A HIGHLY EFFICIENT PARALLEL ALGORITHM FOR H.264 VIDEO ENCODER

Zhuo Zhao, Ping Liang Senior Member IEEE

Department of Electrical Engineering
University of California, Riverside, CA, 92521, U.S.A
Email: {zhaozhuo, liang}@ee.ucr.edu

ABSTRACT

In this paper, a highly efficient parallel algorithm of H.264 encoder is proposed. This algorithm fully exploits all data dependencies for maximal compression and the best reconstructed quality. Furthermore, this new method achieves the optimal compression at a frame rate that increases approximately linearly as the number of parallel processing elements [1]. Both the encoding speed and compression efficiency are improved compared to previous approaches. This paper also gives the relation between the number of parallel processing elements and the theoretical encoding time and the relation between the number of processors and the number of concurrently processed frames. Our simulation results show that this algorithm can also achieve the optimal encoding quality as a sequential processing encoder provided by Joint Video Team (JVT).

1. INTRODUCTION

H.264/AVC [2] developed by the JVT significantly outperforms previous standards in compression ratio due to many new features including quarter-pel motion estimation (ME) with variable block sizes and multiple reference frames, intra-prediction, CAVLC, CABAC, in-loop deblocking filter and more.

Compared with MPEG-4 Simple Profile, up to 50% bitrate reduction is achieved at the cost of more than four times of computational complexity [3]. Therefore, hardware or software acceleration, especially parallel structure, is a must for real-time applications.

Multiple-processor and multiple-threading encoding system have been used for real-time video encoding [4]. However, up to now, no satisfactory solution has been found that can partition video data for parallel processing while at the same time maximally exploit the temporal and spatial data dependencies for optimal coding efficiency.

Parallel algorithms are discussed in many papers,e.g., [4] [5] [6] [7] [8]. A single chip encoder for H.264 in [5] used a four-stage macroblock pipeline architecture. Although satisfactory R-D tradeoff is reported, it made approximations of neighboring encoding information in finding the optimal coding mode of the current MB. Therefore, its coding result can only be sub-optimal. An H.264 encoder using the Intel hyper-threading architecture is reported in [4]. It splits a frame into several slices and these slices are processed by multiple threads. However, these extra slices bring heavy overheads because of the slice header and the impairments to data dependencies among MBs. Y.K.Chen,et al. performed experiments to find relations between bit-rate and the number of slices in a picture [4]. In another word, the approach in [4] sacrifices coding efficiency in exchange for parallel threading. In [6], a frame is divided into many small partitions with overlapping areas and these partitions are processed concurrently. Unfortunately, this kind of partition is not feasible for H.264, because the encoder needs previous MVs in raster-scan order.

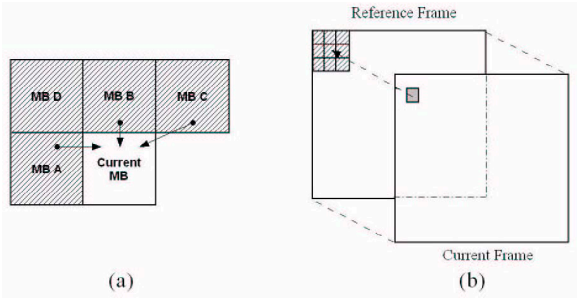

Fig. 1. Intra- & inter-frame data dependencies

We will show in this paper that how our new method does not suffer from the problems faced by prior art approaches.

The structure of this paper are as follows. Section 2 provides an overview of the data dependencies in H.264. Section 3 gives a detailed introduction for our data partition and task assigning methods. Section 4 provides the software simulation results and Section 5 is the conclusion and our future work.

2. PRIMARY DATA DEPENDENCIES IN H.264

The reference software JM 9.0 for H.264 provided by JVT adopts sequential processing of each macroblock (MB) and creates data dependencies that makes parallel processing difficult. However, by exploring these data dependencies, the JM

encoders can produce the optimal bitstream in terms of coding efficiency, therefore, the highest compression ratio. For easy comparison with the JM, this paper only considers coding efficiency. We realize that optimal subjective quality may have different requirements, however it is not dealt in this paper.

Our objective is to explore elements of the encoder that can be processed in parallel and at the same time, maximally exploit the temporal and spatial data dependencies for optimal coding efficiency.

2.1. Predicted Motion Vector & Inter-prediction

In inter-prediction, predicted motion vector (PMV) defines the search center of motion estimation. This variable is very useful in maintaining continuity of the motion field. It is determined by the MVs of its neighboring subblocks (MV_A, MV_B and MV_C) and the corresponding reference indexes [2], as shown in Fig.1 (a). Only the difference (MVD) between the final optimal motion vector (MV') and PMV will be encoded.

Similar to previous standards, H.264 also needs the reconstructed images from encoded frames as reference to exploit temporal redundancy. Thus, at least the co-located MB and its eight neighboring MBs in the reference frames must be available before current MB can be encoded, as shown in Fig.1 (b).

2.2. Quarter-pel interpolation and deblock-filtering

In H.264, traditional half-pel accuracy prediction is extended to quarter-pel. Before the reconstructed result of current MB can be used as reference for its next frame's coding, it must be interpolated to get these pixel values in quarter-/half-pel positions. This operation in the boundary area of current MB needs 3 rows/columns of pixels from its neighboring MBs (A, B, C, D, see Fig.1 (a)).

Also, in order to compensate the block artifacts brought by block-based ME and transform. H.264 uses a adaptive deblock-filter. The filter-strength factor is also determined by the prediction mode and pixel values of neighboring blocks [2]

2.3. 4×4 & 16×16 intra-prediction & mode decision

H.264 has 9 4×4 intra-predictions and 4 16×16 intra-predictions. For detail, see [2]. As shown in Fig.2, the dark pixels from neighboring blocks of current MB are needed for intra-prediction.

2.4. Context-adaptive variable length coding (CAVLC)

CAVLC is another powerful tool which makes H.264 efficient. The first VLC, coeff_token, encodes both the total number of non-zero transform coefficients (TotalCoeffs) and the

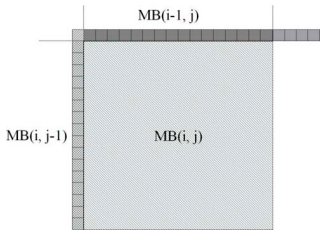

Fig. 2. Intra-prediction data dependencies

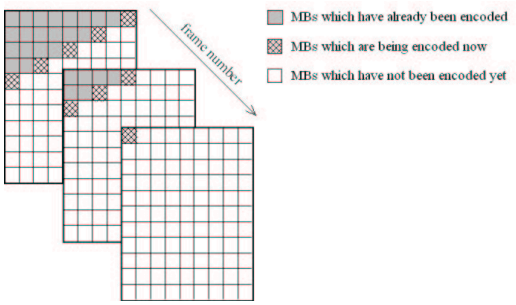

Fig. 3. Concurrently processed MBs

number of trailing $\pm 1 (T1)$. There are 4 choices of look-up table to use for encoding coeff_token, described as Num-VLC0, Num-VLC1, Num-VLC2 and Num-FLC (3 variable-length code tables and a fixedlength code). The choice of table depends on the number of non-zero coefficients in upper and left-hand previously coded blocks N_U and N_L.

3. DATA PARTITION AND TASK PRIORITY

3.1. Data partition

From section 2, we observe the following:
(1) MBs in different frames can be processed concurrently, only if its necessary reconstructed MBs from reference frame are all available.
(2) MBs from different MB rows in the same frame can also be processed concurrently, only if its neighboring MBs in its top MB row all have been encoded and reconstructed.

Fig.3 is an illustration of this observation. This simple but powerful observation is the basis of our new data partition and task scheduling method, hereafter referred to as Wavefront Parallelization. In this example, three frames are concurrently processed. The gray areas represent MBs which have been encoded and checkered MBs are being encoded concurrently now.

A distinguishing property of Wavefront Parallelization [1] is that if there are sufficient numbers of processors and the video sequence is long enough, Wavefront Parallelization can achieve a constant frame rate regardless how large the video

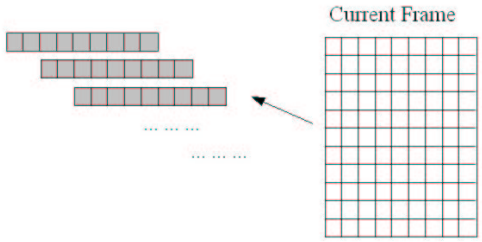

Fig. 4. Processing units in a frame

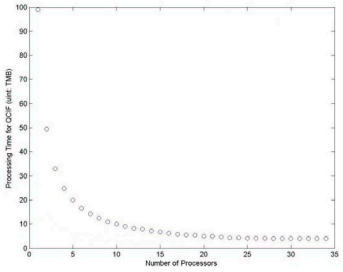

Fig. 5. Theoretical processing time per frame for QCIF

format is, e.g., QCIF, CIF or HDTV720. Taking Fig.3 for instance, the encoding process of a new frame can be started as soon as the 2x2 MBs in the top-left corner of its previous frame are all encoded, because this is the minimum requirement for motion search of inter-prediction. This means, with the increase of frame number, the average encoding time for a frame approaches to only 4 TMB (where TMB is the encoding time for a macroblock). The number of processor units needed to achieve this is:

$$P_n = \lceil frame\_size\_in\_MB/4 \rceil$$

In practice, we cannot have a large number of processor units. We will show that majority of the benefit can be achieved with only a small number of processor units.

In Wavefront Parallelization, each frame is first partitioned into MB rows as shown in Fig.4. This is because a MB cannot be processed until its left neighbor in the same MB row is encoded. All MBs in the same MB row will be processed by the same processor or thread to reduce data exchanges between processors.

3.2. Task assigning and priorities

Fig.7 gives the task assigning timing diagram for the ideal case (number of processors = P_n). Here, $T = TMB$. However, this requires 25 processors for QCIF to achieve the optimal encoding speed; 99 processors for CIF, and 900 for HDTV720P. This is not practical.

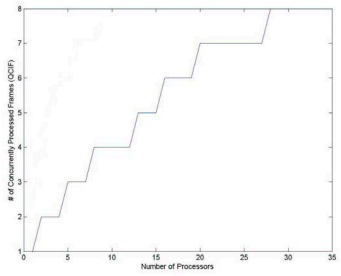

Fig. 6. Number of concurrently processed frames (QCIF)

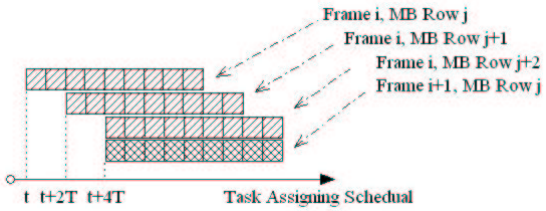

Fig. 7. Task assignment timing diagram

Fig.5 is our simulation result of the theoretical encoding time of a video frame with QCIF format. The results for CIF, HDTV720P are also similar. This figure shows the relation between the number of processors (horizontal axis) and the encoding time of a frame (vertical axis and with TMB as the unit). As can be seen, most of the speedup benefit can be achieved using a small number of processors.

If only a small portion of processors are used, not all the checkered MBs in Fig.3 that are ready for processing can be processed concurrently. Thus, priority based task scheduling is necessary to guarantee that the processors can be fully utilized.

Here we defined the priorities in two levels: the first is the inter-frame level, and the second is the intra-frame level. The priority of the inter-frame level is higher than the intra-frame level. The inter-frame level priority means that in the video source buffer, if several MBs belonging to different frames are ready to be encoded concurrently, the MBs in the frame with smaller frame number should be encoded first. The intra-frame level priority means that if several MBs belonging to different MB rows in the same frame are ready to be encoded concurrently, the MBs in the row with smaller row index should be encoded first.

4. SOFTWARE SIMULATION

Our wavefront simulator is developed using C language and implemented in a PC with a P4 2.8GHz processor and a 512MB memory. The simulation results are compared with those from JM 9.0, a sequential encoding structure.

Table 1. Simulation Result for "Grandma.yuv" (QCIF)

	Avg enc time per frame	SnrY	SnrU	SnrV	# of bits	Speed up
Wavefront simulator	273 ms	37.157	39.869	40.450	61464	3.17
JM 9.0	865 ms	37.157	39.869	40.450	61464	1

Table 2. Simulation Result for "Paris.yuv" (CIF)

	Avg enc time per frame	SnrY	SnrU	SnrV	# of bytes	Speed up
Wavefront simulator	1272 ms	35.729	39.181	39.279	128419	3.08
JM 9.0	3914 ms	35.729	39.181	39.279	128419	1

In our software simulation of an encoder for H.264 baseline profile, four processors are simulated. Some of the encoder parameters are: one reference frame, searching range for ME: ± 10, Hadamard transform is used for motion cost estimation, full R-D optimization, CAVLC for entropy coding.

We also performed simulations to obtain the relationship between the number of processors (horizontal axis) and the number of concurrently processed frames (vertical axis) for different video formats. Due to space limitation, only result for QCIF is given in Fig.6.

In order to simplify the encoder, we choose to encode 2 frames simultaneously at the most. From Fig.6, we found that 4 processors are the upper limit.

The simulator collects the maximal encoding time among every 4 concurrently processed MBs and the corresponding time spent on data partition and communication. Some of the simulation results are presented in Table.1 and Table.2. In Table.1, we used "Grandma.yuv" (QCIF) as video source, 50 frames are encoded as "IPPP" structure. In Table.2, we used "Paris.yuv" (CIF) as video source, 50 frames are encoded as "IPPP" structure. Experiments show that a speedup of more than 3 times is achieved and the encoding quality is the same as JM 9.0.

5. CONCLUSION AND FUTURE WORK

This paper presents the new Wavefront Parallelization method for H.264 encoder. Analysis and simulation results show that it can achieve the optimal compression at a frame rate that increases approximately linearly as the number of parallel processing elements. To our knowledge, no prior art approach is able to achieve this. We are investigating the memory and data flow optimization to further improve its performance. Future research include a multi-core SOC implementation and mapping to a multithread processor.

6. REFERENCES

[1] Z. Zhao and P. Liang, "A highly efficient parallel processing system for h.264/avc real-time video encoder," *Pending US Patent Application 60/691,603*.

[2] JVT, "Draft recommendation and final draft international standard of joint video specification," *ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003*.

[3] T-C. Chen, Y-W. Huang, and L-G. Chen, "Analysis and design of macroblock pipelining for h.264/avc vlsi architecture," in *Proceedings of the 2004 International Symposium on Circuits and Systems*, May 2004, vol. 2, pp. II–273–6.

[4] Y-K. Chen, S. Ge T. X, and G. M, "Towards efficient multi-level threading of h.264 encoder on intel hyper-threading architectures," in *18th Int.Parallel and Distributed Processing Symposium*, Apr 2004, p. 63.

[5] 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," in *IEEE Int.Conf.Solid-State Circuits*, Feb 2005, pp. 128–130.

[6] 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.

[7] 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.

[8] 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.