Performance and Energy Consumption Analysis of the X265 Video Encoder

Dieison Silveira[1,3], Marcelo Porto[2] and Sergio Bampi[1]
[1]Federal University of Rio Grande do Sul - INF-UFRGS - Graduate Program in Microelectronics (PGMicro) - Brazil
[2]Federal University of Pelotas - CDTec-UFPEL - Graduate Program in Computing (PPGC) - Brazil
[3]Federal Institute of Rio Grande do Sul - Campus Canoas (IFRS) - Brazil
Email: dssilveira@inf.ufrgs.br, porto@inf.ufpel.edu.br, bampi@inf.ufrgs.br

***Abstract*—The x265 video encoder aims at improving the speed and the computational efficiency of HEVC encoders implementation. In this paper we present a detailed energy consumption analysis, considering the consumption components of CPU, cache memories and main memory, for all x265 presets executing in a multicore system. Ten HD 1080p test video sequences with different motion and brightness characteristics are used in the experiments. Three tools are used to obtain the results: CACTI, PCM and Perf. To get more reliable time/energy results, 10 executions were performed for each preset. The results show that fast presets are $47\times$ faster than slower presets. However, slower presets use robust configurations and achieve large reductions in bitrate. Due to this, the *ultrafast* preset has a bitrate 45% higher than *placebo* preset. Furthermore, the system energy consumption increases $45\times$, from *ultrafast* preset to *placebo* preset. Our experiments clearly present the dependence between bitrate and energy consumption for all encoding presets, which allows us to choose the best bitrate/energy trade-off for each platform at hand.**

***Index Terms*—Video encoding, HEVC, x265, energy consumption, multicore system**

I. Introduction

Nowadays the innovations in multimedia devices introduced an increasing demand for videos with high resolutions and better quality. Nonetheless, such videos require a large amount of data for representation, storage, and eventual transmission. As a consequence, video coding represents a key challenge to make multimedia support a feasible task for current systems. The video coding process incorporates many tools and techniques that are constantly in development, thus there is an intense research activity in this field. With these techniques, digital videos are represented with much smaller volume of data at the cost of heavy processing and some losses in visual quality.

These tools and techniques are used to define video coding standards, such as the High Efficiency Video Coding (HEVC) [1], which is the most recent video compression standard, and the H.264/AVC [2], current market-dominant standard [3]. The HEVC introduces a plenty of innovations to the video processing, such as: new coding structures, larger prediction units, variable size transforms and two features to enhance parallel processing capability: tiles and WPP (Wavefront Parallel Processing) [3]. The HEVC promoted 40% bitrate reduction for similar objective video quality in relation to H.264/AVC [4].

However, the new encoding tools introduced by HEVC generated an increase in processing complexity [5], which increases processing time and energy consumption. Moreover, energy consumption is known as a crucial factor that constrain the design of many computer architectures [6]. Besides, this performance metric should be also taken into account when analyzing video encoders [7].

The HEVC is supported by Joint Collaborative Team on Video Coding (JCT-VC), and they developed a reference software for HEVC, which is called HM (HEVC Test Model). The HM provides an encoder software capable of producing bitstreams in conform to HEVC specification, and the HM is used for many researchers around the world for theoretical research and quality analysis. However, the HM presents a sequential implementation, due to this it is not suitable for performance and energy analysis in multicore processors.

In this way, some high performance HEVC encoders were developed, such as: x265 [8], an open source project, and DivX265 [9], a commercial encoder. These encoders can be 1000 times faster than HM [10]. This is possible because such encoders use the parallelism tools provided by the HEVC standard. Thus, they present a great speed up on multicore processors.

There are some works that investigated performance and energy consumption considering the high performance encoders and the reference software. The authors in [7] evaluate four video encoders, x264, HM, VP8 and VP9 (Google's video encoders), studying execution time, encoding efficiency, scalability and total energy consumption. The results were obtained using five HD 1080p video sequences. Following this methodology, Hu et al. [10] present the performance results in terms of execution time and quality for three video encoders: x265, HM and x264. However, they did not present the energy consumption results for these encoders.

Huangyuan et al. present in [11] a comparison between two implementations of the HEVC: x265 and DivX265. They evaluate the rate distortion performance and execution time for seven UHD 4K video sequences. Their experiments showed that the optimized HEVC encoders achieve best performance when compared to HEVC reference software, the HM. These same results were achieved by others authors in [12], however, the results were achieved with low resolution videos (720x576).

ISBN 978-0-9928626-7-1

The authors in [13] present an energy comparison between two video encoders: HM (HEVC) and JM (H.264). They evaluate different coding configuration for both encoders, and the total energy results were obtained using the RAPL tool. The results showed an improvement of 25% in coding efficient from HEVC reference software, HM, to H.264 reference software, JM. However, the HM consumes 17% more energy than JM, as expected.

The aforementioned studies present significant and insightful information about video encoder. However, these papers focus in performance, lacking in detailed system energy consumption analysis, including CPU and memories system. In addition, some studies have not analyzed all available encoder settings, for example the x265, which has ten predefined presets. The preset options optimize the trade-off between encoding speed and compression efficiency. Thus, this work presents a detailed performance and energy consumption analysis of the x265 video encoder, considering the energy consumption related to CPU, main memory and caches.

In this paper, the 10 presets of the x265 encoder [8] are evaluated with 10 HD 1080p test sequences in a multicore system. Three profiling tools are used to obtain the results: Perf Linux [14], Intel PCM [15], and HP CACTI [16]. The Perf profiler collects the CPU statistics and monitors the hardware counters. The PCM provides the energy consumption results for main memory (DRAM memory) and CPU, and CACTI estimates the energy consumption per access of the cache memories.

The paper is organized as follows: Section II describes the analysis methodology, Section III presents the results and discussion and Section IV concludes the paper.

II. Methodology

This section presents the methodology used, which includes the description of the experimental setup, the test sequences, the profiling tools, and the model for energy consumption proposed and utilized in our measurements.

A. Experimental Setup

The 10 x265 presets are evaluated: ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow and placebo. These presets employ different coding features and its combinations, such as motion estimation algorithms, number of reference frames, CTU size, among others. The complete list can be found on the x265 documentation [8]. When faster presets are used, x265 takes shortcuts to improve performance at the expense of quality and compression efficiency. When slower presets are used, the encoder tests more encoding options, using more computations to achieve the lowest bit rate at the selected quality.

The 10 HD 1080p (1920×1080 pixels) test sequences used in the experiments present different motion and brightness characteristics; thus, a wide variation of different decisions must be done by the encoder. The HD 1080p test sequences used are: BasketballDrive (500 frames), BQTerrace (600 frames), Cactus (500 frames), Kimono (240 frames), ParkScene (240 frames), Pedestrian Area (375 frames), Tennis (240 frames), Rush Hour (500 frames), Sunflower (500 frames) and Traffic (480 frames).

TABLE I
Memory specification for CACTI simulation

Parameters	Cache L1	Cache L2	Cache L3
Capacity	$32KB$	$256KB$	$8MB$
Block size	$64B$	$64B$	$64B$
Associativity	8	8	16
Technology	$32nm$	$32nm$	$32nm$
Banks	1	1	1
Model	UCA	UCA	$NUCA$
Dynamic Energy	$0.0164nJ$	$0.0731nJ$	$0.332nJ$
Static Power	$0,011W$	$0,085W$	$0,344W$

The workstation used in the experiments is a Xeon E3-1271 processor, with four cores running at 3.60 GHz. Each core has 64 KB of L1 cache (32 KB data and 32 KB instruction), 256 KB of L2 cache and 8 MB of L3 cache. The external memory system is composed by two 16GB DDR3 memory modules (DIMM). In the experiment, 10 executions were performed for each preset. These executions were done to get more reliable results since the time and energy consumption vary with the operating system and other process running concurrently on the machine.

B. Energy Profiling Tools

Three profiling tools are used to obtain the energy measurements: 1) Intel Performance Counter Monitor (PCM) [15], 2) Perf Linux [14], and 3) HP CACTI [16].

The Intel PCM is a profiling tool that provides energy consumption results of any application that is executed in recent architectures, such as Intel Xeon, Sandybridge and Ivy Bridge processors [15]. If a parallel HPC application is executed in various sockets, PCM will output CPU energy, Dynamic Random Access Memory (DRAM) energy, NUMA details, performance flaws, and so forth in various formats to end users [15]. The tool considers Machine Specific Registers (MSR) using RAPL counters to disclose the energy consumption details of the application.

Perf is a profiling tool for Linux-based systems that abstracts away CPU hardware differences in Linux performance measurements [14]. The Perf tool is based on the perf_events interface exported by recent versions of the Linux kernel.

CACTI is an integrated cache and memory access time, area, leakage, and dynamic power model [16]. The CACTI cache access model takes in the following major parameters as input: capacity, block size, associativity, technology generation, the number of ports, and the number of independent banks. Table I presents the main input for CACTI simulation and the output results for that specification.

C. Cache Energy Consumption Measurement

The amount of cache loads, stores, load misses, and store misses are generated by the Perf tool. However, this tool does

ISBN 978-0-9928626-7-1 © EURASIP 2017

TABLE II
AVERAGE X265 PRESETS RESULTS FOR ALL VIDEO SEQUENCES

Presets	**Runtime** (FPS)	**Bitrate** (KB/s)	**PSNR** (dB)	**CPU** (J)	**DRAM** (J)	**Cache** (J)
Ultrafast	22.1	2764	37.8	6.29	0.28	0.050
Superfast	18.8	2621	37.9	7.50	0.33	0.058
Veryfast	17.9	2463	37.5	7.80	0.35	0.061
Faster	15.6	2460	37.6	8.87	0.40	0.068
Fast	12.6	2111	37.4	10.58	0.48	0.082
Medium	8.4	2059	37.5	16.67	0.72	0.130
Slow	3.5	2027	37.7	41.46	1.71	0.325
Slower	0.9	1952	37.8	159.06	6.36	1.267
Veryslow	0.7	1929	37.8	209.57	8.57	1.656
Placebo	0.4	1904	37.7	291.17	12.57	2.288

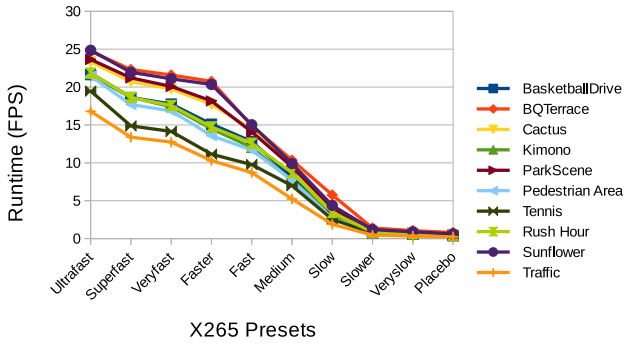

Fig. 1. Runtime results, in FPS, for all x265 presets

not provide events for intermediate levels of cache, such as L2. To estimate the L2 events it was used the results obtained from L1 and L3, where L1 misses represents the L2 total accesses and L3 total accesses represents the L2 misses. The energy per access used in the next equations was obtained with CACTI and they are presented in Table I.

Equation 1 presents the method used to estimate the energy consumption for write operations, where the $Miss_{wtCL}$ is the amount of write miss accesses to cache level, E_ac_{wtCL} is the energy consumption per write access to a cache level, and E_ac_{rdCL} is the energy consumption per read access to a cache level. Hit_{wtCL} is the amount of write hit accesses to cache level (i.e. L1, L2 or L3).

Equation 2 presents the method used to estimate the energy consumption for read operations and follows the same idea aforementioned to calculate the energy for write operations. Both equations 1 and 2 are used to estimate the cache energy consumption for the L1, L2, and L3 cache levels.

Equation 3 represents the total cache energy consumption calculated by the sum of the energy consumed (dynamic and static) in the three levels of the cache memory. The dynamic energy consumption is presented by equations 1 and 2. And the static energy consumption is represented by the static power dissipated ($Power_{Static}$) over the time ($Time$), where static power dissipation is the power that is lost while circuit signals are not actively switching, such effect is also called leakage power.

$$EC_{Cache_{Wr}} = (Hit_{wtCL} \times E\_ac_{wtCL}) + (Miss_{wtCL} \times (E\_ac_{rdCL} + E\_ac_{wtCL})) \quad (1)$$

$$EC_{Cache_{Re}} = (Hit_{rdCL} \times E\_ac_{rdCL}) + (Miss_{rdCL} \times (E\_ac_{rdCL} + E\_ac_{wtCL})) \quad (2)$$

$$TotalEC_{Cache} = EL1_{rd} + EL1_{wt} + EL2_{rd} + EL2_{wt} + EL3_{rd} + EL3_{wt} + Power_{Static} \times Time \quad (3)$$

III. RESULTS AND DISCUSSIONS

This section presents and discuss the performance results, the energy consumption results for DRAM and CPU, and the component of cache memory energy consumption.

Table II presents the performance and energy consumption results for the experiment. The results correspond the average value of the 10 runs for each test sequence. In this table, the energy consumption results for CPU, DRAM and cache memories are related to one encoded frame. The average runtime is presented in terms of frames per second (FPS). As can be seen in Table II the *ultrafast* preset presents the highest runtime among all presets, 22 FPS for the test set. However, the *placebo* preset reaches 0.4 FPS, i.e. $47\times$ less than *ultrafast* preset. A detailed version of the runtime (FPS) results is presented in Figure 1. Where each point in this figure corresponds the average value of the 10 runs for each test sequence.

The *ultrafast* preset uses a maximum CU size of the 32x32 pixels. Due to this, this preset profits from a higher level of parallelism with more rows of CUs encoded in parallel. However, the larger the maximum CU size, the more efficiently x265 can encode flat areas of the frame, resulting in large reductions in bitrate. Because of this the *ultrafast* preset presents a bitrate 45% higher than *placebo* preset for the same quality.

The energy consumption results presented in Table II show that when slower presets are used, i.e. x265 tests more encoding options, more computations are performed to achieve the results and more energy is consumed, as expected. The system energy consumption increases $45\times$, from *ultrafast* preset to *placebo* preset. In this way, for every 1% reduction in bitrate there is a 145% increase in energy consumption.

This increase in energy consumption happens because some parameters are modified when the presets change, such as: maximum amount of consecutive bi-predictive frames (b-frames), increase in the transform unit (TU) depth and the substitution of fast algorithms (hexagon-pattern search) by slower algorithms (star-pattern search) in the motion estimation. The star algorithm is a three step search adapted from the HM encoder: a star-pattern search followed by

ISBN 978-0-9928626-7-1 © EURASIP 2017

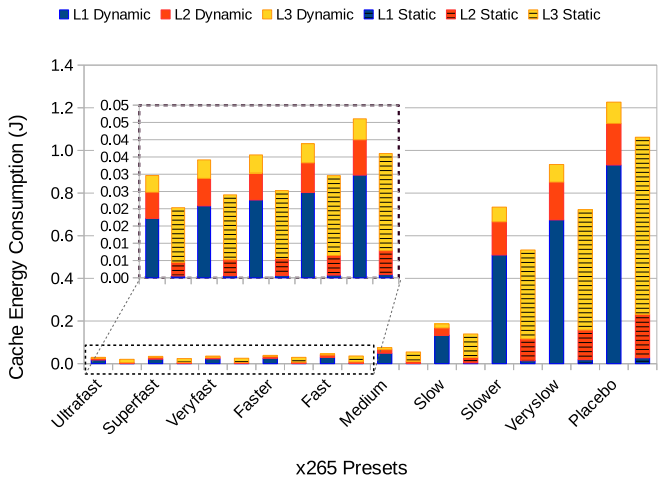

Fig. 2. Dynamic and static energy consumption results for the cache memories for all x265 presets

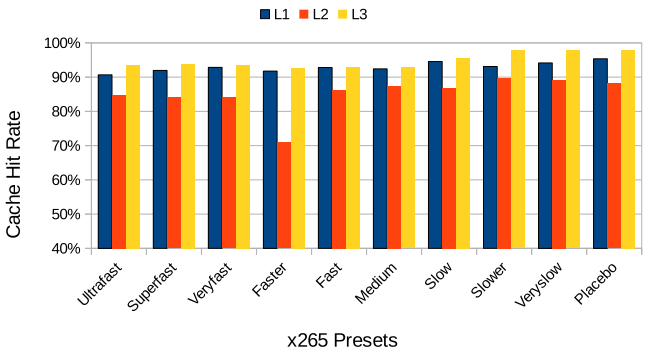

Fig. 3. L1, L2 and L3 cache hit results for all x265 presets

an optional radix scan followed by an optional star-search refinement. The number of b-frames has a quadratic effect on the amount of memory allocated and in the effort in determining the b-frames placement. These changes can be observed from *slow* preset.

It is possible to see in Table II that for all presets the energy consumption of the CPU is greater than the energy consumed by the memories. The energy consumption maintain the same behavior, since these three components are also affected by the preset settings. Table II also shows that the difference among the energy consumption from CPU, DRAM and cache memories remain the same, independent of the chosen preset.

From the energy consumption results is possible to observe the participation of each part of the system, CPU, DRAM and cache, in the total energy consumption. The CPU energy consumption represents 95% of the total energy consumption, DRAM represents 3%, and the cache memories represent 2% of the total energy consumption.

To obtain the energy consumption for the cache memories, it was necessary to estimate the consumption by access of each level of the cache. The estimated cache results results were obtained following the Equations 1, 2 and 3 presented in Section II with Perf and CACTI tools. The Perf tool uses the hardware counters to generate the amount of accesses and the CACTI generates the energy consumption per access.

Figure 2 presents a detailed energy consumption (dynamic and static) results per frame of the cache memories. It can be seen in this figure that the dynamic energy is higher than static energy for all presets. This is the case since the video encoder uses the cache memory efficiently, achieving a high hit rate.

As can be seen in Figure 2 the dynamic energy consumption represents 55%, on average, of the total cache energy consumption for all presets. From this value, the L1 cache accounts for 72% of the dynamic energy consumption, the L2 cache has 18% and the L3 cache has 10%. For the static energy consumption these values are different, L3 consumes the highest static energy with 78%, L2 has 19% and L1 has 3%. This is due to the high static power presented by the L3 cache, which is $4\times$ higher than L2 and $31\times$ higher than L1, since the L3 cache has a higher density than L1 and L2 cache.

These results could be confirmed by the average read and write accesses for all presets, where L1 is accessed 94.2% of the time, L2 5.2% and L3 only 0.6%. In addition, Figure 3 presents the hit rate results for the three levels of the cache memory. These results were obtained from the hardware counters using the Perf tool. It can be seen in this figure that hit rate is high for all levels of the cache and increases when slower presets are used, thus the miss rate decreases. In L1 cache the hit rate varies from 91% for the *ultrafast* preset up to 96% in the *placebo* preset, reaching 94.5% in average for all presets. The remaining cache levels have an average hit rate of 88.2% to L2 and 97% to L3.

Figure 4 shows the energy efficiency considering the energy consumption per encoded byte. It is possible to note that the energy consumption per byte significantly varies among the videos. This is due to the fact that video sequences present distinct texture and motion characteristics with different amount of homogeneous/textured areas. So, the encoder employs greater computational effort to efficiently encode videos with many textured areas, especially for high motion activity scenes. This behavior can be observed in sequences Sunflower and Traffic, which present high compression ratio and have high energy consumption.

Figure 5 presents the behavior of bitrate and energy consumption for all presets. It is then possible to conclude that the fast, medium and slow presets present the best bitrate/energy trade-offs. Since these three presets have the smallest values of bitrate and energy consumption among all evaluated presets. In this way, if the device has bitrate and energy constraints these three presets are good alternatives.

However, for energy-constrained devices, such as battery-powered embedded systems, a good alternative is the use of fastest presets, which have the lowest energy consumption. Otherwise, for high-resolution videos such as UHD 4K and 8K, slower presets are required, since these

ISBN 978-0-9928626-7-1 © EURASIP 2017

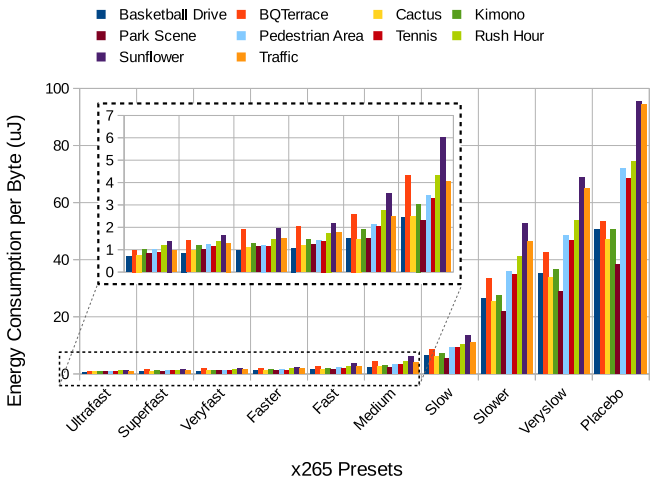

Fig. 4. Energy consumption per byte (uJ) encoded for all x265 presets

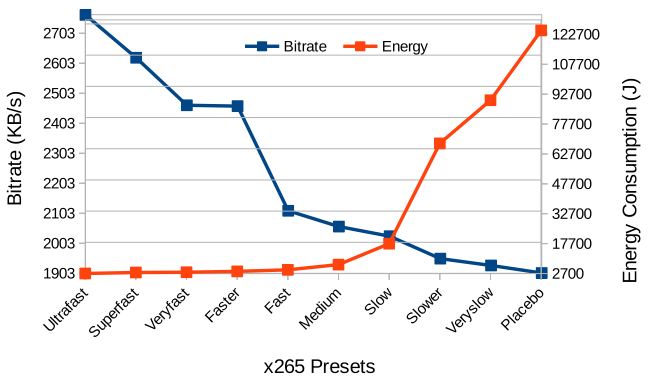

Fig. 5. Bitrate results versus energy consumption results for all x265 presets

presets achieve best compression ratio, and these videos require a large amount of data to be represented and stored.

IV. Conclusion

The performance and energy consumption of the CPU and memory system for all x265 presets in a parallel system were evaluated in this paper. Ten HD 1080p test sequences were used in the experiment and three profiling tools were used to generates the results: PCM, Perf, and CACTI. The results showed that fast presets can be $47\times$ faster than slower presets. However, slower presets use robust configurations and can achieve large reductions in bitrate. Due to this the *ultrafast* preset has a bitrate 45% higher than *placebo* preset. The results also showed that the system energy consumption increases $45\times$, from *ultrafast* preset to *placebo* preset. This means that for every 1% reduction in bitrate there is a 145% increase in energy consumption. This occurs because slower presets tests more encoding options and more computations are performed to achieve the results, spending more energy. In addition, it was possible to measure the energy consumption of each part of the system, where the CPU energy consumption represents 95% of the total energy consumption, the DRAM represents 3%, and the cache memories represent 2% of the total energy consumption for the test sequences.

As future work we will investigate and compare the performance and energy consumption of the CPU and memory systems for others video encoders, such as DivX265, VP8 and VP9. Such work will consider multithreaded high performance systems.

Acknowledgements

This work was partially financed by the National Council for Scientific and Technological Development (CNPq), Coordination of Improvement of Superior Education Staff (CAPES), and Research Support Foundation of Rio Grande do Sul (FAPERGS).

References

[1] H.265, "ITU-T recommendation h.265: High efficiency video coding, audiovisual and multimedia systems," Tech. Rep., 2013. [Online]. Available: http://www.itu.int/rec/T-REC-H.265-201304-I

[2] H.264/AVC, "ITU-T recommendation h.264 (01/12): Advanced video coding for generic audiovisual services," Tech. Rep., 2012. [Online]. Available: http://www.itu.int/rec/T-REC-H.264-201402-P

[3] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of hevc and avc video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885–1898, 2012.

[5] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1899–1909, 2012.

[6] R. Ge, X. Feng, S. Song, H. C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 658–671, 2010.

[7] R. Sanchez, F. D. Igual, J. L. Martinez, R. Mayo, and E. S. Q. Orti, "Parallel performance and energy efficiency of modern video encoders on multithreaded architectures," *European Signal Processing Conference*, pp. 191–195, 2014.

[8] X265, "x265 software," October 2016. [Online]. Available: http://x265.readthedocs.io/en/default/index.html

[9] DivX265, "Divx hevc encoder," October 2016. [Online]. Available: http://labs.divx.com/Divx265

[10] Q. Hu, X. Zhang, Z. Gao, and J. Sun, "Analysis and optimization of x265 encoder," *Visual Communications and Image Processing*, pp. 502–505, 2014.

[11] Q. Huangyuan, L. Song, Z. Luo, X. Wang, and Y. Zhao, "Performance evaluation of h.265/mpeg-hevc encoders for 4k video sequences," *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–8, 2014.

[12] O. Zach and M. Slanina, "A comparison of h.265/hevc implementations," *International Symposium ELMAR*, pp. 1–4, 2014.

[13] E. Monteiro, M. Grellert, S. Bampi, and B. Zatt, "Rate-distortion and energy performance of hevc and h.264/avc encoders: A comparative analysis," *International Symposium on Circuits and Systems*, pp. 1278–1281, 2015.

[14] Perf, "Linux perf tool," October 2016. [Online]. Available: http://perf.wiki.kernel.org/

[15] I. PCM, "Intel performance counter monitor," October 2016. [Online]. Available: http://www.intel.com/software/pcm

[16] H. CACTI, "Cacti 6.5," October 2016. [Online]. Available: http://www.hpl.hp.com/research/cacti