Advanced Science and Technology Letters
Vol.28 (CIA 2013), pp.72-76
http://dx.doi.org/10.14257/astl.2013.28.14

An OpenCL-based Implementation of H.264 Encoder

Young Chun Kwon[1], Chanho Park[2], Daeseok Oh[2],
WooSuk Jang[3], and Nakhoon Baek[1,*]

[1] School of Computer Sci. and Eng., Kyungpook National Univ., Daegu 702-701, Korea
[2] Kumoh National Institute of Technology, Gumi 730-701, Korea
[3] Yeungnam University, Gyeongsan 712-749, Korea
kown100@nate.com, cksgh303@gmail.com, rar555@nate.com,
passions707@naver.com, oceancru@gmail.com

Abstract. We present an accelerated implementation of high-speed video stream encoder for the H.264 digital video codec standard. Based on the parallel processing techniques with GPU's, we used an OpenCL-based GPU kernel programs. We achieved a high-level CPU-GPU interoperability, through making CPU perform all input/output operations and overall stream control, while GPU does the core encoding operations. Our final result shows remarkable speed-up in comparison with the previous implementations.

Keywords: Video codec, H.264, OpenCL, parallel processing, implementation.

1 Introduction

H.264 is one of the high-quality digital video codec standards [1]. It is now one of the most widely used video codes. The overall encoder processing of the H.264 standard requires huge amount of computations, to guarantee the final video quality. Though most previous H.264 encoder implementations are focused on their CPU-based accelerations, there are a few results on the parallelized H.264 encoders, recently.

In this paper, we represent a new acceleration method for the H.264 decoders, based on the OpenCL (Open Computing Language). OpenCL [2] is currently one of the most widely used de facto standards in the field of general-purpose GPU (GPGPU) computing. Most previous researchers are concentrated on the acceleration of CPU computing, or introducing intuitive parallel processing architectures. In contrast, we focused on the overall parallelization of the encoding operations, and finally achieved remarkable speed-ups.

In Section 2, we present previous CPU-based and GPGPU-based encoding schemes for the H.264 encoder. Our overall parallelized architecture and details of implementation algorithms are followed in Section 3. We show the actual implementation results in Section 4. Section 5 shows our conclusion and future works.

* corresponding author: Nakhoon Baek, oceancru@gmail.com

† This investigation was financially supported by Semiconductor Industry Collaborative Project between Kyungpook National University and Samsung Electronics Co. Ltd.

ISSN: 2287-1233 ASTL

2 Previous Works

GPGPU-based methods have been widely used for not only encoder processing but also various areas including image processing, big-scale data processing, and others. Specifically for the encoder processing, CUDA [3] and OpenCL are used for various video encoding schemes to enhance their internal algorithms. In the case of H.264, it shows relatively good encoding performance, and there are CUDA-based acceleration examples [4] and also OpenCL-based acceleration examples [5].

In this paper, we are focusing on the OpenCL standard. In the case of CUDA, it was released from a single manufacturer, NVIDIA, and thus, it cannot be easily used for the general devices from other hardware vendors.

3 Design and Implementation

There has been a previous work [6] for the H.264 encoding pipeline implementation, with OpenCL. Our work is based on their implementation. We analyzed their overall architecture and located the key operations to be parallelized. Using OpenCL, we accelerated those key operations to finally achieve more optimized performance.

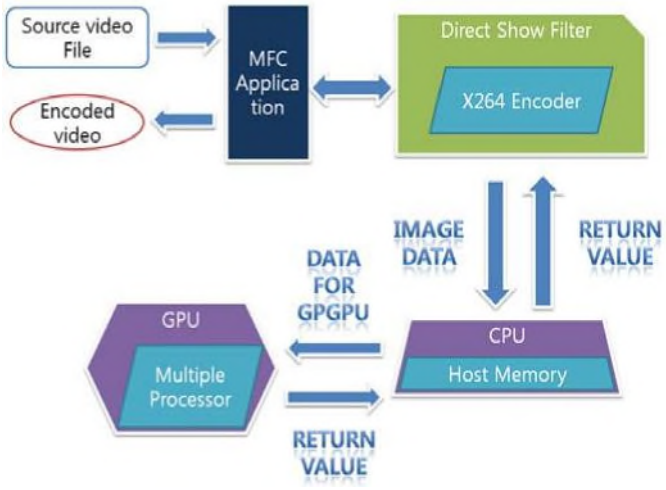


Fig. 1. Overall architecture of our parallelized H.264 encoder.

Figure 1 shows the overall architecture of our proposed algorithms. To concentrate on the encoding operations, we used the input/output features of the existing DirectShow filters [7] for the file input/output operations. When the target video files are read from the disk images, the DirectShow filters construct the raw image data in the host memory. Then, these image data are sent to the GPU, and the OpenCL-based GPU kernel program does the core encoding operations. The finally encoded results

are copied back the host memory, and the DirectShow filter converts them to the final encoded video files on the disk.

At the early stages of our parallelized program design, we analyzed that the overall operations are concentrated on the per-screen operations. Thus, we first tried to optimize these per-screen operations. However, our optimization immediately causes rapid increase of data transfer between the GPU and host memory. This kind of bottle-neck problem limits the overall performance of our architecture.

Thus, we changed our optimization strategy to optimize another processing step of look-ahead operations. The look-ahead operations examine the future frames in the video stream, and decide the frame types and slice types of them in advance, according to the H.264 standard specification.

The whole look-ahead processing is implemented as a single thread, and this thread uses the OpenCL-based GPU kernel to determine each slice type. The OpenCL kernel internally performs 4 steps: down-scale, intra-prediction, motion-search, and mode select. At the down-scale stage, the H.264 macro blocks are converted into lower resolution images. And then, the intra-prediction stage calculates the intra-cost estimation values. At the motion search stage, the motion vectors are obtained from the scaled-down images and intra-cost values. At the mode select stage, each slice types are selected from the cost values from the intermediate results.

Our accelerated look-ahead processing based on the OpenCL-based GPU kernels achieves overall enhancements on the system processing speed. More detailed results are shown in the following section.

4 Results

Our system is implemented on a Windows PC with an Intel i7 CPU and GeForce GTX560 Ti GPU. To check the execution speeds of a set of H.264 encoders, we use the same parameter settings as possible. Figure 2 shows the final benchmark results from a set of encoders, our implementation, x264 [6], MeGUI [8] and StaxRip [9]. Other encoders, x264, MeGUI, and StaxRip are selected from widely used H.264 encoders.

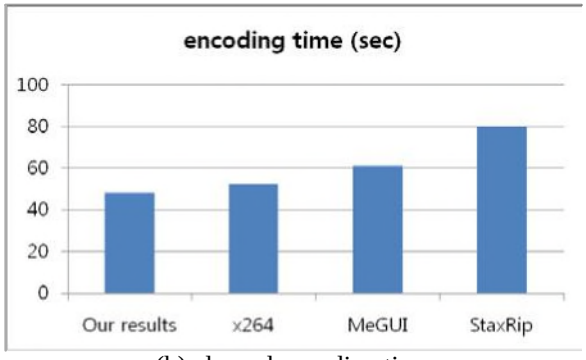
As shown in Figure 2.(a), for the example video file of 154MB with 1920x1080 resolution, the finally encoded file sizes are almost same sizes of about 34MB. In the case of encoding time, our implementation shows the best result. It shows about 50% enhancement in comparison with StaxRip. And, it shows at least 10% speed-up even in comparison with the previous x264 implementation.

5 Conclusion

We aimed to reduce the H.264 digital video encoding time, with GPGPU techniques. We started from analyzing the pros and cons of existing H.264 encoders, and located the heavy-computation areas in the real implementation. And, we applied parallel-processing techniques to those locations, while we preserve the already-optimized areas in the existing implementation. Finally, our OpenCL-based GPU kernel

implementation achieves 10%-to-50% speed-ups in comparison with widely-used H.264 encoders.

(a) final file sizes



(b) elapsed encoding time

Fig. 2. Experimental results from various implementations, for 154MB video file with 1920×1080 resolution.

Acknowledgments. This investigation was financially supported by Semiconductor Industry Collaborative Project between Kyungpook National University and Samsung Electronics Co. Ltd.

References

1. Wiegand, T., et al.: Overview of the H.264/AVC Video Coding Standard, IEEE Trans. on Circuits Systems for Video Technology, vol.13, no.7, pp.560-576 (2003)
2. Munshi A.: The OpenCL Specification, version 1.0.29, Khronos OpenCL Working Group (2012)
3. NVIDIA: CUDA C Programming Guide, version 2.5, NVIDIA (2012)
4. Wu, N., Wen, M., Su, H., Ren, J., Zhang, C.: A Parallel H.264 Encoder with CUDA:Mapping and Evaluation, IEEE 18th Int'l Conf. on Parallel and Distributed System (2012)

5. Marth, E., Marcus, G.: Parallelization of the x264 encoder using OpenCL, ACM SIGGRAPH 2010 Posters Article No. 72 (2012)
6. Merritt, L., Vanam, R.: Improved Rate Control and Motion Estimation for H.264 Encoder, IEEE Int'l Conf. on Image Processing 2007, vol. 5, pp.309-312 (2007)
7. Polinger, A.: Developing Microsoft Media Foundation Applications, Microsoft Press (2011)
8. MeGUI, http://sourceforge.net/projects/megui/
9. StaxRip, http://staxmedia.sourceforge.net/

