Parallelization of the x264 encoder using OpenCL

Erich Marth, University of Mannheim, erich.marth@googlemail.com
Guillermo Marcus, University of Heidelberg, guillermo.marcus@ziti.uni-heidelberg.de
Sources Online at: http://li5.ziti.uni-heidelberg.de/x264gpu

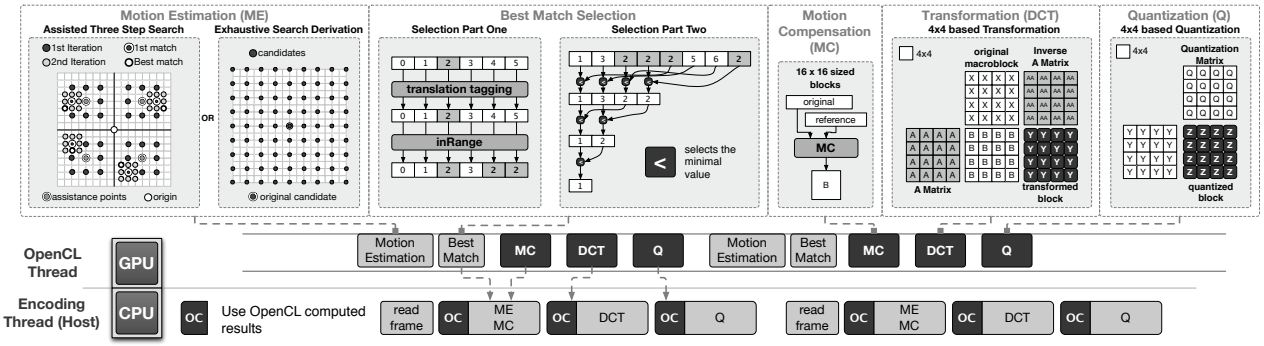

Figure 1: *OpenCL Powered Modules Of the x.264 Encoding Pipeline*

1 Introduction

With the introduction of H.264, the complexity on video encoders has increased dramatically. As hardware based encoding solutions profit from the strict sequential design and already feature real time capabilities for high definition material, software solutions lack most of the encoding performance. More precisely, the performance of software encoders is limited due to the computation power of encoding system as well as the high level of codec-intern dependencies. As a consequence, software encoders supporting high definition needs are very rare.

The increasing computation power of massive parallel architectures such as modern graphics devices can be used to speed-up the encoding of H.264 video material. Compared to plain hardware solutions, graphics device powered encoders have the advantage of much lower initial costs and at the same time offer the flexibility of boosting the performance with future device upgrades. In addition, computers of today already include high performance graphics devices, which improve encoding times with nearly zero extra costs.

While other stand alone GPU accelerated encoding solutions exist for H.264, this work shows the first working parallelization of the open source H.264 encoder x264 using OpenCL.

2 Parallelization using OpenCL

In the beginning, the parallelization was targeting a straight forward OpenCL based motion estimation without the actual integration into the encoding process. One straight forward approach was based upon the sub-optimal Three Step Search (TSS) algorithm. The implemented Assisted Three Step Search introduced additional assistant points for more concurrency. In addition, a second algorithm was implemented, derived from the computationally intensive Exhaustive Search (ES) – Full Search – algorithm. The Exhaustive Search Derivation (ESD) differs in using a reduced set of candidates – only a fourth of the original set – examining even positioned translations only.

After finishing the motion estimation, the OpenCL powered computation was integrated into the encoding flow of the x264 encoder by a plain serial design. In favor of higher encoding speeds, better device utilization as well as better adaption to the encoder architecture, the serial design was later replaced by a more autonomous OpenCL working thread approach. The new working thread pipeline was optimized by using principles from the RISC architecture. More precisely, the estimation and selection modules were stripped down to a single process, moving the extracted functionality to discrete modules. In a final step, the sub-sequential Motion Estimation, Transformation and Quantization processes were ported to OpenCL and merged into the pipeline as well.

While the Transformation was applied on blocks with 4x4 size conforming to the H.264 specification, the final Quantization process was implemented equally to the variant used inside the original x264 encoder. Compared to the H.264 specification, the x264 encoder merges the element-wise multiplication of the DCT with the Quantization step using an LUT based approach.

3 Results

Considering the fact that only a fraction of the motion estimation capabilities have been ported to OpenCL, the OpenCL powered encoding is up to 55% faster than the original Full Search based encoding of the unmodified x264. While other GPU solutions claim up to 20x speedup, independent tests against unmodified x264 shows similar gains as our implementation for FullHD. Furthermore, the current work is the first open-source, working integration into the x264 encoder that enables it to profit from the computing power of high performance graphics devices.

References

CHEN, W.-N., AND HANG, H.-M. 2008. H.264/avc motion estimation implementation on compute unified device architecture (cuda). Tech. rep., National Chiao-Tung University.

CROSS, J., 2008. GPU Accelerated Video Transcoding. Online Article, December. [online] `http://www.extremetech.com/article2/0,2845,2337057,00.asp`.

SCHWALB, M., EWERTH, R., AND FREISLEBEN, B. 2009. Fast motion estimation on graphics hardware for h.264 video encoding. *Trans. Multi. 11*, 1, 1–10.

SHIMPI, A. A., 2008. Badaboom: A Full Test of Elemental's GPU Accelerated H.264 Transcoder. Online Article, August. [online] `http://www.anandtech.com/show/2586`.

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007