Energy Efficient Video Decoding on Multi-Core Devices

Damla Kılıçarslan[1, 2], Göktuğ Gürler[1], Öznur Özkasap[1] and A. Murat Tekalp[1]
{dkilicarslan, cgurler, oozkasap, mtekalp}@ku.edu.tr

ABSTRACT

Emergence of high quality media applications results in larger data sizes as well as higher bitrates of digital multimedia contents, and their significant share on the overall Internet traffic. These lead to an increase in the energy consumption rates and performance requirements for real-time video decoding. In this study, we propose parallel video decoding solutions to provide real-time decoding performance with reduced energy consumption on multi-core devices. Various approaches of parallelism at data and task levels can be incorporated in video decoders, bringing efficiency in energy consumption rates and/or performance. We offer and develop two approaches for the H.264 standard. The former is based on a coarse-grained frame level, and the latter is a fine-grained macroblock level parallelism. The implementations were conducted on a shared memory multi-core platform as an all software solution for real-time scalable video decoding. We also discuss energy efficiency as well as performance results. As part of our ongoing work, further parallelization methods such as parallelism at slice level, and parallel decoding of consecutive groups of pictures on the H.264/SVC decoder are discussed.

Keywords

Energy efficiency, parallel video decoding, multi-threaded video decoding.

1. INTRODUCTION

Video streaming applications have become the major constituent of the global Internet traffic. Video traffic is expected to exceed 91% of global consumer traffic by 2014 [1]. Introduction of higher resolutions and 3D media contents result in more demanding computation power and energy requirements in video codecs. For instance, real-time multi-view video encoding/decoding can be a significantly timely and energy demanding process that requires multiple processors at a time [2]. Energy demanding video decoders can be inefficient in terms of users' playback experience especially on battery constraint mobile devices. The current trend in designing more powerful processors is based on multi-core architectures [3] and the gap between desktop and mobile processors is narrowing rapidly [4]. The effective software design is the key factor to divide the workload efficiently on multiple cores. It is a well-known fact that multi-core technologies can deliver noticeable levels of energy and performance gain. The motivation in this study is that significant amounts of energy savings can be achieved by exploiting multi-core technologies on the decoder side of video applications.

For video decoding, various software implementations on multi-core systems such as task-based parallelism and thread/data based parallelism are possible. The choice of an efficient parallelism approach is a matter of optimization between the overhead introduced and the performance gain achieved. The decoding process in the H.264 video decoder has several levels dependent on each other that lead to constraints for parallelization. Parallelization can be achieved at various levels: (1) the coarsest grain being the parallel decoding of parts of the video sequence on different cores, (2) parallelism of hierarchical decoding levels of group of pictures (GOPs), (3) parallel decoding of independent slices within each frame, and (4) the finest grain being the parallel decoding at the macroblock level on I and B frames. In this study, we address two of these parallel video decoding approaches, namely frame level and macroblock level parallelism.

Other related research conducted on parallel video decoders involves parallelization of real-time MPEG-2 decoding [5]. Various parallelization techniques for the H.264 standard were also examined in the macroblock level [6],[7],[8]. Coarser grain implementations were carried out at slice, GOP and frame levels [9]. Hardware based application-specific integrated circuit architecture was presented in [10]. In contrast to our current study, the results and evaluations of the all-software solutions have been focused more on performance rather than energy-efficiency.

In this study, we develop two parallel decoding approaches for the H.264 standard that are applicable on multi-core devices, one based on a coarse-grained frame level, and the other based on a fine-grained macroblock level. The aim is to keep the energy consumption of video decoding low without degrading its performance. We discuss the benefits and shortcomings of these approaches with results. Our results demonstrate that multi-threaded decoding at various levels not only provides speed-up in performance but also reduces energy consumption.

This paper is organized as follows: Section 2 describes the data based parallelism technique carried out at the frame level of the decoding process. Section 3 explains the task based parallelism applied at macroblock level and its implementation details. Section 4 presents the energy consumption and performance measurements of the two approaches implemented in this work. The overall design features and outcomes are discussed in Section 5. The ongoing research being carried out to achieve better performance and lower energy requirements for the SVC extension of the H.264/AVC video decoder are summarized in Section 6.

2. FRAME LEVEL PARALLELISM

In this approach, the YUV format of the original video sequence is split into *n* threads. This splitting pattern allows load balancing over threads that perform the decoding process later on. Each

[1] Koç University College of Engineering 34450 Sarıyer Istanbul

[2] Türk Telekom Group R&D Directorate İTÜ Ayazağa Kampüsü Maslak 34469 Istanbul

individual split YUV sequence is encoded using the MPEG-4, AVC/H.264 or SVC standard and decoded separately on different cores simultaneously as depicted in Figure 1.

During the decoding process, a size 30 GOP buffer is used. Each independently decodable stream is merged into one video sequence inside the frame buffer during the course of decoding. When the buffer is full, the threads are put to sleep for one second since there is no need to decode more frames until the previous ones are played. This method allows the decoding process to slow down whenever it goes over the speed of the actual display rate. Therefore, it avoids unnecessary CPU usage allowing further decrease in the overall energy consumption. This approach trades off increased parallelism with the encoding efficiency because consecutive frames are distributed over different cores. On the other hand, if frame level parallelism is ordered according to the levels of hierarchy in hierarchical B-pictures encoding, then the encoding efficiency will not be affected at the cost of slightly reduced parallelism.

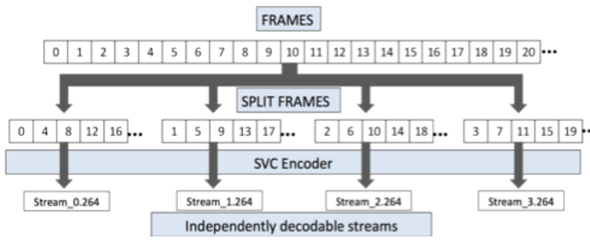


Figure 1: Splitting the original video in multiple threads to be decoded individually with four threads

3. MACROBLOCK LEVEL PARALLELISM

H.264/AVC performs block-based video coding approach in which frames are partitioned into rectangular areas, known as macroblocks (MB). The size of a MB is 16x16 pixels for a/the luma layer and 8x8 chroma layers for source sequences in 4:2:0 YUV format. A MB is either spatially or temporally predicted depending on the type of the frame [11]. MBs of predictive (P) or bi-direction predictive (B) frames can be both spatially or temporally predicted whereas the prediction for MBs in intra coded (I) frames is restricted to spatial prediction.

A certain decoding order was applied for spatially predicted MBs, as depicted in Figure 2. Since encoding is performed in raster scan order, MBs can be decoded in the same order. However, it is possible to decode the MBs in a different order as long as all the dependent MBs are decoded prior to the current MB. Note that if MBs are temporally predicted, there are no such restrictions for the decoding order of the remaining MBs.

The dependency hierarchy enables decoding of multiple MBs simultaneously. One such possibility is depicted in Figure 3(a), revealing that if MBs with decoding order 1 and 2 are processed then two MBs (numbered as 3) can be decoded simultaneously. Note that the number of MBs that can be processed in parallel increases in the later stages of the decoding process.

In contrast to the frame level parallelism (described in Section 2), macroblock level parallelism is a lot finer grain and requires considerable modifications on the decoder source code. The Intel Thread Building Blocks (TBB) library was chosen to implement the parallel algorithm [12]. Due to the dependencies shown in Figure 2 left, top-left, top and top-right MBs should be fully decoded before the current MB can be started. The decoding of MBs can be represented as a directed acyclic graph with each node of the graph representing the corresponding decoding of that MB by one processor.

Decoding of each MB is considered as a task and the numbers in Figure 3(b) represents the number of references required to start processing that task. Consequently, the upper left most MB can be initiated as the first MB. Since it is the only MB with no requirements, its reference count is set to zero. Upon completion of a task, the reference count of the successor MB(s), which are represented with the arrows in Figure 3(a), is decremented. Thus, once the first MB is decoded, the second task (the MB next to it) becomes available. Likewise, upon completion of the second MB, reference count of two successor MBs is decremented making them available parallel decoding. The process continues until all MBs are decoded in that frame.

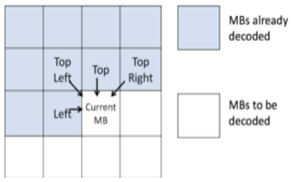
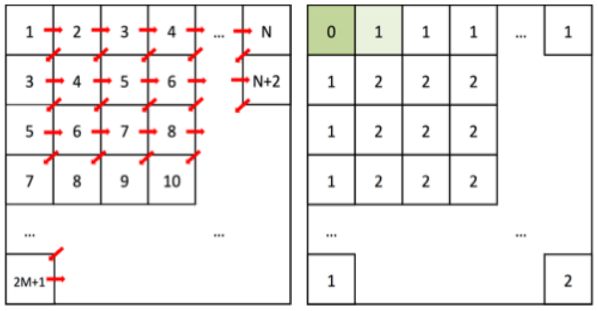


Figure 2: Spatial dependencies between neighboring MBs



a) b)
Figure 3: For a frame with MxN (width x height in MB)
a) Decoding order for MBs and their successor(s)
b) Number of references for MBs

4. MEASUREMENTS AND RESULTS

4.1 Evaluation Platform and Benchmarks

The performance and energy consumption measurement tests were carried out on a laptop running on Intel® Core^{TM} i7 720-QM quad-core processor at 1.60 GHz with 6M cache. The Enhanced Intel Speedstep® Technology (EIST), Turbo Boost Technology and the Intel® Hyper-Threading Technology offered with this processor provide the adjustability on the processor performance to observe the changes in the energy consumption for a given task.

Tests were conducted on input videos Iceberg (video with still background and moving camera), Race (video with fast-moving objects and moving camera) and Rena (video with still camera and background with moving figure) for the frame level parallelism approach, and on input videos Adile (animation video with still back ground and slow moving objects), Flower (video with moving figures and camera) and Train (video with fast-moving objects with still camera and background) for the MB level parallelism approach.

In order to measure the overall energy consumption of the whole device, its instantaneous outlet power consumption was measured over the time span while the decoding takes place by making use of a commercially available power meter called WattsUp PRO power meter [13]. The timing measurements were carried out by the tick_count class of the Intel TBB library.

4.2 Frame Level Parallelism Results

The three input videos Iceberg, Race and Rena were encoded from *n* split video sequences using the MVC mode with a quantization parameter of 22 and frame rate of 30 fps. The power consumption rate measurements were carried out by a power meter over the course of the whole decoding process as the decoded frames were being played at the same time.

The average idle power consumption rate of the laptop was taken to be 50 watts throughout the measurements. Figure 4 shows net power consumption rates per number of threads that are calculated by subtracting the idle power from the average instantaneous power consumption rates throughout the decoding process.

This approach trades off increased parallelism with the encoding efficiency since consecutive frames of the original video sequence need to be distributed over different cores. However, this approach offers better load balancing among threads. Since similar frames are decoded over different cores, each core gets a balanced amount of tasks bringing a more efficient parallelism approach. The energy measurement results indicate that with 8 threads 20% energy efficiency can be achieved using this technique and elapsed times for the complete decoding process decrease considerably.

To observe the true effect of parallelism, the elapsed times included only the time span for the decoding portion of the whole process excluding the frame buffer storage time and display times. Speedups computed for decoder running on 2, 4 and 8 threads are shown in Figure 5.

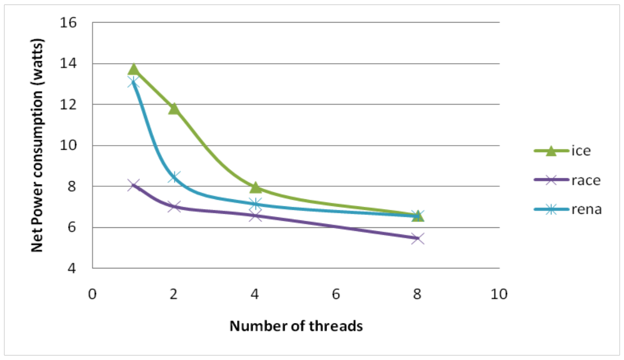


Figure 4: Net instantaneous power consumption rates in frame level parallelism

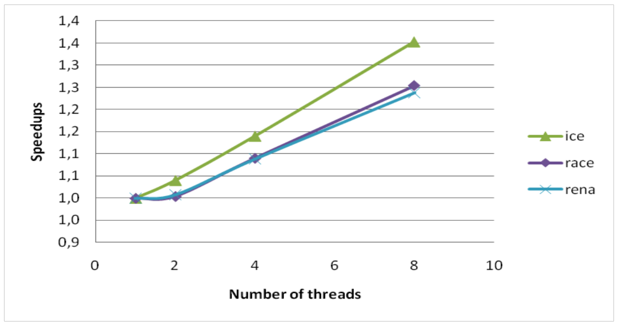


Figure 5: Speedups achieved in frame level parallelism

4.3 Macroblock Level Parallelism Results

The three input videos Adile, Flower and Train were encoded using the SVC extension of the H.264/AVC encoder with base and enhancement layers. The frame rate was set to 30 fps and quantization parameters of level 0 and 1 were 46 and 34 respectively. Figure 6 shows the power consumption of the decoding process measured using a power meter when the player is switched off.

The macroblock level parallelism approach offered a relatively smaller energy efficiency difference compared to frame level parallelism since the overall effect of the region parallelized in decoding of MBs had a minor impact on the overall decoding performance. When compared to previous work carried out in MB level parallel decoding methods our speedups are consistent with the static scheduling average speedup results up to 8 cores presented in [6] and [7]. This resulted in a smaller improvement on percentage changes of the energy consumption rates of the decoder. Our ongoing research to improve this approach is explained in further detail in section 6.

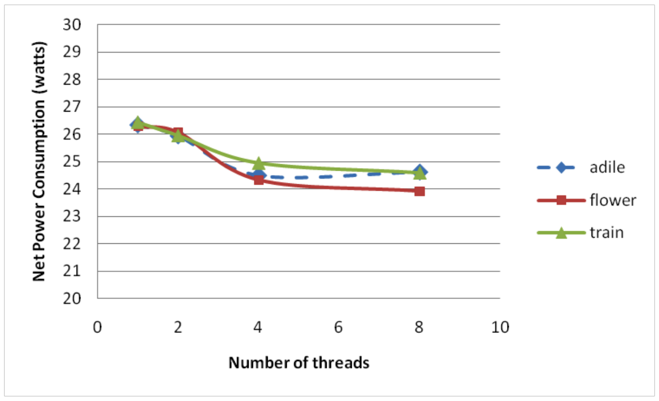


Figure 6: Net instantaneous power consumption rates in MB level parallelism

The timing measurement sets carried out in this section observe the performance change introduced with parallel MB decoding. Speedups calculated with respect to the original Open SVC decoder running of sequential MB decoding algorithm for 2, 4 and 8 threads are shown in Figure 7.

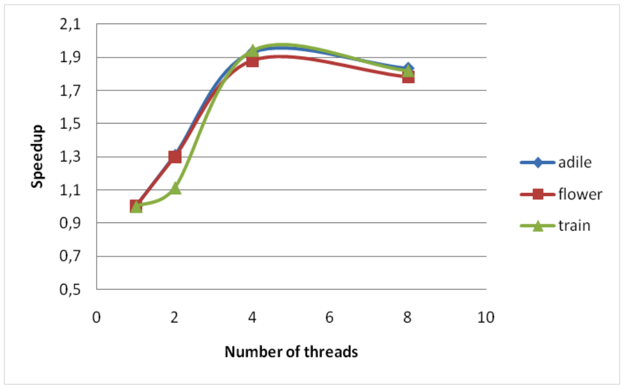


Figure 7: Speedups achieved in MB level parallelism

5. DISCUSSION

Frame level and macroblock level parallelization techniques described in this work are at the two extremes of granularity scale of parallelism for video decoding. The Frame level approach is based more on data-level parallelism which is coarse-grained and relatively simpler compared to other task-based parallelism techniques. On the other hand, the macroblock level parallelism approach is a lot finer grain and thus requires more complicated algorithms to achieve similar or better performance speed-up and energy savings.

Parallelization of video decoders at the software level requires careful synchronization between the dependent tasks in a video decoder. Most of the time, high level implementations are not sufficient and major modifications on the original video decoder source code are required. The key strategy for determining the regions to parallelize inside the decoder relies on careful assessment of the function declarations and the execution flow of the program.

A careful optimization between the choice of level at which parallelism will be applied and the overhead caused after it, can lead to great impacts both in terms of performance and energy efficiency in video decoders. Since video applications have become a major part of the global Internet traffic, developing better performing and more energy efficient video decoders will lead to a more fulfilling playback experience at the user-end, longer battery life sustained on mobile devices and, most importantly, massive amounts of energy saving on the global video traffic per user each year.

6. FUTURE WORK

Our ongoing research involves enhancements on the parallelization techniques analyzed in this paper as well as parallelization on different levels of the Open SVC H.264 video decoding standard. Further enhancements on the macroblock level of parallelism include parallelizing B-frames' macroblock decoding functionality. Since B- frames are not intra-predictable frames, the dependencies of MBs on other frames will be more challenging than parallel MB decoding on I-frames only. This feature will surely bring additional performance and energy efficiency when implemented with low overhead and careful load-balancing. Moving on to higher levels from the macroblock decoding region, we aim to look into the slice level parallelism that is parallel decoding of independent slices within each frame. Various challenges may occur when synchronizing the use of shared variables while decoding several slices at a time.

Another alternative approach is the GOP level parallelism that corresponds to the hierarchical parallelization of consecutive GOPs by exploiting the fact that each GOP consists of independently decodable I-frames and consecutive GOP's I-frames can be decoded at the same time on different cores. However, since the arrival order of the decoded frames to the frame buffer may vary when running on multiple cores, this method will bring some challenges on the management of the frame buffer of the H.264 decoder standard.

7. ACKNOWLEDGMENTS

This work was partially supported by the COST (European Cooperation in Science and Technology) framework, under Action IC0804, and by TUBITAK (The Scientific and Technical Research Council of Turkey) under Grant 109M761. A. Murat Tekalp also acknowledges support from Turkish Academy of Sciences (TUBA).

8. REFERENCES

[1] Cisco Visual Networking Index: Forecast and Methodology, 2009-2014, White paper, CISCO, June 2, 2010, [Online] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[2] Gurler C.G., Aksay A., Akar G.B., Tekalp A.M., 2010, Architectures for multi-threaded MVC-compliant multi-view video decoding and benchmark tests, *Signal Processing: Image Communication*, 25(5), pp. 325-334

[3] Geer D., 2005, Industry trends: Chip makers turn to multicoreprocessors, *Computer*, 38(5), pp. 11-13.

[4] Rintaluoma, T., Silven, O., Energy efficiency of mobile video decoding, 2007, *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, IC-SAMOS*, pp.103-109.

[5] Bilas, A., Fritts, J., Pal J., and Paper, S., 1997, Real-Time Parallel MPEG-2 Decoding in Software, *11th International Parallel Processing Symposium (IPPS).*

[6] Jike C., Satish, N., Catanzaro, B., Ravindran, K. Keutzer, K., 2007, Efficient Parallelization of H.264 Decoding with Macro Block Level Scheduling, *IEEE International Conference on Multimedia and Expo*, pp.1874-1877.

[7] Alvarez, M., Ramirez, A., Martorell, X., Ayguade, E., and Valero, M., 2008, Scalability of Macroblock-level Parallelism for H.264 Decoding, In ACACES.

[8] C.H. Meenderinck, A. Azevedo, M. Alvarez, B.H.H. Juurlink, A. Ramirez. 2008. Parallel Scalability of H.264, in: *Proceedings of the first Workshop on Programmability Issues for Multi-Core Computers*, Geteborg, Sweden.

[9] Rodriguez, A.; Gonzalez, A.; Malumbres, M.P., 2006. Hierarchical Parallelization of an H.264/AVC Video Encoder Parallel Computing in Electrical Engineering, PAR ELEC, pp.363-368.

[10] Finchelstein D.F., 2009, *Low-Power Techniques for Video Decoding*, Doctoral Thesis, MIT.

[11] Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A., 2003. Overview of the H.264/AVC video coding standard *IEEE Transactions on Circuits and Systems for Video Technology,* 13(7), pp.560-576.

[12] Intel ® Threading Building Blocks Tutorial [Online] http://www.threadingbuildingblocks.org/documentation.php

[13] WattsUp Products. [Online] https://www.wattsupmeters.com/secure/products.php?pn=0