International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:4, No:12, 2010

Dynamic Data Partition Algorithm for a Parallel H.264 Encoder

Juntae Kim, Jaeyoung Park, Kyoungkun Lee, and Jong Tae Kim

***Abstract*—**The H.264/AVC standard is a highly efficient video codec providing high-quality videos at low bit-rates. As employing advanced techniques, the computational complexity has been increased. The complexity brings about the major problem in the implementation of a real-time encoder and decoder. Parallelism is the one of approaches which can be implemented by multi-core system. We analyze macroblock-level parallelism which ensures the same bit rate with high concurrency of processors. In order to reduce the encoding time, dynamic data partition based on macroblock region is proposed. The data partition has the advantages in load balancing and data communication overhead. Using the data partition, the encoder obtains more than 3.59x speed-up on a four-processor system. This work can be applied to other multimedia processing applications.

***Keywords*—**H.264/AVC, video coding, thread-level parallelism, OpenMP, multimedia

I. Introduction

THE H.264/AVC standard is developed by Joint Video Team (JVT) for video compression. The bit-rate efficiency of a H.264 encoder has been improved in variable block sizes, multiple reference frames, CAVLC, CABAC, and more. However, the complexity of the encoder is increased 10 times and it entails power consumption issues [1]. For a real time encoder, the encoding speed is not enough with a single core processor. Hence, hardware acceleration or a parallel software algorithm is needed for increasing the speed of the encoder.

Parallel algorithms have been discussed in several papers [2]-[5]. There are frame-level, slice-level, and macroblock-level parallel algorithms. Especially, macroblock-level parallelism which are adopted in [6]–[8] satisfies capabilities of parallelism with no video quality losses. The interests of macroblock level parallelism are the communication overhead between processors and load balancing. Data communication is needed where one macroblock encoded by a processor has the neighboring macroblocks which are performed by other processors. Load balancing is to assign macroblocks onto processors similarly. Processors should wait until dependencies are avoided except the first processor which encodes a macroblock first among processors.

The parallel algorithm with wave-front technique splits a frame into macroblocks and mapped onto different processors by horizontal axis [6]. The load balancing for the technique is well because it takes 6 TMB (where TMB stands for the time to encode one macroblock) to start processors in parallel. However, the data communication is needed at every encoding process expect outer blocks of a frame so that the data exchanging slows down the process. The simulation results are 3.08x speed-up in QCIF format and 3.17x speed-up in CIF video formats.

The MBRP parallel algorithm which adopts wavefront technique is focused on reducing the data communication between processors by data partition [7]. The method of data partition is to assign a macroblock region for each processor so that neighboring macroblocks are mostly handled by the same processor. However, the waiting time of processors until they start to encode a macroblock in parallel is mostly longer than the waiting time of [6]. The simulation results are 3.32x in CIF format and 3.33x speed-ups in SD video formats.

The research of macroblock-level parallelism is also discussed in [8]. The data partition of the research is that the macroblocks when they can avoid dependencies starts to encode in a frame and over frames. Encoding over frames is only initiated when reconstructed macroblocks are more than half of a frame. Hence, it can increase the concurrency of the thread-level parallelism to process multiple frames. Some improved techniques are used in the implementation such as motion estimation and mode decision. The implementation results show 3.8x speed-ups for CIF, SD, and HD video formats.

In this paper, a method of data partition for macroblock-level parallelism is presented in order to reduce data communication overhead and to improve concurrency. The objective of this research is to maximize the efficiency of parallelization by dynamic data partition to choose a macroblock where several macroblocks can be encoded. The implementation is limited in intra-prediction. Inter-prediction implementation is remained for future works.

This paper is organized as follows. Section 2 provides the dependencies in intra-prediction and several data partition schemes. Section 3 describes multi-threading scheme and dynamic data partition based on macroblock region. Section 4, software simulation is provided. Section 5 is the conclusion.

Jun Tae Kim is with the Department of Mobile Systems Engineering, Sungkyunkwan University, Suwon, Gyeonggi-do 440-746, Republic of Korea (phone: +82-31-290-7173; fax: +82-31-299-4613; e-mail: kjtjjang@ skku.edu).

Jaeyoung Park and Kyungkun Lee are with the Department of Electrical and Computer Engineering, Sungkyunkwan University (e-mail: jyp8389@gmail.com, ethanleekk@skku.edu).

Jong Tae Kim is with the Department of Mobile Systems Engineering and the Department of Electrical and Computer Engineering, Sungkyunkwan University (e-mail: jtkim@ skku.edu).

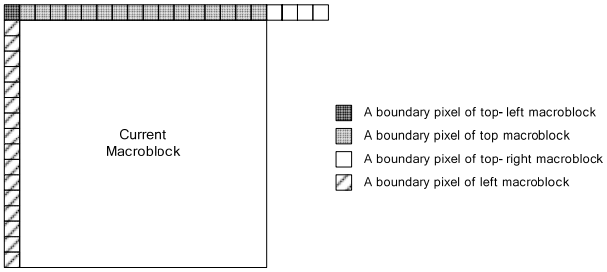

Fig. 1 Data Dependencies in Intra-prediction

II. Data Dependencies in Intra-prediction and Data Partition

A. Data Dependencies in Intra-prediction

In parallel encoding process, data dependencies constrain the order of macroblocks. The process of a H.264 encoder in intra-prediction consists of prediction, CAVLC, and output bitstream. A macroblock to be encoded needs the reconstructed pixels which are neighboring macroblocks as shown in Fig. 1. Therefore, left, top-left, top and top-right neighboring macroblocks are needed for the parallel encoding process. The dependencies of CAVLC are the non-zero coefficients of the left and top neighboring macroblocks. The output bitstream is frequently accessed by all processors so that it should be guaranteed that only one processor accesses at a time.

B. Data Partition

Data Partition is to assign maroblocks onto different processors. Data partition is important because the way to partition macroblocks determines the concurrency. There are several types of data partition for macroblock-level parallelism. Three types of data partition which are dynamic, horizontal, and vertical data partition are compared in terms of the data communication overhead and the processor waiting time until all processors run simultaneously. The waiting time represents the last processor waiting time which is delayed by other processor. The three data partitions are shown in Fig 2. T stands for TMB which is mentioned in introduction.

Dynamic data partition determines each processor how to select a macroblock. The partition cannot guarantee the amount of the data communication because processors are uncertain which macroblocks they will encode. An example of the partition for quad-core system is shown in Fig 2 (a). Mostly, macroblocks needs 37 pixels from neighboring macroblocks which is the maximum amount of data communication. The last processor should wait for 6 TMB and 12 macroblocks can be encoded during the time.

Horizontal data partition is similar to the data partition used in [6]. The marcoblocks in the same row are mapped onto the same processor as shown in Fig 2 (b). This data partition makes the data communication smaller than dynamic data partition. There are 21 pixels to communicate from up, up-left and up-right neighboring macroblocks. The waiting time is 6 TMB as the same as dynamic data partition, however, TMB is shorter then dynamic data partition.

Vertical data partition is based on the MBRP algorithm [7]. Macroblocks are assigned to particular processors as shown in Fig 2 (c). Processors only encode in their own macroblock regions. In the data partition, data communication overhead is minimized because the communication is needed only at the boundaries between macroblock regions. At the right macroblocks of the boundaries, the data communication is 17 pixels from left and up-left neighboring macroblocks. At the left macroblocks of the boundaries, 4 pixels from up-right macroblocks are communicated. In the aspect of the waiting time, processor 4 starts at 10 TMB so that the processor should wait during 9 TMB.

Overall, dynamic data partition needs 32 TMB which is the total encoding time including the data communication overhead. The concurrency of processors is the highest of three partitions, for the reason, the waiting time caused by the dependencies is shorter. On the other hand, data communication is uncertain for

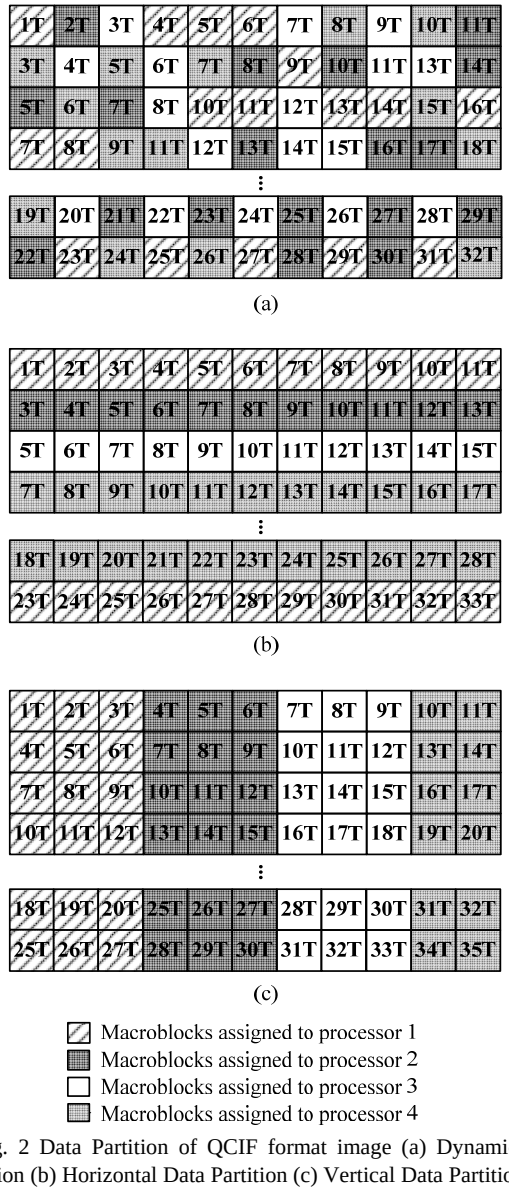

Fig. 2 Data Partition of QCIF format image (a) Dynamic Data Partition (b) Horizontal Data Partition (c) Vertical Data Partition

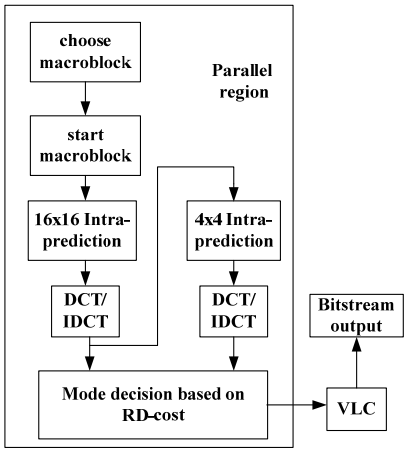

Fig. 3 Multi-thread scheme of a H.264 encoder

situations. Improved data communication overhead, horizontal data partition has smaller data communication and the total encoding time is 33 TMB longer than dynamic data partition. Vertical data partition features the smallest data communication overhead and the longest encoding time of the three data partition.

III. Multi-threading Implementation of a H.264 Encoder and Dynamic Data partition

A. Multi-threading Implementation of a H.264 Encoder

A H.264 Encoder using rate-distortion optimization consists of intra-prediction, DCT/IDCT, mode decision, CAVLC and bitstream output. The multi-threading implementation scheme is shown in Fig. 3. In the reference code [9], the function of bitstream output includes a buffer which is necessary to access exclusively. The bitstream output is called by many functions so that the output should be separated from parallel regions.

The function of "choose macroblock" is for dynamic data partition and scheduling processors. In "choose macroblock" function, all processors run simultaneously to choose a macro-block. Different from the sequential code, macroblocks which are obeying the dependencies can be encoded.

B. Dynamic Data Partition Based on macroblock region

Dynamic data partition is aimed for the improvement of load balancing and reducing data communication overhead. For the load balancing, all processors should be well-assigned by their conditions. Each macroblock spends different time according to its complexity. Considering the software implementation of a H.264 encoder, the main thread takes more burdens by operating system so that these conditions should be in consideration. The horizontal and vertical data partitions have fixed regions which are in charge of their first-assigned processors. In aspect of data communication overhead, static data partition has advantages of communication quantity where neighboring macroblocks are encoded by the same processor. Therefore, the mixed data partition, which takes advantages of the static and the dynamic data partitions, is proposed.

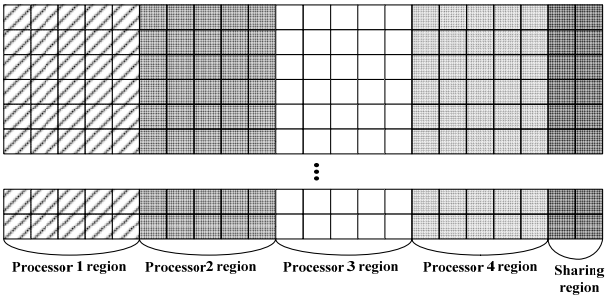

Fig. 4 Data partition for dynamic data partition based on macroblock region

Before partition a frame, there are some video formats which have not the same number of macroblocks in column as a multiple of the number of processors such as CIF format. In this case, the partition has 5 regions shown in Fig 4. Processors have the same number of macroblocks in their regions for load balancing. Sharing region can be encoded by any processors. If the number of macroblocks in column is the same as a multiple of the number of processors, the partition has 4 regions without a sharing region.

The scheduling algorithm for processors is shown in Fig. 5. Following processor region, processors have their own regions. There are 3 conditions to choose a macroblock to encode. The conditions represent priorities considering the performance as shown in Fig 5. The first condition has the highest priority; the second condition has the second priority and others. Processor-n can be processor 1 to 4.

Firstly, macroblocks are available in each region except sharing region as shown in Fig 6 (a). Processors encode macroblocks in their own region. Second, a macroblock in sharing region is available as shown in Fig 6 (b). A macroblock is encoded by any processor which has no macroblock to encode in its own region. Finally, multiple macroblocks are available in a same region row as shown in Fig 6 (c). In this case, the available maroblocks exist over rows in a region. The processor, which owns the region, encodes the macroblock in the upper row, mentioned as k in Fig 5, and other chosen processor encodes the one in the lower row, mentioned as k+1 in Fig 5. Processors should wait if there is no satisfying condition.

```
while ( there are macroblocks to encode )
{
  if ( a macroblock is available in processor-n region)
            assign a maroblock in row k to processor-n

  else if ( a maroclocks is available in sharing region )
       assign a maroblock in sharing region to processor-n

  else if ( macroblocks are available in other processor region )
       assign a macroblock in row k+1 to other processor

  else          //there is no macroblock available
       continue;
}
```

Fig. 5 Pseudo code of scheduling for processors

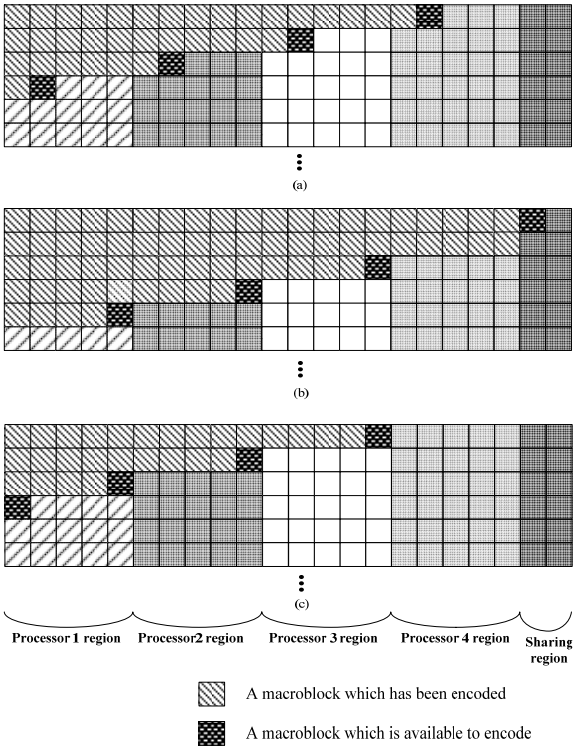

Fig. 6 various cases of parallel encoding process (a) macroblocks are available in their own region (b) a macroblock in sharing region is available (c) multiple macroblocks are available in a region

IV. Software Simulation Results

The software implementation of a H.264 encoder is developed by C language. The quad-core system has four 2.33GHz Intel Pentium 4 processor and a 4GB memory. The reference software is JM 11.0 [9] which is sequential encoding structure developed by JVT. The simulation is performed with rate-distortion based mode decision, 28 quantization parameter and intra-prediction. A variety of test images are used from CIF format to HD format and 300 frames of the images are encoded.

Fig. 7 shows the speed-ups where the reference software is set to 1.0. The speed-ups are measured in the total encoding time. The parallel algorithm is the one with dynamic data partition based on maroblock. The simulation results show that the speed-ups with the parallel algorithm are 3.59x for CIF format, 3.88x for 4CIF format and 3.89x for HD foremat.

V. Conclusion

Parallelism in a H.264 encoder has been researched by the software and the hardware implementation. The performance of parallelism depends on load balancing and data communication overload. To balance the workload, macroblocks which obey the dependencies should be scheduled. The dynamic data partition has high concurrency even though the communication overhead is uncertain. Assigning processors based on data partition, communication overhead can be reduced. The MBRP algorithm [7] shows the way to reduce the data communication overhead. Taking the advantages of the data partitions, the dynamic data partition based on maroblock region is proposed.

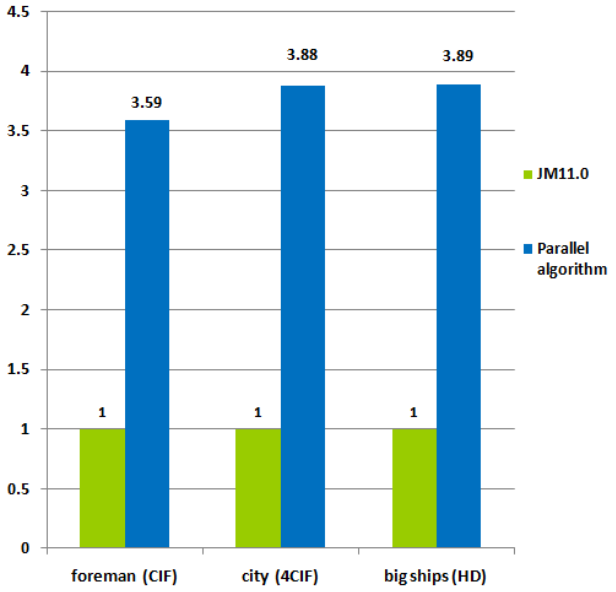

Fig. 7 Speed-up for several types of video image

Dynamic data partition based on macroblock region partition a frame with regions. The number of regions is the same as the number of processors, and sharing region is optional. The data partition has priorities which macroblocks are assigned to a processor. For the load balance, processors can encode macroblocks over the regions with the lowest priority. The data partition is implemented in software and simulated. The results are more than 3.59x speed-ups for several formats of images.

References

[1] C. Luo, J. Sun, and Z. Tao, "The research of H.264/AVC video encoding parallel algorithm," *2nd IEEE International Symposium on Information Technology Application*, 2008.

[2] Y. W. Huang, T. C. Chen, C. H. Tsai, C. Y. Chen, T. W. Chen, C. S. Chen, C. F. Shen, S. Y. Ma, T. C. Wang, B. Y. Hsieh, H. C. Fang, and L. G. Chen, "A 1.3tops H.264/AVC single-chip encoder for HDTV applications," *IEEE Int.Conf.Solid-State Circuits*, Feb 2005, pp. 128-130.

[3] S. M. Akramulah, I. Ahmad, and M. L. Liou, "Parallelization of mpeg-2 video encoder for parallel and distributed computing systems," in *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, Aug 1995, vol. 2, pp. 834-837.

[4] P. Tiwari and E. Viscito, "A parallel mpeg-2 video encoder with look-ahead rate control," in *Int. Conf. Acoustics, Speech, and Signal Processing*, May 1996, vol. 4, pp.1994-1997.

[5] N. H. C.Yung and K. K. Leung, "Spatial and temporal data parallelization of the h.261 video coding algorithm," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 11, no. 1, pp. 91-104, Jan. 2001.

[6] Z. Zhao, P. Liang, "A highly efficient parallel algorithm for H.264 video encoder," *31st IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.

[7] S. Sun, D. Wang, and S. Chen, "A highly efficient parallel algorithm for H.264 encoder based on macro-block region partition," *HPCC 2007, LNCS 4782*, pp. 577-585, 2007.

[8] Y. Chen, E. Q. Li, X. Zhou, and S. Ge, "Implementation of H.264 encoder and decoder on personal computers," *Journal of Visual Communications and Image Representation, 17*, 509–532.

[9] JM11.0, http://iphome.hhi.de/suehring/tml/download/old_jm/jm11.0.zip