Anthony Vetro
Mitsubishi Electric Research Labs

Leveraging Parallel Computing in Modern Video Coding Standards

Kiho Choi and Euee S. Jang
Hanyang University, Korea

The demands for multimedia processing have soared in recent years. It is now common for all forms of computing equipment, from high-end servers and computing clusters to handheld devices, to be equipped with multicore general processors and graphics processing units (GPUs). Computationally expensive applications such as 1,080-pixel high-definition video recordings with real-time image stabilization and high-quality video playback have been the driving force behind this recent boom in multicore processing.

In the single core era, most multimedia applications could not be run on a single CPU because of their high computational complexity. As a result, hardware-oriented solutions such as special purpose coprocessors, digital signal processors, and dedicated hardware accelerators were required to cope with multimedia applications. Although hardware-oriented solutions provide the best performance, they result in increased cost with limited design flexibility and generally require a longer time to market. Industry has introduced reconfigurable hardware solutions such as field-programmable gate arrays (FPGAs) to shorten design time and cost and to ensure greater flexibility, but with reduced performance.

The parallel computing paradigm enables software-oriented solutions using multicores or GPUs to compete with hardware-oriented solutions. Software-oriented solutions are becoming more attractive to vendors because they do not require specialized hardware. With an ever-increasing number of cores, more computationally intensive tasks are now possible with software-oriented solutions. Performance improvements in multicore systems come from the divide-and-conquer approach to computing model design by dividing and executing as many simultaneous independent tasks as possible. Dramatic performance improvements have been reported based on parallel computing. For example, one bioinformatics study reportedly produced a solution after 13 hours using 7,168 GPUs instead of spending 10 years using a single CPU.[1] As one of the most computationally intensive areas, people are now applying parallel computing concepts to a broad range of multimedia applications, including video analysis, understanding, retrieval, and coding.

This article reviews the influence of the parallel computing trend on the design of modern video coding standards. In particular, we examine the requirements for parallel video coding and describe changes to the processing architecture as well as modifications in the design of specific tools to facilitate improved parallel processing.

Parallel Computing in Video Coding

A typical video encoder is required to encode each video frame within tens of milliseconds to support real-time applications at a given frame rate—for example, ranging from 24 to 120 Hz. In the early days of MPEG-2 video, such requirements could only be satisfied with ASIC implementations. Nowadays, we can meet the real-time requirement of video codecs with increasingly high resolutions and frame rates.

Generally, the number of cores proportionally improves system performance. However,

Editor's Note

Video coding has always been a computationally intensive process. Although dramatic improvements in coding efficiency have been realized in recent years, the algorithms have become increasingly complex, making it necessary to realize the capabilities of multicore processors. This article discusses how recent trends in parallel computing have influenced the design of modern video coding standards.

1070-986X/12/$31.00 © 2012 IEEE

Published by the IEEE Computer Society

The key difficulty in designing efficient parallel computing algorithms results from the discrepancy between the algorithm definition and architectural modeling.

this does not hold true if the parallelism of the given algorithm is limited, which is the case for conventional video coding standards. One of the primary limitations stems from the sequential nature of the bitstream, which is a 1D sequence of bits that is produced by an encoder and must then be processed by the decoder to reconstruct a sequence of pictures. This makes it difficult to design parallel implementations of such algorithms. Early video coding standards such as H.261 and MPEG-2 introduced the concept of a slice to independently encode and decode partitions of a video frame. By removing the dependency between different partitions, we can support resilience to errors in the bitstream and facilitate parallel processing. However, slice-level parallelism incurs a decrease in compression efficiency because they do not fully exploit spatial correlations. Researchers reported bitrate increases of approximately 15 to 20 percent when the number of slices in each frame is limited to 9.[2]

The key difficulty in designing efficient parallel computing algorithms results from the discrepancy between the algorithm definition and architectural modeling. Most often, the algorithm is defined before the architecture is modeled. Recently, co-exploration between algorithm and architecture (CEAA) has been identified as an important research topic for overcoming these challenges.[3] Within the CEAA approach, algorithm definition takes place while simultaneously considering the architectural aspects.

A good example of the CEAA approach being applied during the algorithm definition stage is the High Efficiency Video Coding (HEVC) standard, which is being jointly developed by ISO/IEC JTC1/SC29 WG11 (MPEG) and ITU-T SH16/Q.6 (VCEG). This new video coding standard must meet two critical requirements: twice or better compression efficiency than MPEG-4 AVC/H.264 and reasonable computational complexity to guarantee high-resolution video—for example, 150 megapixels per second (Mpels/s).[4] These two requirements are hard to meet because they are, in fact, contradictory. Therefore, the introduction of the CEAA concept seems to be unavoidable.

Parallel Computing Designs in HEVC

The key requirement of the CEAA approach in HEVC is to remove dependencies (such as data dependency, syntax dependency, and ordering constraints) as much as possible while maintaining compression efficiency. We can categorize the various dependency removal methods in HEVC into two types: *high-level dependency removal*, which minimizes the dependencies in the bitstream syntax and the video coding structure, and *low-level dependency removal*, which attempts to minimize dependencies required by a specific tool or algorithm.

High-Level Dependency Removal

Video decoding processes are highly sequential because most of the decoding process depends on sequential parsing of the bitstream. A simple method for accelerating the decoding process with parallel computing is to remove dependency at the bitstream level by specifying the bitstream syntax so that several independent chunks of bits can be decoded simultaneously. Another possible method for removing dependency is to exploit the video coding structure to better support parallel computing with additional syntax information. Such a change can facilitate efficient modeling with parallelism, with a marginal increase in bitstream size due to the extra overhead information in the bitstream. HEVC uses the following technologies for high-level dependency removal: entropy slices, wavefront parallel processing, tiles, and parallelized merge/skip mode.

Entropy slices are a good example of bitstream syntax dependency removal. An entropy slice is an independent slice that can be parsed completely without reference to other slices, but in contrast to conventional slices,

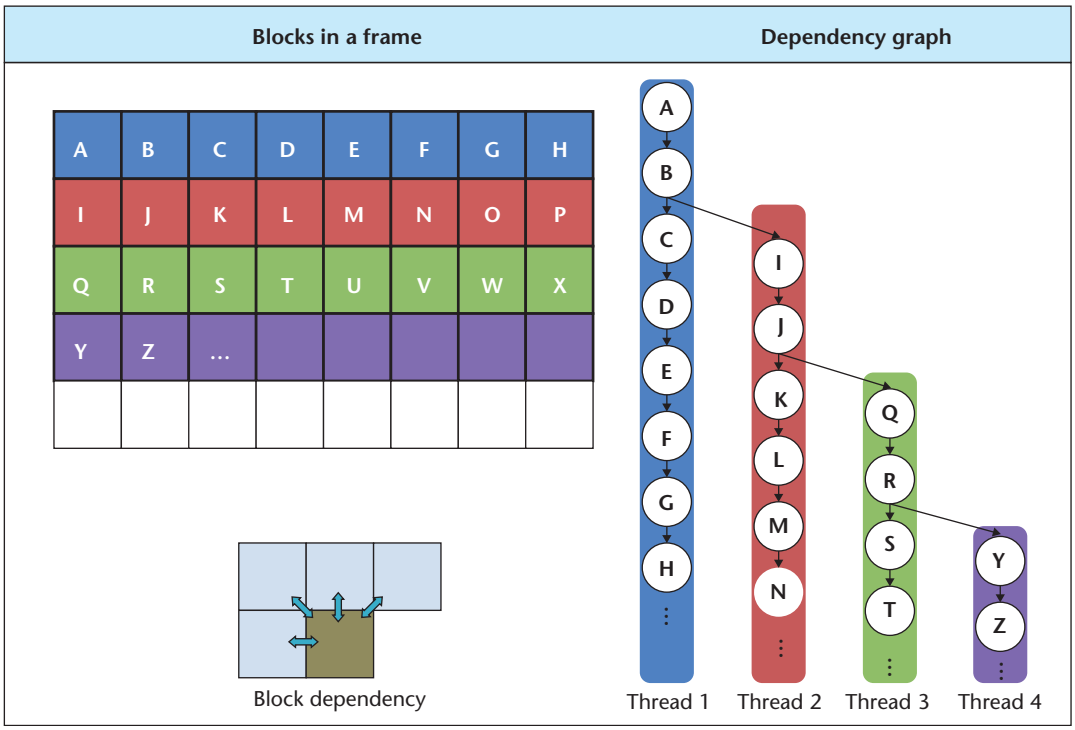

Figure 1. Dependency graph. The multithreaded High Efficiency Video Coding (HEVC) codec uses wavefront parallel processing (WPP) with four threads.

other decoding processes such as intraprediction and motion compensation can refer to neighboring slices to minimize the loss in coding efficiency. One obvious advantage of using entropy slices is the parallelism of bitstream parsing, which can accelerate the decoding process in proportion to the number of entropy slices.

Wavefront parallel processing (WPP) is another example of bitstream syntax dependency removal. WPP rearranges the coding order of blocks to introduce parallelism, as Figure 1 shows. The WPP concept is not new, but HEVC is the first video coding standard to adopt a bitstream syntax that can embrace WPP technology without sacrificing compression efficiency. With the newly added bitstream syntax, we can avoid reordering operations, which directly affects compression efficiency. One study reported that a decoder using WPP—that is, two threads with 16 substreams—performed about twice as fast as a decoder without WPP, with negligible loss in compression efficiency.[5]

Tile structure is an example of video structure dependency removal. To provide the parallelism capability while preserving coding efficiency, HEVC has incorporated the tile concept frequently used in image coding standards such as JPEG and JPEG 2000. Compared to the slice structure, the tile structure provides a flexible rectangular shape structure to partition a picture frame into a set of rectangular times. Figure 2 shows that that a tile might contain more than one slice and vice versa. The advantage of using tiles is that the structure is beneficial to parallel processing owing to its support of the explicit division of a frame into multiple independent tiles. This tile structure, while supporting the same parallelism capability as the slice structure, improves the coding efficiency over the slice structure. Arild Fuldseth and his colleagues reported that a well-defined

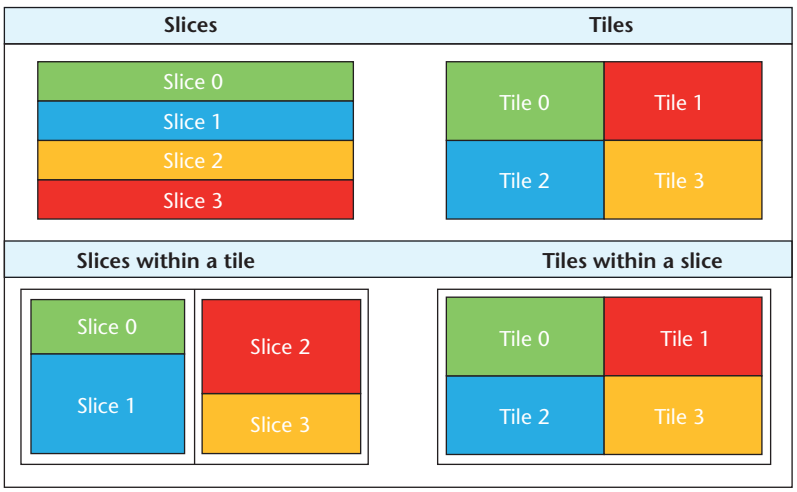

Figure 2. Tile structure compared with conventional slice structure.

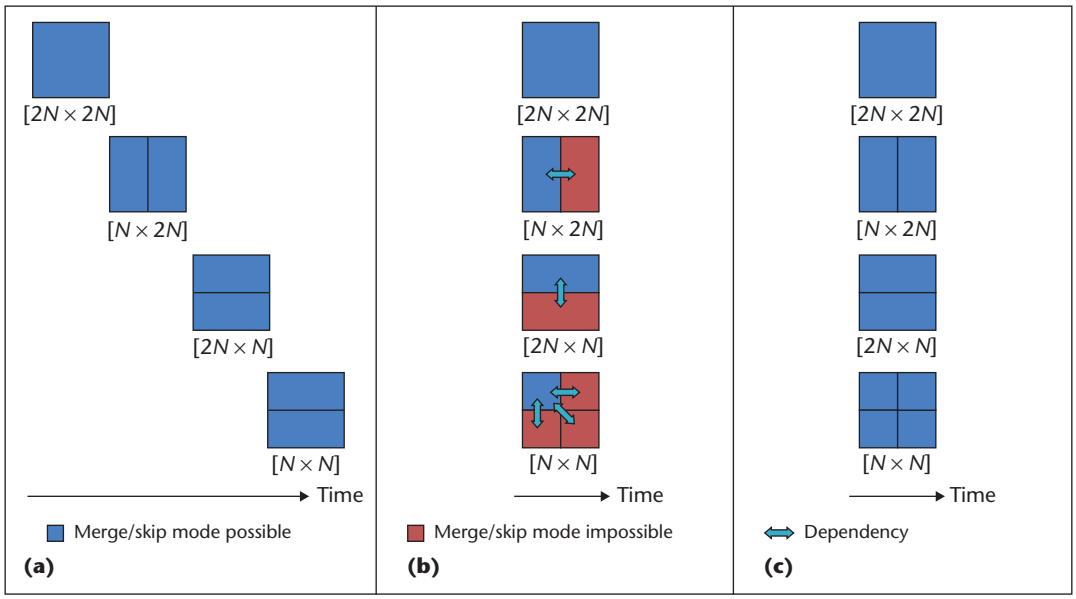

Figure 3. Possibility of parallel merge/skip mode according to data dependency. (a) Sequential merge/skip mode, (b) parallel merge/skip mode without syntax, and (c) parallel merge/skip mode with syntax.

tile structure demonstrated the same parallelism functionality as the slice structure with a coding gain of up to 4.8 percent when compared with the slice structure.[6] Thus, the CEAA approach does not always sacrifice compression efficiency to decrease data dependency.

Video coding standards only specify the bitstream syntax and the decoding process, thereby allowing the encoding process to be optimized for coding performace and implementation efficiency. As such, the bitstream syntax is typically designed from a decoder's perspective. However, the standardization of HEVC considered syntax to support efficient encoding. Specifically, HEVC adopted syntax to support a *parallelized merge/skip mode* to address the challenges associated with the implementation of a highly sequential merge/skip mode algorithm.[7] A parallelized merge/skip mode in the bitstream syntax signals the decoder to use only the available motion vector information, which results in a substantial reduction of data dependency at the encoder side. As Figure 3 shows, this syntax change not only enables the parallelism of the encoder, but also preserves the coding efficiency.

Low-Level Dependency Removal

The objective of low-level dependency removal is to reduce data dependency at the tool level. Although high-level dependency removal can effectively increase the parallelism in the high-level processing of a bitstream, we can conduct further dependency removal on computationally intensive tools, especially on the decoder side. Typical examples of low-level dependency removal include inverse transform and context-adaptive binary arithmetic coding (CABAC) decoding.

Single instruction, multiple data (SIMD) optimization methods are widely used in real-time system implementations. Because the SIMD technology enables multiple arithmetic operations to be executed simultaneously, codecs using SIMD optimization have significantly improved decoding speed. The transform operation, in particular, is a well-suited tool that can be efficiently improved with SIMD technology. The design of the transform core in HEVC considered SIMD-level support.[8] A key advantage of this design is that the transform is favorable to parallel processing with minimal dependency and control logic. For example, the adopted transform design enhances the implementation flexibility of matrix multiplications and partial butterfly by providing 16-bit data representation before and after each transform stage.

CABAC is one of the most coding efficient tools in MPEG-4 AVC/H.264 owing to its use of efficient context modeling. Context modeling is good for coding efficiency, but it incurs high data dependency and is a highly sequential processing method. The HEVC effort is working to eliminate the dependencies and the critical path delays in CABAC decoding for parallel processing.[9] For example, the significance map coding design in CABAC allows

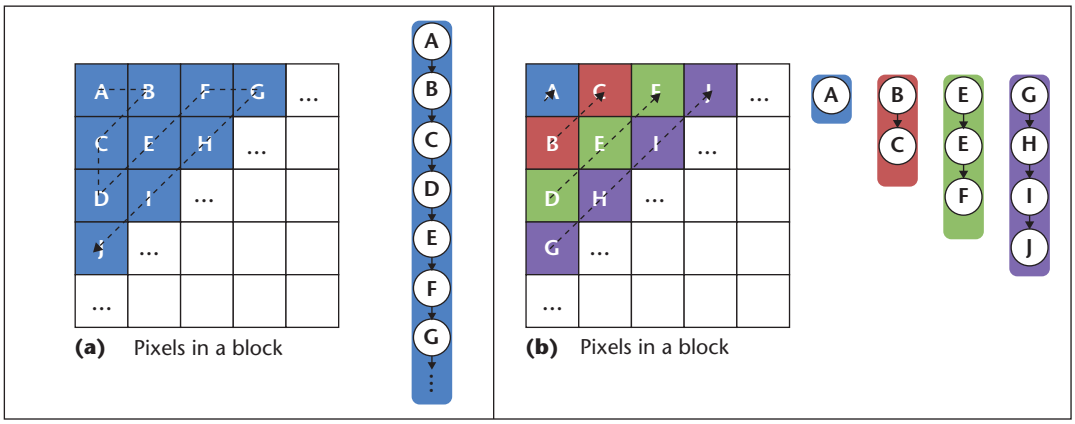

Figure 4. Scan order and parallelism in significance map coding using context-adaptive binary arithmetic coding (CABAC). (a) Zigzag scan order and sequential processing. (b) Diagonal scan order and parallel processing.

multiple simultaneous bitstream parsing by adopting a top-right direction triangle-shaped scan order to determine the adjacency relationship. This change of scan order in significance map coding enables parallel CABAC processing with reduced dependency among pixels (see Figure 4).

Conclusion

In the future, the CEAA approach to video coding will draw attention because it attempts to address conventional algorithm definition requirements as well as implementation requirements by exploring algorithm and architecture together. In that sense, HEVC will be recognized not only as the most compression-efficient coding standard, but also as the first coding standard in which the CEAA approach was seriously considered. **MM**

Acknowledgments

This work was supported by a National Research Foundation (NRF) of Korea grant, funded by the South Korean Government (MEST) (201100000001179) and Seoul R&BD Program (PA100094), Korea.

References

1. M. Feldman, ''BGI Speeds Genome Analysis with GPUs,'' Dec. 2011; www.hpcwire.com/hpcwire/2011-12-15/bgi_speeds_genome_analysis_with_gpus.html.
2. Y.K. Chen et al., ''Implementation of H.264 Encoder and Decoder on Personal Computers,'' *J. Visual Communication and Image Representation,* vol. 17, no. 2, 2006, pp. 509–532.
3. G.G. Lee et al., ''Algorithm/Architecture Co-exploration of Visual Computing on Emergent Platforms: Overview and Future Prospects,'' *IEEE Trans. Circuits and Systems for Video Technology,* vol. 19, no. 11, 2009, pp. 1576–1587.
4. T. Wiegand et al., ''Special Section on the Joint Call for Proposals on High Efficiency Video Coding (HEVC) Standardization,'' *IEEE Trans. Circuits and Systems for Video Technology,* vol. 20, no. 12, 2010, pp. 1661–1666.
5. G. Clare, F. Henry, and S. Patexu, ''Wavefront Parallel Processing for HEVC Encoding and Decoding,'' *Proc. Joint Collaborative Team on Video Coding (JCT-VC) 6th Meeting,* 2011, article JCTVC-F274.
6. A. Fuldseth et al., ''Tiles,'' *Proc. Joint Collaborative Team on Video Coding (JCT-VC) 6th Meeting,* 2011, article JCTVC-F335.
7. M. Zhou, ''AHG10: Configurable and CU-group level parallel merge/skip,'' *Proc. Joint Collaborative Team on Video Coding (JCT-VC) 8th Meeting,* 2012, article JCTVC-H0082.
8. A. Fuldseth, G. Bjøntegaard, and M. Budagavi, ''Core Transform Design for HEVC,'' *Proc. Joint Collaborative Team on Video Coding (JCT-VC) 7th Meeting,* 2011, article JCTVC-G495.
9. B. Bross et al., ''High-Efficiency Video Coding (HEVC) Text Specification Draft 6,'' *Proc. Joint Collaborative Team on Video Coding (JCT-VC) 8th Meeting,* 2012.

Kiho Choi is a doctoral candidate in the Digital Media Laboratory at Hanyang University, Korea. Contact him at aikiho@gmail.com.

Euee S. Jang is a professor in the Digital Media Laboratory at Hanyang University, Korea. Contact him at esjang@hanyang.ac.kr.