encoder time /frame	PSNR (Y)	PSNR (U)	PSUN (V)	Bits/frame	Speedup
JM10.2	22.343 s	37.84	42.67	44.14	58437 bits	1
MBRP Algorithm	6.73535 s	37.84	42.67	44.14	58437 bits	3.32

In figure 5, we give the relationship between the speedup and the number of processors in multi-processor system, and we used the "Foreman" (CIF) and the "Foreman" (SDTV) as the video source. As seen in Fig.5, when the number of processors is small, the compressing performance improves linearly approximately as the number increases; when the numbers are of some values, the speedups keep unchanged, this is because the loads of different processors are imbalanced, and the most heavy load is the same for these number values; when the numbers reach certain values, 11 and 23 for "Foreman" (CIF) and "Foreman" (SDTV) respectively, the speedups get the highest value.

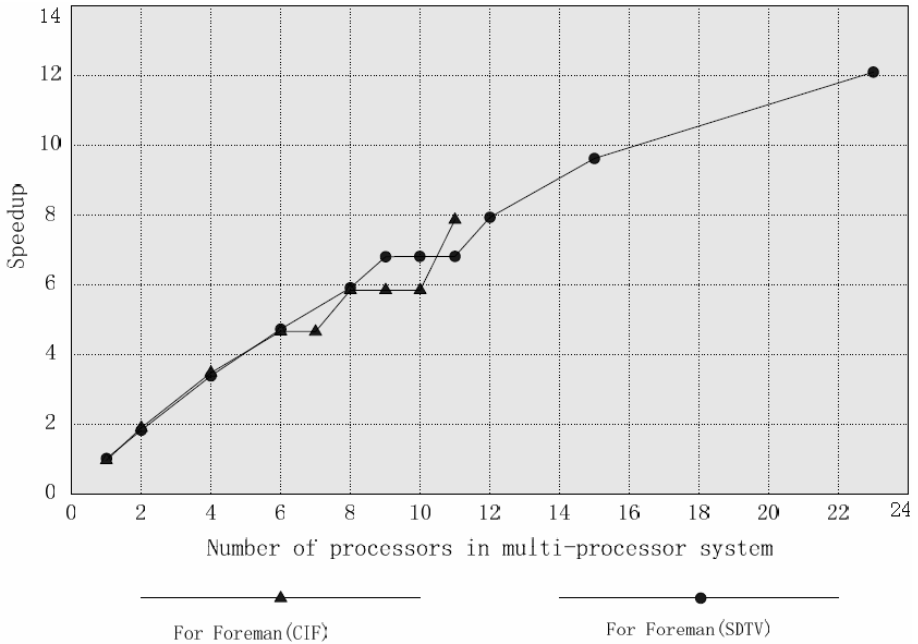


Fig. 5. The relationship between the speedup and the processor number

In table 4, we give the comparison between different parallel algorithms of H.264 encoder. Compared to other parallel algorithms, MBRP algorithm achieves higher speedup, there are 2 reasons: 1) as a MB row is partitioned into different part and mapped onto different processors, the parallel granularity in MBRP algorithm is smaller, which means the loads between processors could be balanced more easily; 2) each MB region comprises several adjoining columns of MBs, which means the amount of data that need to be exchanged between processors are smaller. On the other hand, since each MB region should comprise 2 columns of MBs at least, the highest speedup our MBRP parallel algorithm could achieve is restricted by the resolution of the video source. However, our algorithm could be combined with some other parallel algorithms easily to improve the performance further.

Table 4. Comparison between different parallel algorithms

Parallel Algorithm	Parallel Granularity	Number of Processors	Compression Ratio degradation	Speedup
Algorithm in [4]	Frame and Slice	4 (8 logic processors)	Yes	3.74~4.53
Algorithm in [5]	Frame and MB Row of frame	4	No	3.1
MBRP Algorithm	MB row of MB region	4	No	3.3

5 Conclusions and Future Work

We propose the MBRP parallel algorithm for H.264 encoder in this paper, in which frames are split into several MB regions; each MB region includes several adjoining columns of MBs and is mapped onto a different processor to be encoded; with the wave-front technique, data in different MB regions could be encoded synchronously. The simulation results show that the MBRP parallel algorithm is quite efficient, and the compressing performance improves linearly approximately as the number of the processors in the multi-processor system increases.

Future work includes the implementation of the algorithm on a multi-core SoC and the investigation on parallel algorithm of H.264 encoder with CABAC entropy coding mode.

References

1. Chen, T.C., Huang, Y.W., Chen, L.G.: Analysis and design of macro-block pipelining for H.264/avc VLSI architecture. In: Proceedings of the 2004 International Symposium on Circuits and Systems (2004)
2. JVT Draft recommendation and final draft international standard of joint video specification. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC (2003)
3. Gulati, A., Campbell, G.: efficient mapping of the H.264 encoding algorithm onto multi-processor DSPs. In: Proceedings of SPIE-IS&T Electronic Imaging (2005)
4. Chen, Y-K., Ge, S., T. X., G.M.: towards efficient multi-level threading of H.264 encoder on intel hyper-threading architectures. In: 18th International Parallel and Distributed Processing Symposium (2004)
5. Zhao, Z., Liang, P.: A Highly Efficient Parallel Algorithm for H.264 Video Encoder. In: 31st IEEE International Conference on Acoustics, Speech, and Signal Processing (2006)
6. Jacobs, T.R., Chouliaras, V.A., Nunez-Yanez, J.L.: A Thread and Data-Parallel MPEG-4 Video Encoder for a SoC Multiprocessor. In: Proceedings of 16th International Conference on Application-Specific Systems, Architecture and Processors (2005)
7. JM10.2, <http://bs.hhi.de/suehring/tml/download/jm10.2.zip>