UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DANIEL PALOMINO

Application-driven temperature-aware solutions for video coding

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Altamiro Susin
Co-orientador: Prof. Dr. Luciano Agostini

Porto Alegre
2017

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Palomino, Daniel
Application-driven temperature-aware solutions for video coding [manuscrito] / Daniel Palomino. – 2017.
87 f.:il.
Orientador: Altamiro Susin; Co-orientador: Luciano Agostini.
Tese (Doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2017.
1. Gerenciamento de temperatura. 2. Codificação de vídeo. 3. HEVC. 4. Conhecimento da aplicação. 5. Gradientes de temperature. 6 Plataformas de hardware. 7. Arquiteturas. 8. Circuitos integrados I. Susin, Altamiro. II. Agostini, Luciano. III. Application-driven temperature-aware solutions for video coding.

ACKNOWLEDGMENTS

Em primeiro lugar, gostaria de agradecer aos cidadãos brasileiros que financiaram o meu curso de doutorado através da universidade pública e gratuita e também pela bolsa de doutorado que recebi pelo período de dois anos. Vou me esforçar para retribuir essa grande oportunidade dentro da própria universidade pública, agora como docente.

Gostaria de agradecer também a todos os amigos e conhecidos que facilitaram a minha vida nesses quatro anos. Em especial àqueles que, com caronas, facilitavam meu deslocamento entre Pelotas e Porto Alegre.

Gostaria de agradecer a todo apoio que tive da minha família, em especial os meus pais, que foram os grandes patrocinadores dessa oportunidade, não só financeiramente, mas também psicologicamente.

Um agradecimento especial à minha esposa, Carolina, pelo incentivo e pela paciência nos momentos mais difíceis. O carinho dela foi de suma importância para que eu conseguisse atravessar o doutorado inteiro (que não é fácil) e chegar até aqui.

Agradeço também ao grupo de colegas e amigos (Cláudio Diniz, Felipe Sampaio, Eduarda Monteiro, Cauane Silva, Leonardo Soares, Matheus Grellert) que dividiram o laboratório 215 do INF comigo durante o período do curso e ao professor Sérgio Bampi que, mesmo não sendo meu orientador formalmente, sempre me apoiou bastante. Além disso, um agradecimento ao pessoal do *Chair for Embedded Systems* – CES do KIT em Karlsruhe na Alemanha por ter me recebido para o estágio sanduíche em 2013, em especial ao professor Jörg Henkel, o doutor Muhammad Shafique e o doutor Hussam Amrouch por terem contribuído diretamente para minha tese.

Finalmente, um agradecimento aos meus professores orientadores. O co-orientador Luciano Agostini, que foi o primeiro a me oportunizar o trabalho com pesquisa científica, ainda nos tempos de graduação na UFPel e está junto na caminhada desde lá. O orientador Altamiro Susin, que foi meu orientador também de mestrado e me deu a oportunidade e a confiança de terminar esse curso de doutorado em uma universidade de qualidade como a UFRGS.

ABSTRACT

This thesis presents application-driven temperature-aware solutions for next generation video coding systems, such as the High Efficiency Video Coding (HEVC). Different from state-of-the-art works, the proposed solutions raise the abstraction of temperature management to the application-level, where video coding characteristics and video content properties are used to leverage thermal-aware solutions for video coding with low QoS (Quality of Service) degradation. Several video coding and temperature analyses are performed to understand the behavior of temperature when encoding different video sequences. Based on the analyses results, different approaches are proposed to mitigate the temperature effects on video coding systems. Application-driven temperature management for HEVC uses run-time encoder configuration selection to keep temperature under safe operational state while providing good visual quality results. Temperature optimization using approximate computing uses content-driven approximations to reduce the on-chip temperature of HEVC encoding. Application-driven temperature-aware scheduler leverages application-specific knowledge to guide a scheduling technique targeting reducing the spatial temperature gradients that are resulted from the unbalance workload nature of multi-threaded video coding application. The proposed solutions are able to provide up to 10 °C of chip temperature reduction with negligible compression efficiency loss. Besides, when compared with previous works the resulted objective video quality (PSNR) is from 12 dB up to 20 dB higher. Moreover, the proposed scheduler eliminates spatial temperature gradients greater than 5 ºC of multi-core architectures. As conclusion, this thesis demonstrates that leveraging application-specific knowledge and video content properties has a significant potential to improve temperature profiles of video coding systems while still keeping good quality results.

Keywords: Temperature management, video coding, HEVC, application-driven, temperature-aware, application knowledge, temperature gradients, hardware platforms, architectures, integrated circuits.

Soluções para o gerenciamento de temperatura de sistemas de codificação de vídeo

RESUMO

Esta tese apresenta soluções para o gerenciamento e otimização de temperatura para sistemas de codificação de vídeo baseados nas características da aplicação e no conteúdo dos vídeos digitais. Diferente dos trabalhos estado-da-arte, as soluções propostas nesta tese focam em técnicas de gerenciamento de temperatura no nível da aplicação e características da aplicação codificação de vídeo e as propriedades dos vídeos digitais são explorados para desenvolver soluções termais para a codificação de vídeo com baixas perdas na qualidade de serviço das aplicações. Diversas análises são realizadas considerando a aplicação de codificação de vídeo para entender o comportamento da temperatura durante o processo de codificação para diferentes sequências de vídeo. Com base nos resultados das análises, soluções com diferentes abordagens são propostas para atenuar os efeitos da temperatura nos sistemas de codificação de vídeo. Gerenciamento de temperatura baseado nas características da aplicação para o padrão de codificação HEVC usa uma técnica de seleção de configuração em tempo de execução para manter a temperatura abaixo dos limites seguros de operação com bons resultados de qualidade de vídeo. Otimização de temperatura baseado em computação imprecisa usa aproximações baseadas em conteúdo para reduzir a temperatura de chips executando o HEVC. Um escalonador de tarefas que usa características da aplicação para guiar o escalonamento de *threads* focando na redução dos gradientes espaciais de temperatura que são resultantes do desbalanceamento natural de cargas entre as *threads* da aplicação. As soluções propostas são capazes de reduzir em até 10 ºC a temperatura do chip com perdas insignificantes na eficiência de compressão. Os resultados de qualidade objetiva (medida usando PSNR) são de 12 dBs até 20 dBs maiores quando comparados com trabalhos da literatura. Além disso, o escalonador de tarefas proposto é capaz de eliminar os gradientes espaciais de temperatura maiores que 5 ºC para arquitetura multi-cores. Como principal conclusão, esta tese demonstra que as técnicas de gerenciamento de temperatura que usam o conhecimento da aplicação de maneira conjunta com as propriedades dos vídeos digitais tem um alto potencial para melhorar os resultados de temperatura de sistemas de codificação de vídeo mantendo bons resultados de qualidade visual dos vídeos codificados.

Palavras-chave: Gerenciamento de temperatura, codificação de vídeo, HEVC, conhecimento da aplicação, gradientes de temperatura, plataformas de hardware, arquiteturas, circuitos integrados.

FIGURES LIST

TABLES LIST

ABREVIATION LIST

AC	Approximate Computing
AVC	Advanced Video Coding
BD-PSNR	Bjontegärd-Delta Peak Signal-Noise to Ratio
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CPU	Central Processing Unit
CTU	Coding Tree Unit
CU	Coding Unit
dB	Decibel
DCT	Discrete Cosine Transform
DTM	Dynamic Thermal Management
DTS	Digital Thermal Sensor
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
FIR	Finite Impulse Response
FS	Frequency Scaling
HCI	Hot-Carrier Injection
HD	High Definition
HEVC	High Efficiency Video Coding
IR	Infra-Red
JCT-VC	Joint Collaborative Team on Video Coding
MPEG	Moving Picture Experts Group
MPSoC	Multi Processor System on Chip
MSE	Mean-Squared Error
NBTI	Negative Bias Temperature Instability
PC	Personal Computer

PECI	Platform Environment Control Interface
PSNR	Peak Signal-Noise to Ratio
QoS	Quality of Service
QP	Quantization Parameter
RF	Reference Frames
RGB	Red Green Blue
SA	Search Area
SAD	Sum of Absolute Differences

TABLE OF CONTENTS

1 INTRODUCTION

Video services have widely spread in the consumer, communication and industrial markets in the past years. Several devices such as TVs, smartphones, tablets are able to reproduce digital videos and most of these devices also have the capability of video encoding/decoding. According to (CISCO, 2013) video based services will consume 80%-90% of the global internet traffic by 2017. High resolutions such as 4k (4096×2160 pixels) and 8k (7680×4320 pixels) and high immersion provided by Multiview and 3D technologies are rising to improve the user experience. As these new features demand much more data than previous HD videos, the efficiency of video compression plays an important role to make possible the use of such improvements in daily used applications. In this context, video compression standards such as High Efficiency Video Coding (HEVC) (ITU-T, 2013) have recently been released to provide the high compression efficiency (i.e., low bit rates with high video quality) to fulfil the demands of high-end video contents. However, the computational complexity of the video coding process has increased when compared with previous standards such as MPEG-2 and H.264 (VANNE, VIITANEN, *et al.*, 2012).

The high complexity of the HEVC can be well complemented by the high integration density due to the continuous technology shrinking of the transistor feature sizes. The transistor shrinking together with the microarchitectural advancements provided the high-performance computing systems with many processing cores, operating in parallel and at high clock frequencies. However, the expected voltage scaling did not followed the transistor shrinking due to the failure of Dennard's scaling (DENNARD, GAENSSLEN, *et al.*, 1974) (BOHR, 2007) that has resulted in high power densities (power/area) and consequently elevated on-chip temperatures (thermal hotspots).

High on-chip temperatures can lead to higher cooling costs (HENKEL, BAUER, *et al.*, 2013) which are more challenging for embedded systems since they have limited space to allow sink devices. Furthermore, thermal hotspots negatively affect the chip reliability and lifetime since most of the aging effects are aggravated at high temperatures (HENKEL, BAUER, *et al.*, 2013) (GNAD, SHAFIQUE, *et al.*, 2015). In addition to thermal hotspots, high spatial and temporal temperature variations also have a negative impact on system reliability and performance (PECHT, LALL e HAKIM, 1992).

Many computing intensive applications may be split into several threads and thus can be suitably parallel processed by multi-core systems. This is the case of image and video processing, computer graphics, image synthesis and videogames, many artificial intelligence algorithms like neural networks, data mining and control and instrumentation, to say only the most common ones. More than that, the computational complexity of some algorithms may be gracefully reduced by approximate computation to accomplish deadlines or to lower power dissipation. The management of tasks' parametrization and distribution among the processing elements is tricky because it is complex and cannot consume too much power itself.

Video processing (especially video coding) is an important application domain with respect to temperature since it is very data and compute intensive. With the new video coding standards, like HEVC, trends of Multiview and 3D videos, the computational complexity of applications dealing with this type of content increases radically, which impacts on the thermal profiles of the encoding system. Besides, the workload variations generated by the video processing applications translate to temperature variations and spatial and/or temporal thermal gradients, thus leading to thermal problems as mentioned above.

There are several literature works that deal with temperature issues focusing in all different computing stack levels. Solutions at design-time and at run-time are employed in hardware (FISHER, CHEN, *et al.*, 2011) (EBI, FARUQUE and HENKEL, 2009) (BARTOLINI, CACCIARI, *et al.*, 2013) and system (COSKUN, ROSING and GROSS, 2008) (COSKUN, ROSING, *et al.*, 2008) (KUMAR, SHANG, *et al.*, 2008) levels in order to mitigate the temperature effects. Traditionally, Dynamic Thermal Management (DTM) is discussed by literature works in order to deal with the temperature issues at run-time. The main goal of DTM techniques is to keep the system below a safe operating temperature while maintaining real-time constraints for time/performance critical applications.

In case of video coding/decoding, the quality of service (i.e., bit rate vs. quality) is sacrificed to guarantee safe temperature levels. However, it is possible to mitigate these temperature effects if application-specific characteristics and video properties are taken into account on developing thermal management techniques for video coding systems.

The main goal of this thesis is to mitigate temperature effects on video coding systems at the application level. This approach differs from most of the state-of-the-art works by raising the concept of dynamic thermal management from hardware and system levels to the application level such that the temperature can be managed proactively based on the video coding application characteristics and video content properties. As result, our proposed techniques are successful on managing temperature of video coding systems considering different hardware platforms with low overhead in the final video quality.

We address the goal of this thesis with the following key contributions:

1. **Thermal analysis and workload distribution on video coding:** we have performed an extensive thermal analysis of the video coding application in order to study the thermal behavior of the encoding process using different methodologies for temperature measurement. We have used a Peltier-based infra-red thermal measurement setup to have accurate thermal reports from the video coding application running into a real processor. We also have evaluated temperature of video coding systems using a well-known thermal modeling tool with two other performance/power simulators. The digital thermal sensors (DTS) of recent processors also have been used to measure temperature. Moreover, we have measured workload distribution considering a parallel video coding system and the impact on the temperature profiles.
2. **Relationships between video coding characteristics and video content properties with temperature:** we have analyzed temperature response considering different encoding configurations, different encoding algorithms and different video content properties. Based on these analysis we have built relationships between these different application knowledges and temperature in order to provide hints for designing thermal solutions for video coding systems.
3. **Application-driven dynamic thermal management for video encoders:** we propose an adaptive temperature optimization technique for the next generation video encoders. It exploits both application-specific knowledge and video content properties in order to manage the temperature of advanced video coding systems at the application layer. The proposed technique performs application-level prediction of the temperature trend followed by an application-level thermal management policy. It dynamically manages the temperature by performing an adaptive encoder configuration selection while providing minimum penalties in terms of bit rate and video quality.

4. **Thermal optimization using adaptive approximate computing for video coding:** we propose a thermal optimization technique that adaptively employs varying degree of approximations at both algorithm and data levels in order to reduce the temperature associated with the high efficiency video coding process while maintaining quality results. The concept of imprecise computing is used adaptively depending upon the varying resilience properties of coding different regions with different content properties. With this technique, we demonstrate the potential to improve thermal profile of video coding systems.
5. **Application-driven temperature-aware scheduling of multithreaded workloads on multicore-systems:** we propose a temperature-aware scheduling technique for on-chip multicore systems executing multi-threaded video coding. It leverages application-specific knowledge to guide the scheduling technique in order to mitigate temperature effects generated by the unbalanced workload distribution across different threads when the video coding applications is considered.

The rest of this thesis is organized as follows: chapter 2 presents the state-of-the-art works related to temperature management. Chapter 3 presents a video coding overview with basic concepts to understand the contributions of this work. Chapter 4 shows the methodologies used to perform all analyses and all evaluations presented in this thesis. Chapters 5, 6 and 7 give detailed description of the main contributions of this work. Chapter 8 present the main conclusions related to the contributions presented and also point future works towards the end of this thesis. Finally Chapter 9 shows the main publications resulted from the Ph.D. course period.

2 STATE-OF-THE-ART

2.1 Thermal related problems

For many years technology advancements were based in two main reasons. First, the growing number of transistors in a chip as observed by the Moore's law (MOORE, 2009) that states the number of transistors in a chip approximately doubles every 24 months. Due to the transistor feature size shrinking it is possible to put more components in the same chip area. Also, smaller components lead to tighter register to register paths which made possible to increase the operational clock frequency of circuits. The second reason is the advancements on microarchitecture by extracting instruction level parallelism from application using techniques such as pipeline, superscalar and simultaneous multithreading (SMT) (STALLINGS, 2010). Due to these technology advancements, CPUs have become high-performance computing systems with many processing cores operating at high clock frequencies. However, power density (power/area ratio) is no longer constant with technology scaling due to the failure of Dennard's voltage/current scaling down prediction (BOHR, 2007). For instance, power density grows from 2W/mm^2 in 65nm technology to 7.2W/mm^2 for 45nm (LINK and VIJAYKRISHNAN, 2006).

High power densities result in high on-chip temperature/thermal hotspots (LINK and VIJAYKRISHNAN, 2006). With high temperature the first problem that arises is associated with cooling costs. Cooling techniques are necessary for high performance devices in order to keep the chip temperature under safe operational limits. Cooling costs are related to technology price of designing and implementing advanced techniques to cool down the temperature of devices. From the embedded circuits perspective, in addition to the energy consumption by the circuit itself, there are concerns about the cooling costs related to area (for instance, to employ a heat sink) and energy (for instance, to power a cooler) since embedded devices usually have limited area and power constraints.

Moreover, thermal hotspots negatively affect the reliability and lifetime of devices (HENKEL, BAUER, *et al.*, 2013) (HENKEL, EBI, *et al.*, 2013). The two key mechanisms that increase the delay of transistors during their normal, failure-free operation are Negative Bias Temperature Instability (NBTI) and Hot-Carrier Injection (HCI) (BERNSTEIN, NASSIF, *et al.*, 2006). Both NBTI and HCI cause a gradual elevation of the threshold voltage (V_t) incurring on increased transistor switching delay where high temperature is one of the main factors inducing these aging effects. For instance, temperature impacts exponentially in the NBTI effect and linearly in the HCI effect (TIWARI and TORRELLAS, 2008). Electromigration, stress migration and dielectric breakdown are other failure mechanisms which cause permanent device failures that are aggravated by high temperatures and thermal hotspots (HENKEL, BAUER, *et al.*, 2013).

In addition to thermal hotspots, spatial and temporal variations in temperature (or temperature gradients) have also a negative impact on system reliability and performance. Spatial temperature variation (or spatial temperature gradient) is the temperature difference that may occurs across the chip between any two points. Temporal temperature variation (or temporal temperature gradient) is the temperature difference that may occur within a time window of chip operation. Temporal temperature variations causes accelerated package fatigue, plastic deformation of materials, and can lead to cracks and other permanent failures (COSKUN, ROSING, *et al.*, 2008). Temporal gradients increase failure rate when the magnitude and frequency of temperature cycles increase. A 10 °C increase in the magnitude of temperature cycles can cause about a 16× decrease in mean time to failure for metallic structures (JEDEC, 2006). These temperature cycles are generated either by low frequency power changes such as system power on/off cycles, or by workload rates changes and power management decisions, which are more frequent. Spatial temperature variations cause performance and logic failures (COSKUN, ROSING, *et al.*, 2008). Global clock networks are especially vulnerable to spatial variation. Every 20 degrees increase in temperature causes 5-6% increase in Elmore delay in interconnects resulting in clock skew problems (AJAMI, BANERJEE and PEDRAM, 2005).

All the above mentioned temperature problems become even more challenging when process variation is considered. Process variation is the deviation of transistor parameters from their normal specifications. The difficulty of controlling the precision of the fabrication process due to decreasing of feature size is the main factor that causes process variation. Studies have shown that for high performance microprocessors in 180nm technology, measured variation is found to be as high as 30% in performance and 20 times in chip level leakage within a single wafer (BORKAR, KARNIK, *et al.*, 2003).

Therefore, temperature management/reduction is one of the primary design objectives especially for embedded systems that have limited power and area to employ complex thermal solutions. This way, it is important to develop thermal techniques in order to provide better thermal profiles and reliable operation of systems.

2.2 State-of-the-art works

There are different ways to deal with the above mentioned temperature problems. Traditionally techniques to reduce the temperature effects in microprocessor chips were handled at package level by heat sinks of different material and coolers. Also, there are techniques that propose different solutions at design-time (such as temperature-aware placement) to forecast and avoid thermal hotspots by statically evaluating several benchmarks applications (HUNG, ADDO-QUAYE, *et al.*, 2004) (ZHANG, OGRENCI-MEMIK, *et al.*, 2015).

In this thesis, the main focus is Dynamic Thermal Management (DTM), which is employed in most of the literature works to deal with temperature problems at run-time. DTM techniques use diverse low power techniques in order to control/keep temperature under safe operational thresholds that are usually defined by chip vendor. The most common low-power techniques used in literature and a set of related works are listed below.

Low power techniques at run-time:

Task migration: This technique can be employed on architectures with more than one processing unit. For a particular reason, for instance, to balance power between processing units, a given running application can be moved from its core to another core. The overhead associated with task migration is the time and power that takes to move all cached memory content between two different processing elements.

Dynamic Voltage and Frequency Scaling (DVFS): This technique is used to reduce power by either reducing the voltage supply, which has a squared impact in the total power, and/or by reducing the operational frequency, which impacts linearly in the total power. Reducing voltage and/or frequency has a direct impact on power. However, it can dramatic reduce the performance, which may compromise applications with tight timing constraints.

Clock gating: This technique is performed at hardware level to stop the clock frequency. This way, the dynamic power is reduced since transistors switching operations are stopped.

Power gating: This technique shuts down either the whole chip or parts of the chip to avoid both dynamic and static power.

Low power techniques at design-time

Optimizing the number of transistors: This type of technique aims at reducing the total number of transistors by replacing the current circuit for an equivalent with fewer transistors.

Transistor gate sizing: This technique selects cells (for each gate in a circuit) among different implementations in a given library to optimize the total power of the circuit (combination of gate sizing and threshold voltage) according to time constraints.

The work (YEO, LIU and KIM, 2008) proposes a predictive dynamic thermal management technique for multicore systems. The authors use predictors to estimate future temperature behavior based on steady state temperature and workload. Task migration is used to move the running application from the possible overheated core to the future coolest core. This way, overall chip temperature is reduced in exchange for application performance. Average temperature can decrease about 10%, and peak temperature by 5 ºC with less than 1% impact of performance.

The work (COSKUN, ROSING and GROSS, 2008) proposes temperature balancing solution for MPSoCs with low performance cost. They use autoregressive moving average (ARMA) modeling for estimating future temperature based on previous temperature measurements. With the predicted temperature, the authors propose a job allocation policy to balance workload among cores in an MPSoC based on the spatial temperature difference between cores. In the works (COSKUN, ROSING, *et al.*, 2008) and (COSKUN, ROSING and GROSS, 2009) the authors extended this idea to reduce temporal variations in temperature by using an adaptive dynamic temperature scheduling using both variation in the applications workload and history-based temperature prediction. The results show 60% reduction in hot spot occurrences, 80% reduction in spatial gradients, and 75% reduction in thermal cycles on average.

A hybrid dynamic thermal management technique is proposed in (KUMAR, SHANG, *et al.*, 2008). It can use both hardware-based and software-based DTM techniques in a coordinated fashion to enable a more fine-grained approach to tackle thermal emergencies. The authors developed a regression-based thermal model to predict temperature based on hardware performance counter found in modern processors. Based on the predicted temperature, task migration (in the software layer) and clock gating (in hardware layer) are used to deal with thermal emergencies. The proposed hybrid technique is able to manage temperature at different levels with an average execution-time overhead of 10.4% opposed to the purely hardware-based DTM that achieves 23.4% overhead.

The work (FISHER, CHEN, *et al.*, 2011) also proposes a thermal-aware scheduling solution for multicores. However, opposed to the workload balancing solutions, in this paper, the authors propose a run-time hardware configuration of processors in a homogenous multicore system to avoid peak temperature of sporadic real-time tasks. The ideal speed of each core is derived by the proposed model considering the time constraints of each task. Depending on the multicore platform the temperature improvement can achieve up to 22 ºC.

In (EBI, FARUQUE and HENKEL, 2009) the authors propose an agent-based power distribution approach to balance the power consumption of multi/many-core architectures in a pro-active way. System thermal state is considered on distributing the power throughout the chip. The authors claim that this solution is more efficient and more scalable than usual DTM techniques since it can perform better in many-cores systems (hundreds and thousands of cores). The proposed agent solution acts in a group of cores to do self-configuration and self-optimization (changing hardware settings) with the main goal of distributing the power evenly over the chip. In the work (EBI, KRAMER, *et al.*, 2011) the authors improve the thermal solution by applying a temperature prediction and also defining voltage islands for the many-core system to avoid the overhead of using separate voltage level for each core. A hierarchical approach is proposed where higher layers use regional core information to perform coordination between local cores. The solution reaches a decrease in peak temperature of around 4% while also decreasing the number of deadline misses in periodic applications when compared to a fully distributed approach.

The work (ZANINI, ATIENZA, *et al.*, 2009) aims at achieving a smooth thermal control action in contrast to approaches that avoid thermal violations by forcing abrupt operating points changes (e.g. processor shutdown). The authors propose a thermal management policy yielding a smooth optimum control on working frequencies and voltages of multicore systems. The problem is based on model predictive control (MPC) using a control-theoretic approach. This way, dangerous thermal profiles are avoided and also temperature variation over time is reduced between 2.5× and 5× in comparison with previous methods. In the works (BARTOLINI, CACCIARI, *et al.*, 2011) and (BARTOLINI, CACCIARI, *et al.*, 2013) the authors extend the idea of using MPC for thermal management by proposing a fully-distributed solution. The MPC complexity is reduced by splitting it in a set of simpler interacting controllers, each allocated to a core in the system. Frequency and voltage configurations are set individually at run-time for each core by the low complexity MPCs.

The work (CHO, KERSEY, *et al.*, 2013) presents the effects of spatiotemporal power multiplexing as an execution principle for many-core architectures for managing temperature. The authors exploit lateral heat flow and thermal capacity of materials to redistribute the generated heat in a many-core chip by using task migration. Faster migration, i.e., a smaller migration interval, can result in lower maximum temperature and better spatiotemporal uniformity in temperature. As results the work shows that better energy efficiency is achieved even considering the energy overhead of several core transitions.

All these state-of-the-art works propose solutions for controlling temperature at different layers of the computing stack. Also, they target two goals regarding temperature problems that arise at different types of architectures. There are solutions targeting reducing the peak/average temperature of the chip focusing on single-core architectures. The other type of solutions target reducing the peak/average temperature and also the spatial temperature gradients focusing on multi-cores architectures. We classify these works as in Table 2.1 in order to have a better visualization where each state-of-the-art work acts in the computing stack layer to perform thermal management and also which type of architecture is focused.

Table 2.1: State-of-the-art works classification.

	Single-core	**Multi-core**
Hardware-level	(FISHER, CHEN, *et al.*, 2009) (EBI, FARUQUE and HENKEL, 2009) (EBI, KRAMER, et al., 2011) (ZANINI, ATIENZA, et al., 2009) (BARTOLINI, CACCIARI, et al., 2011) (BARTOLINI, CACCIARI, et al., 2013)	
System-level		(YEO, LIU and KIM, 2008) (COSKUN, ROSING and GROSS, 2008) (COSKUN, ROSING, *et al.*, 2008) (COSKUN, ROSING and GROSS, 2009) (KUMAR, SHANG, *et al.*, 2008) (CHO, KERSEY, et al., 2013)

All the above mentioned works sacrifice system performance in exchange of better thermal profiles by using low-power techniques in both hardware (DVFS) and system layers (OS scheduling). None of these works consider the application characteristics on defining their thermal techniques. They all basically use temperature history information to either react to thermal emergencies or predict future temperature. Without application knowledge the reactivity of thermal management can be slow. Moreover, the performance degradation imposed by these solutions may be intolerable for some applications (in particular for real-time applications), since the adaptation to the new timing constraints is performed by decreasing the resulting quality of the application. The above system-level schedulers consider that workloads of different threads (same application) are always the same, which is not true for all applications. Video coding/decoding is an example where the workloads can vary abruptly between threads and along time and temperature history based schedulers may fail on predicting future application behavior.

There are also DTM techniques for specifically targeting video coding/decoding systems. The work (SRINIVASAN and ADVE, 2003) proposes a predictive DTM control algorithm for MPEG (Moving Picture Experts Group) encoding. The authors use profiling information from static evaluation that frames of the same type will present similar IPC demand and power dissipation and consequently similar temperature results. With this information they reconfigure the hardware resources (e.g., instruction window size, number of active functional units) in a way that temperature does not achieve the specified threshold with low performance degradation. However, the assumption that frames of the same type have same workload may not be true since the workload is highly dependent on the content being encoded.

In (LEE, PATEL and PEDRAM, 2006) the authors propose a DTM algorithm to distribute the available slack time between different frames to achieve thermally safe state of the microprocessor chip during MPEG-2 decoding. A time demand prediction is proposed based on the GOP (group of pictures) structure. When the temperature threshold is achieved the DTM technique stalls the processor for some cycles in order to reduce the power dissipation. If the predicted number of stall cycles is larger than the available time to decode the next frame the proposed DTM uses two levels of quality degradation. Spatial quality degradation by skipping some operations in the decoding process and temporal quality degradation by skipping the decoding of an entire frame, i.e., displaying the same frame twice. The author extended this idea in (LEE, PATEL and PEDRAM, 2008) with an additional DVFS scheme to adapt the hardware configuration according to the predicted frame processing demand. Spatial and temporal quality degradations are used again when thermal emergencies are detected. In these two works the key target is to lower the temperature irrespective of the incurred video quality loss, which may be significant in presence of frame drops.

The work (YEO and KIM, 2008) proposes a hybrid DTM based on statistic characteristics of multimedia applications (for instance, MPEG-4 and H.264/AVC decoding). First a prediction step estimates the cycle demand for decoding a given frame. The prediction is performed using history of a fixed window size of previous decoded frames. With the estimated cycle demand the DTM adapts frequency to perform the decoding of following frames. If the temperature threshold is reached the frequency should be readjusted. In this work, the performance is not guaranteed for real time decoding and quality degradation steps will be necessary to attend the time constraints.

The work (MARCU, MILOS and TUDOR, 2010) presents a power-thermal analysis of multimedia applications. CPU usage, memory usage, communication bandwidth usage and complexity of the algorithms for decoding video benchmarks are measured to provide specific set of metrics for the power and thermal efficiency evaluation of the multimedia application on different mobile devices. The authors suggest that in order to obtain better results both the application level and system level should be involved in thermal management. In fact, this is exactly what this thesis is focused on, i.e., to develop application level solutions for thermal management targeting the video coding application.

The work (FORTE and SRIVASTAVA, 2010) proposes a thermal-aware video coding solution using a hybrid video encoder. The proposed encoder uses Distributed Video Coding (DVC), that shifts encoder computations to the decoder to have a low complexity encoder) and Predictive Video Coding (PVC) as a way to balance the workload between encoder and decoder. The solution offloads the encoder computations for motion estimation to the decoder in order to balance the encoder/decoder workload, reducing the overall temperature of the encoder. However, as the encoder is not fully performed, it results in more data to be transmitted between the source (encoder) and receiver (decoder) which leads to increased communication power/energy.

The work (MIRTAR, DEY and RAGHUNATHAN, 2012) proposes an adaptation technique for the video encoding application targeting in the inherent effects of dynamic thermal management techniques. It selects a tolerable video quality degradation mode based on reconfiguring the video coding application parameters to compensate for the quality degradation effects of different DTM policies, i.e., the application reacts to future thermal emergencies to avoid the drastic quality degradation effects imposed by DTM techniques.

All these state-of-the-art DTM works for video coding propose solutions that take the multimedia application general characteristics into consideration to perform better temperature management. However, all these works mainly target old video coding standards such as MPEG-2 and H.264/AVC. Therefore, they may not be efficiently applied to recent video encoders due to their novel and highly complex coding tools. High Efficiency Video Coding (HEVC) (ITU-T, 2013), Google’s VP9 (WEBM, 2013) and the Chinese Audio Video Coding Standard (AVS) (ZHENG, ZHENG, *et al.*, 2013) have recently been developed to provide better compression ratios keeping the same video quality. Moreover, these works do not account for the application-specific characteristics and video content properties that may provide a potential for more efficient application-level temperature optimization for video coding systems.

In summary, to address the thermal related challenges there is a prominent need for application-driven dynamic thermal management solutions that efficiently control temperature of video coding systems while still providing high video quality. Moreover, it is important to consider the recent and more complex video coding standards (like HEVC) on designing novel thermal management solutions due to the new complex tools. Finally, application-specific characteristics and video content properties can be used to improve temperature profiles of video encoding systems.

3 VIDEO CODING OVERVIEW

In this chapter the basic concepts of digital video and the video coding application are presented.

3.1 Digital video data representation (how video data is represented)

Usually, a digital video is composed by a sequence of equally sized pictures (called frames) and the picture size is defined by two dimensions of pixels (resolution). The video resolution is a representation of the amount of pixels of each frame in both dimensions. For instance, in a full HD resolution video each frame is composed of 1920 columns of pixels and 1080 lines of pixels (1920x1080) in a matrix fashion. As higher is the resolution better is the perception of the video details. The movement perception occurs when these frames are put in sequence with a given number of frames being displayed per second. The frequency necessary for movement perception for the visual human system is, at least, 24 frames per second (RICHARDSON, 2003). However, with the latest demand for higher resolutions the frequency of frames per second may increase to keep a smooth movement perception.

Digital videos are represented by color spaces where each pixel is separated in color components. This way, the video encoders can better deal with each color component separately by applying specific algorithms, achieving higher compression ratios. Several color spaces such as RGB (Red Green Blue) and YCbCr (Y-luminance, Cb-chrominance blue, Cr-chrominance red) (RICHARDSON, 2003) have been used to depict digital pictures. The well-known RGB color space divides each pixel in three color components: R (red), G (green) and B (blue). The YCbCr is also composed by three components: luminance (Y), chrominance blue (Cb) and chrominance red (Cr), which uses a luminance component with a blue and red deviation to define the color when displaying the pixel (RICHARDSON, 2003). The YCbCr color space is the most used by recent video encoders such as HEVC since different types of algorithms applied separately to luminance and chrominance samples increases the resulted coding quality in comparison with other color spaces.

3.2 Hybrid video coding framework (how video coding works)

The main goal of the video coding application is to represent the high amount of data of digital videos with less information as possible. This is performed using a set of coding tools that are designed considering the inherent presence of data redundancy in digital videos. There are basically three different types of data redundancy exploited by the coding algorithms in most of the current video coding standards. Spatial redundancy comes from the correlation among pixels in one frame. For instance, a region where pixels have the same value (e.g., a blue sky in the background) could be exploited to improve compression.

Temporal redundancy comes from the correlation among neighbor temporal frames that are likely to be very similar. Finally, entropic redundancy is about the amount of similar symbols generated by the video coding process. For 3D and multi-view videos, there is also other redundancy information that can be exploited by specific coding tools. For instance, on multi-view videos the inter-view information (i.e., similarities between two video streams captured from neighbor cameras) can be used on the video coding process as another source of redundancy.

In order to exploit each one of the above mentioned redundancies the current video coding standards, like HEVC, use the block based hybrid video coding framework. Each frame is divided into smaller blocks and the whole coding process is performed block by block in a raster scan order. The hybrid video coding framework is based on three well-established steps: (1) predictions, (2) residual treatment and (3) entropy coding. Figure 3.1 shows a generic block diagram of the hybrid video coding process.

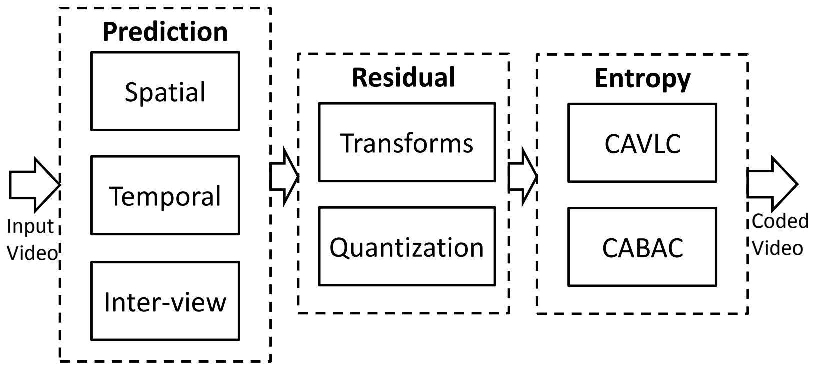


Figure 3.1: Hybrid video coding framework.

The prediction step is composed by algorithms that focus on finding the best representations of a given block being encoded using a set of redundancy references. The set of redundancy references can be composed, for instance, by the spatial and temporal neighbor blocks already coded. The prediction step builds a block using the neighbor blocks as references (called predicted block) and the difference between the original block and the predicted block is the output of the prediction step. It is important to note that there are different algorithms to generate this predicted block considering both spatial and temporal redundancies.

The temporal prediction step is performed mainly by motion estimation algorithms. The motion estimation goal is to find the best representation of the current block being encoded on the temporally neighboring frames. Figure 3.2 shows the main parameters that are considered by any motion estimation algorithm.

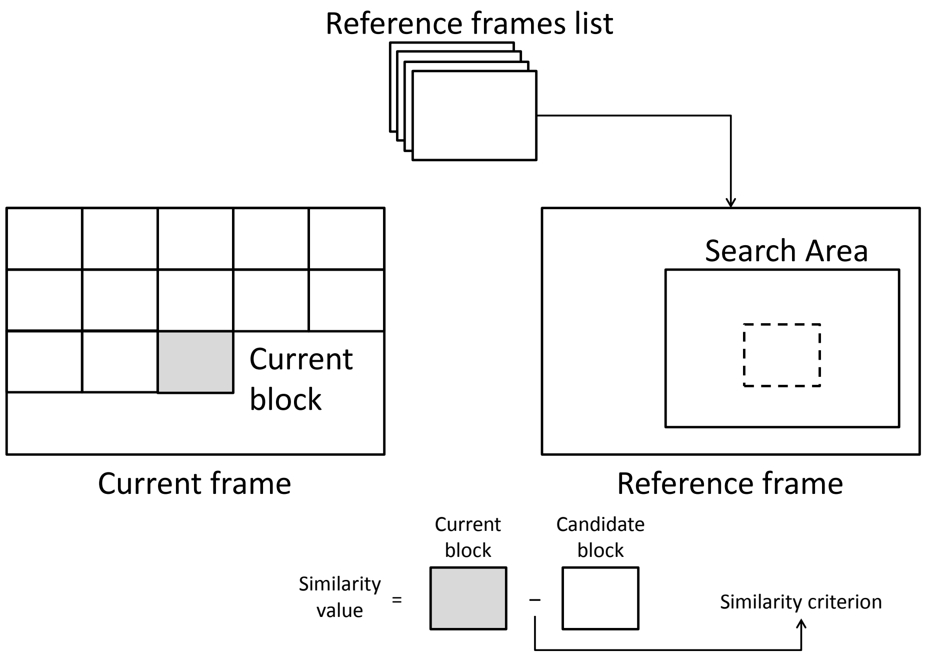


Figure 3.2: Motion estimation parameters.

The current frame is the frame that is being coded. The current block is the block that is being encoded inside the current frame. The reference frames list is the set of previously coded frames that can be used as reference to perform the motion estimation of the current block. It is important to mention that more than one reference frame can be used to perform the motion estimation of one block. The motion estimation will look for similar blocks in the reference frames considering a pre-defined search area. Although the search area can be defined as the size of the whole frame, it is common to limit the search area around the co-located block in the reference frame (as in Figure 3.2). There are several different motion estimation algorithms, i.e., different ways of finding the best representation of the current block in the reference frames. Finally, a similarity criterion is calculated between the current block and a set of candidate blocks from the search area generating a set of similarity values. The motion estimation algorithm will choose the candidate block that is most similar to the current block considering the calculated similarity value. A similarity function is applied to the set of all similarity values as in equation (3.1), where *N* is the number of candidate blocks and the "–" (dash) operator represents a generic similarity criterion where the *CurrentBlock* is compared with a *CandidateBlock.* A minimization function is applied to the union of all calculated similarity values, since as smaller is the difference between the *CurrentBlock* and the *CandidateBlock* higher is the similarity. The most used similarity criterion is the Sum of Absolut Differences (SAD) calculation.

$$min\left(\bigcup_{i=1}^{N}(CurrentBlock - CandidateBlock_i)\right) \tag{3.1}$$

The residual treatment step in the hybrid video coding framework (figure 3.1) also exploits the spatial redundancy in digital videos. Two dimensional transforms, such as DCT (Discrete Cosine Transform) and Hadamard (RICHARDSON, 2003) are applied to the residual information generated by the prediction step. The transforms are applied to separate the low and high frequencies of the input data. After that, a quantization process is performed adaptively considering the coefficients generated from the transformation process. For instance, the quantization strength will be high on frequencies that are less perceptive to the human eye and low to those frequencies that contribute more to the perception of the human eye. The quantization process plays an important role on the video coding process to achieve high compression ratios. However, this is a lossy process (i.e., information will not be recovered by the decoding process) which makes the quantization strength configuration crucial to define the final visual quality of the encoded video.

The entropy coding step uses different algorithms to reduce the redundancy in the symbols generated after the quantization process. Context Adaptive Variable Length Coding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC) (RICHARDSON, 2003) are examples of algorithms that perform entropy coding.

3.3 HEVC Standard (it is complex but it can be configured)

The HEVC is one of the next generation video coding standards that have recently been developed to provide high compression ratios for the new high-end video contents. HEVC follows the hybrid video coding process as its predecessor H.264. It provides 2x higher compression ratios when compared to its predecessor H.264/AVC (OHM, SULLIVAN, *et al.*, 2012) (SULLIVAN, OHM, *et al.*, 2012) by incorporating an advanced set of novel coding tools. However, the computational complexity of the coding process can reach 3.2x when comparing the HEVC standard with its predecessor H.264/AVC (VANNE, VIITANEN, *et al.*, 2012).

3.3.1 HEVC coding structure

The new HEVC coding structure plays an important role in the compression ratios achieved by the standard. The coding process is structured as a recursive quad-tree partitioning of the basic Coding Tree Unit (CTU – 64x64 pixels). The CTU can be divided, in a hierarchical fashion, in four parts from 64x64 to 8x8 generating the so called Coding Units (CU) as demonstrated in figure 3.3. The 64x64 CU can be divided into four 32x32 CUs. Each one of the 32x32 CUs can be divided again into four 16x16 CUs. Finally, each one of the 16x16 CUs can be divided again into four 8x8 CUs. This flexibility is useful to match the encoding process to the variety of different video content possibilities. For instance, heterogeneous and high detailed image regions may be better encoded using smaller CU sizes while homogenous and less detailed regions may be better encoded with larger CUs.

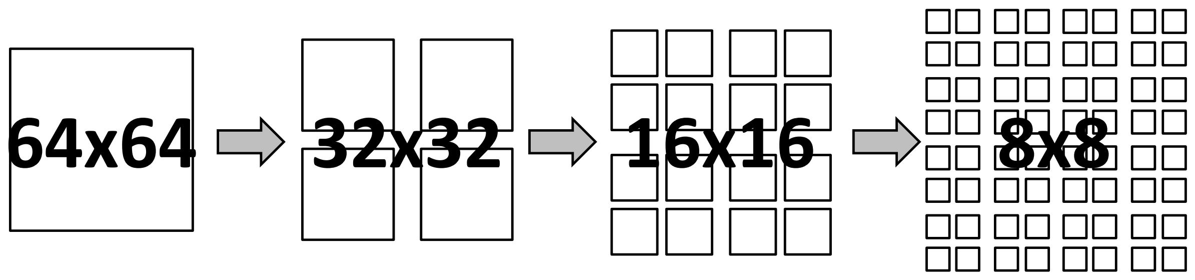


Figure 3.3: HEVC quad-tree encoding structure (PALOMINO, SHAFIQUE, *et al.*, 2016).

Each one of all generated CUs is analyzed by all encoding tools to determine an appropriate compression mode. After that, the best division in terms of coding efficiency (for instance, bit rate and visual quality) is chosen. Figure 3.4 shows an example of one possible division.

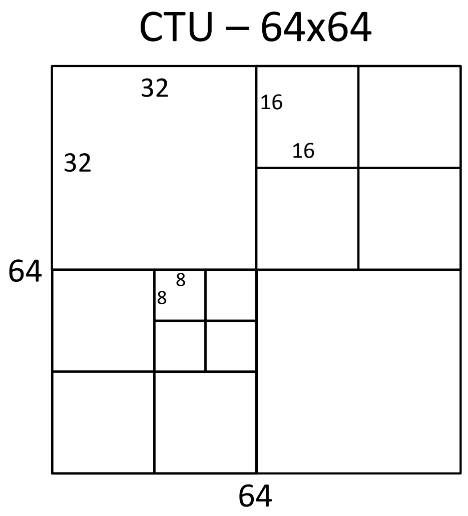


Figure 3.4: Example of best division for encoding a CTU.

This high number of possibilities to encode one CTU highly improves the HEVC coding efficiency in terms of bit-rate and visual quality when compared to older standards. However, the computational complexity associated with processing all CTUs considering all CU sizes highly increases in comparison with older standards. For instance, H.264 limits the encoding process to blocks no larger than 16x16 pixels.

3.3.2 HEVC encoder configuration

The HEVC standard is indeed more complex than old video coding standards and this is the main reason why the compression ratios are expressively higher. However, it is possible to configure the coding process targeting different goals, such as less complexity or better coding quality. There are several encoder parameters that can be used to tune bit-rate, visual quality and computational complexity of the coding process. In this thesis we focus on four of these parameters since they have strong and direct impact on coding results.

QP: The QP is the quantization parameter. It defines the strength of cut that will be applied on each resulting coefficient from the transforms in the residual step. The quantization is applied adaptively to each generated coefficient considering the perception impact on the human eye. The quantization will prepare the coefficients for the entropy coding step. As close to zero “0” is the value of a coefficient easier is the process of entropy coding (less complex) to generate a compressed video with fewer bits. However, this parameter should be used carefully since high QP values can dramatic reduces the quality of the compressed video.

CTU maximum size: Although the CTU default size is 64x64 the coding process can configure this parameter to be smaller. It means that the recursive division can start from other sizes such as 32x32, 16x16 and 8x8. With the CTU maximum size configured smaller than 64x64 less possibilities needs to be evaluated by the encoder and the computational complexity can be reduced. However, this can negatively impact in the bit-rate and visual quality results.

Number of reference frames: This parameter sets the number of reference frames that can be used in the motion estimation process. High number of reference frames implies in more effort from the motion estimation process since more candidate blocks will be evaluated by the similarity criterion. However, it opens space to the motion estimation to find more similar candidate blocks, which improves the result. With low number of reference frames, the motion estimation can be performed faster at a cost of not finding the best candidate block.

Size of search area: This parameter sets the size of search area for motion estimation, i.e., the limits where the motion estimation algorithm will look for candidate blocks to represent the current block. Following the same idea as the number of reference frames, bigger search areas implies in more effort from the motion estimation, but also in better results. While for smaller search areas the motion estimation will perform faster at a possible cost of not finding the best possible candidate block.

In summary each one of the above mentioned coding parameters has a different contribution on the coding results. Number of bits, visual quality and complexity are directly affected by each one of these parameters. Table 3.1 summarizes the coding parameters and their influences on the coding efficiency. It is important to note that it is possible to configure the parameter of the coding process in order to reduce its complexity. However, every configuration towards reducing complexity will come with possible damage in the quality of the encoded video. Each green arrow in Table 3.1 is a good effect of changing one of the coding parameters while a red arrow means a bad effect.

Table 3.1: Summary of coding parameters and their impact on resulted coding attributes.

Coding Parameter		**Compression Efficiency**	**Complexity**
QP	⇧	🟩⇩	🟩⇩
	⇩	🟥⇧	🟥⇧
CTU size	⇧	🟩⇧	🟥⇧
	⇩	🟥⇩	🟩⇩
Reference Frames	⇧	🟩⇧	🟥⇧
	⇩	🟥⇩	🟩⇩
Search area	⇧	🟩⇧	🟥⇧
	⇩	🟥⇩	🟩⇩

3.4 Coding efficiency metrics (different ways to evaluate coding efficiency)

The success of devices and applications that deal with high resolution digital videos is possible due to the high compression rates provided by latest video coding standards (like HEVC). These high compression rates come from the new high complex coding tools and also because the video coding process allows some degree of data loss. There are different metrics to evaluate the generic concept of coding efficiency.

The first one is the bit-rate generated after the coding process. This is a simple measurement of the amount of bits used per unit of time by the coded video (usually bits per second). Looking only from the bit-rate perspective, smaller bit-rate values can mean better coding efficiency. The bit-rate metric can be considered more important than other coding efficiency metrics when band-width is limited and real-time transmission is a constraint to consider in the video coding system.

The second important metric is the visual quality after the coding process. This is a metric not easy to measure and evaluate since there is a subjective component. Although there are subjective metrics to evaluate video sequences, the methodology of gathering groups of people to evaluate and score image quality makes harder the frequent use of this type of measurement. Objective metrics have become very popular to perform image quality evaluation due to the difficult of using subjective metrics. The objective metrics basically calculates a distortion value between the decoded video with the original video.

The most known objective video quality metric used by most literature works on video coding is the Peak Signal-to-Noise Ratio (PSNR in decibel), which is defined in equation (3.2).

$$PSNR_{dB} = 20.\,log_{10}(\frac{MAX}{\sqrt{MSE}}) \tag{3.2}$$

In equation (3.2), *MAX* means the maximum value that a luminance sample can be (2^n-1, where *n* is the number of bits used to represent a luminance sample) and *MSE* is the Mean-Squared Error, defined in equation (3.3).

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(R_{i,j} - O_{i,j}\right)^2 \tag{3.3}$$

In equation (3.3), m and n are the frame dimensions in pixels (height and width, respectively) and *O* and *R* are the luminance components in the original and reconstructed decoded frame, respectively.

For instance, a video coding system targeting high PSNR values will generate high bit-rate values. On the other hand, if the video coding system targets low bit-rates the final visual quality will be degraded. This inversely proportional relation between both coding efficiency metrics bit-rate and visual quality was a problem to compare coding efficiency between two different coders. If the first coder presents better bit-rate results than the second coder and the second coder presents better PSNR results than the first coder it is hard to determine which coder is better.

In order to provide a metric that can be used to compare two coders, (BJONTEGARD, 2001) describes a method for calculating the average difference between two rate-distortion curves considering two metrics: (1) BD-Rate, which measures the percentile (%) of bit-rate variation between two coders for the same visual quality and (2) BD-PSNR, which measures the PSNR variation (dB) between two coders for the same bit-rate.

The last coding efficiency metric discussed in this thesis is the computational intensity demand of the video coding process. This is an important metric for this work, since high temperature generation is usually related with high computational intensity demands. This way, this is a metric that will be often evaluated in this work. The time (in seconds) to code one video sequence is used, in this work, as a metric for computational intensity.

3.5 Opportunities of temperature optimization for video coding

As mentioned in previous sub-chapters, latest video coding standards present high computational complexity when compared to previous standards. It means that video coding applications with real-time constraints will demand high computational intensity from the hardware platform where the system is running on to be performed. Consequently, the video coding system can generate undesirable high temperatures that can jeopardize the safe operational state of the system (as discussed in chapter 2.1). This way, in case of thermal emergencies reconfigurations needs to be performed in the video coding system to avoid unsafe temperature levels. As discussed in chapter 2.2 there are techniques that perform this reconfigurations in both hardware-level and system-level at a cost of quality degradation in the final application result.

Considering the video coding there are application specific opportunities to exploit towards mitigating the temperature effects on video coding systems keeping the coding efficiency loss at minimum.

Application natural resilience to error: most known video coding standards, such as MPEG-2, H.264 and HEVC, uses the hybrid video coding framework that includes data lossy algorithms. These algorithms exploit the limited perception of the human eye (RICHARDSON, 2003) system to eliminate unnecessary information of the digital video. This information elimination is controlled by the coding configuration parameters. Moreover, matching algorithms in the Motion Estimation process have natural resilience to error since even when the optimal match is not found, in some cases, a sub-optimal match can be enough to provide a good quality result. This way, it is possible to exploit this application natural resilience to error in order to mitigate the temperature effects on video coding systems.

Application workload dependent on content: different from many applications where the computational workload is defined by the amount of data required to be processed, in the video coding application, the workload is mainly affected by the video content. Usually, high detailed and high motion video regions are harder to encode than low detailed and low motion regions. This knowledge can be used to drive temperature-aware solutions that keep good quality results.

High configurability: as summarized in section 3.3.2 there are encoder parameters that can be configured to reduce the total computational workload of the encoding process with impact on the compression efficiency results.

Trade-off between temperature and compression efficiency: Temperature management algorithms can exploit the trade-off between temperature and compression efficiency by appropriate choosing of the encoder parameters.

Video coding parallel tools are not efficient distributing workload: the parallel tools of video coding standards can be optimized to improve temperature profiles of multi-core systems. As the video coding computational workload is dependent on content, equally division of the information among different processing units may incur in undesired spatial temperature gradients.

In this thesis, we exploit the above mentioned opportunities targeting improving the temperature profile of video coding systems. We employ temperature-aware solutions driven by these opportunities of temperature optimization for video coding using different algorithms and techniques at the application-level.

4 TEMPERATURE MEASUREMENT METHODOLOGIES

In this chapter we present three different methodologies that are used along this work to collect temperature results considering all evaluations performed in this thesis. Namely IR-Camera setup, Tool Chain setup and DTS setup. We also present the general methodology used for encoding video sequences when extracting the temperature results.

4.1 IR-Camera setup

To circumvent the information deficiency on the thermal distribution across modern processor chips and to accurately analyze their thermal behavior when executing complex real-world applications such as video coding, we have performed our first thermal evaluations considering a real-time thermal imaging of a processor using an infra-red camera setup (CES-KIT, 2013). It facilitates researchers and designers to develop, optimize and evaluate thermal management policies.

The setup is built by removing the whole cooling system (fan, metal heat sink, and packaging) of the processor chip as it is an infra-red opaque in order to expose its die for measuring the emitted thermal radiations. Consequently, to keep the processor temperature in a safe range without affecting the thermal imaging, alternate IR-transparent cooling mechanisms are used. Figure 4.1 shows the components of the IR-camera based setup used to measure temperature from the chip.

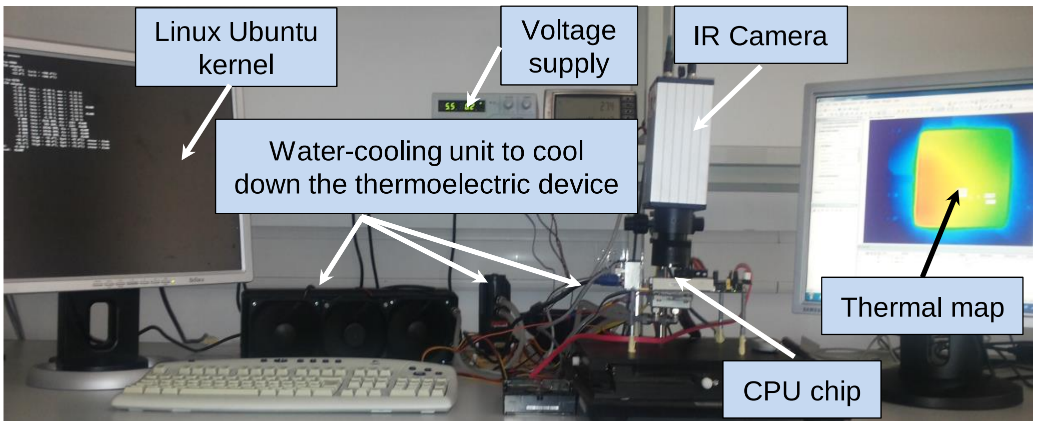


Figure 4.1: Infra-red thermal measurement setup (CES-KIT, 2013).

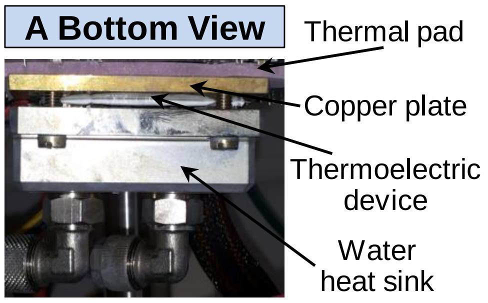


Figure 4.2: Bottom-view of the chip showing the cooling mechanism (CES-KIT, 2013).

State-of-the-art works deploy thermal setups (MESA-MARTINEZ, BROWN, *et al.*, 2008), (REDA, 2011) and (LIAN, KNOX, *et al.*, 2012) introduce an additional transparent layer of mineral oil between the IR-camera and the die-under measurement for cooling. This can disturb the thermal imaging due to its thickness and thermal convection (PALOMINO, SHAFIQUE, *et al.*, 2014). Therefore, we use an oil-free measurement setup (CES-KIT, 2013) that provides more clear thermal imaging (Figure 4.1 and Figure 4.2). It allows the processor to work within operational conditions and does not require adding any new layer between the die-under-measurement and the IR-camera lens. It employs a thermoelectric device that continuously cools down the chip from the bottom side (i.e. right below where the chip is located on the printed circuit board) in order to maintain a safe operation. When the electrical current flows through the thermoelectric device, it generates a temperature variation between both device sides making on side cold and the second one hot. By tuning the voltage fed into the thermoelectric device, a wide range of applied chip cooling can be obtained to support different kinds of processor requirements.

In this IR-camera based methodology, an Intel Atom 45nm dual-core processor operating at a maximum frequency of 1.8 GHz (layout presented in Figure 4.3) is used. The on-chip temperatures are directly measured using a DIAS pyroview 380L compact IR-thermal camera capable of precisely capturing temperatures with accuracy of ±1 °C and a spatial resolution of 50 μm per pixel (DIAS, 2013). The real-time thermal images are taken and sent to a PC (at a frame rate of 50 Hz) that analyzes them to build the corresponding thermal maps of the measured processor over time.

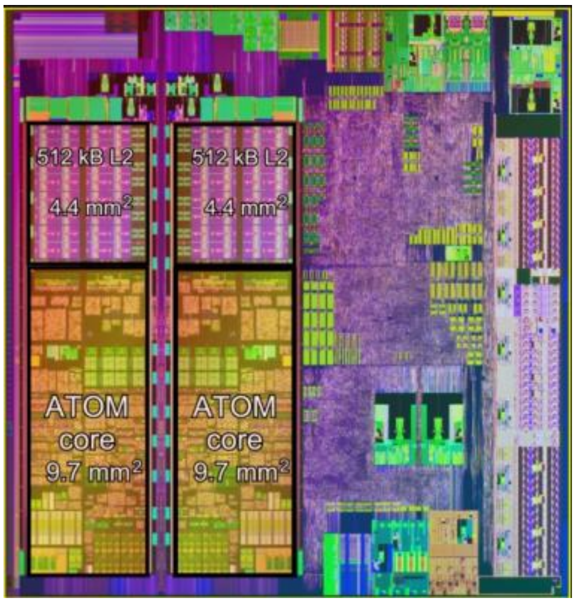


Figure 4.3: Layout of *Intel Atom* 45nm dual-core processor (CHIP-ARCHITECT, 2010).

4.2 Tool Chain setup

The second methodology used in this thesis to extract temperature values is performed using an integrated tool chain of GEM5, McPAT, and HotSpot tools.

First, we use the GEM5 (BINKERT, BECKMAN, *et al.*, 2011) simulation infrastructure which is a merge of M5 (BINKERT, DRESLINSKI, *et al.*, 2006) and GEMS (MARTIN, SORIN, *et al.*, 2005) simulators. GEM5 provides a flexible, modular simulation system that is capable of evaluating a broad range of systems. It offers a diverse set of CPU models, system execution modes, and memory system models. In this thesis, we use a 32nm 2 GHz out-of-order Alpha as targeted processor. We simulate the video coding application over a Linux installation. We also consider different topologies varying the number of cores from one to eight to evaluate parallel video coding scenarios. The output of GEM5 is a set of usage statistic values of all system components. As we intend to evaluate temperature generation on cores our setup is interested on collecting only the core usage statistics.

The core usage statistics generated by GEM5 is used to feed the McPAT (LI, STRONG, *et al.*, 2009) tool. McPAT is an integrated power, area and timing modeling framework that supports design space exploration for multicore configurations ranging from 90nm to 22nm. It is designed to work with a variety of performance and thermal simulators. McPAT allows users specify low-level configuration details. It also provides default values when the user specifies only high-level architectural parameters. In our setup, we use McPAT to generate power traces and high-level chip layout (cores and memory cache) from GEM5 core statistics and system configuration.

Finally, HotSpot (SKADRON, STAN, *et al.*, 2003) thermal modeling tool is used to generate the temperature profile and thermal maps of the application simulation. Hotspot is an accurate and fast thermal model to evaluate temperature profiles in architecture studies. It was built as an equivalent circuit of thermal resistances and capacitances corresponding to microarchitecture blocks and essential aspects of the thermal package (SKADRON, STAN, *et al.*, 2003). One of the advantages of Hotspot is the compatibility with lots of power/performance tools, such as McPAT, which facilitates put the tools to work together. The communication between the GEM5, McPAT and Hotspot used in this work is shown in Figure 4.4.

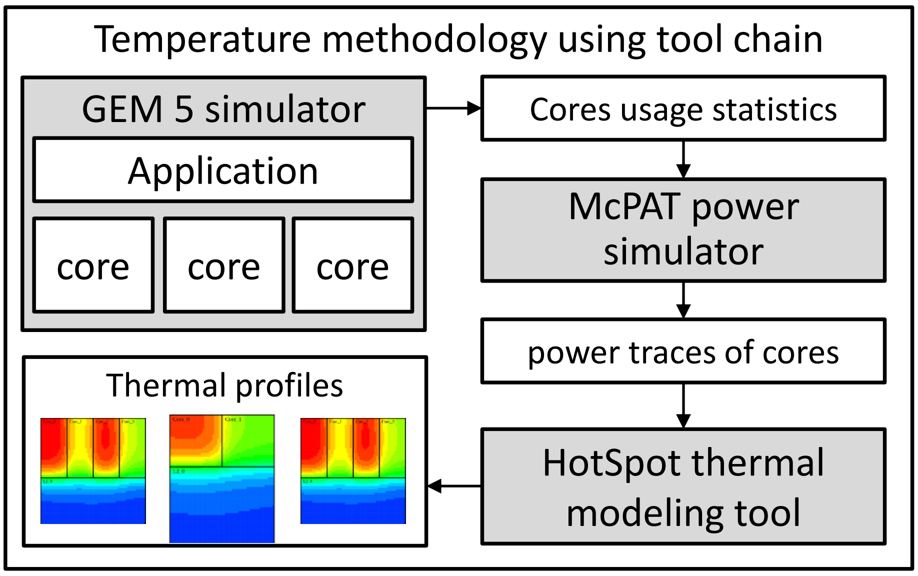


Figure 4.4: Temperature measurement methodology using tool chain.

4.3 DTS (Digital Thermal Sensor) setup

The third methodology used in this work to perform thermal analysis and collect temperature results from the proposed techniques is based on digital thermal sensors (DTS) using the Platform Environment Control Interface (PECI) (BERKTOLD and TIAN, 2010). Modern processors usually have a digital thermal sensor to measure cores temperature independently in real time. Temperature may be used to control fan speed in the system or used by costumers to develop advanced power management or thermal control schemes (BERKTOLD and TIAN, 2010).

We obtain temperature results from an *Intel i7* processor using the Linux monitoring sensors software. The video coding application is restricted to one core, i.e., task migration is disabled. This way, it is possible to obtain the resulted temperature of only the video coding application. Temperature readings are performed every 500 milliseconds by a python written script with only the operational system basic applications running together with the video coding application.

4.4 Video coding experimental methodology

For all video coding experiments the High Efficiency Video Coding (HEVC) standard is used. Table 4.1 shows the set of sequences that are used along all experiments presented in this thesis. All sequences are encoded using the HEVC test Model (HM) software following the recommendation document provided by the standardization committee (BOSSEN, 2012). Different HM software versions are used since this work started right after the HEVC standardization and the software is constantly being updated to new versions.

Table 4.1: Video sequences used on video coding experiments.

Sequences	**Resolution (pixels)**
BQMall	832x480
BasketballDrill	832x480
RaceHorses	832x480
Keiba	832x480
PartyScene	832x480
BQTerrace	1920x1080
BasketballDrive	1920x1080
Cactus	1920x1080

In the next chapters we give detailed description of the temperature-aware solutions for video coding designed in this work. The main characteristic among all proposed techniques is that the abstraction of temperature management is raised to the application-level where the video coding application characteristics and video sequences content are explored to allow efficient temperature management with low quality degradation.

5 APPLICATION-DRIVEN DYNAMIC THERMAL MANAGEMENT FOR HIGH EFFICIENCY VIDEO CODING

The first temperature-aware solution proposed in this work is an application-driven dynamic thermal management (DTM) technique for the high efficiency video coding (HEVC) standard. For designing an application-driven DTM policy, we perform an extensive thermal analysis of the HEVC standard and study the thermal behavior of different coding configurations and video sequences.

5.1 Thermal analysis of the HEVC encoder

This first thermal analysis is performed using our IR-camera based setup. The main goal is to measure temperature when encoding different video sequences with HEVC under different scenarios to understand how video content properties (like motion/texture intensity) can directly affect the computational effort needed to encode each sequence and consequently the temperature behavior.

5.1.1 Thermal analysis of different sequences

Thermal analysis for different video sequences considering Core Idling: For this first thermal analysis we consider same arrival time for each frame in the video coding process. It means that all video sequences were encoded considering the same time window per frame. Therefore, if one video frame is encoded earlier, the core is put in the *idle* state to reduce the temperature. Figure 5.1 show the core temperature over time for two test video sequences: "*RaceHorses*" with high motion intensity (red line) and "*BQMall*" with low motion intensity (blue line). For the high motion sequence the temperature stays high for most of the time due to less idle period, while for the "*BQMall*" sequence the temperature goes down earlier and stays low for a larger time. The average temperature[1] of encoding the low motion sequence is 2.5 ºC lower than encoding the high motion one. This shows that reducing the workload directly influences the cooling period of a core. Hence, *the workload of a high motion sequence can be reduced to increase the cooling down period*.

[1] In this work, average temperature means the mean of all measured temperatures within a frame encoding interval.

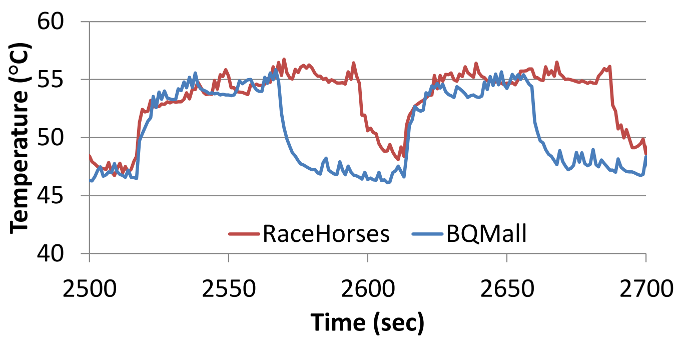
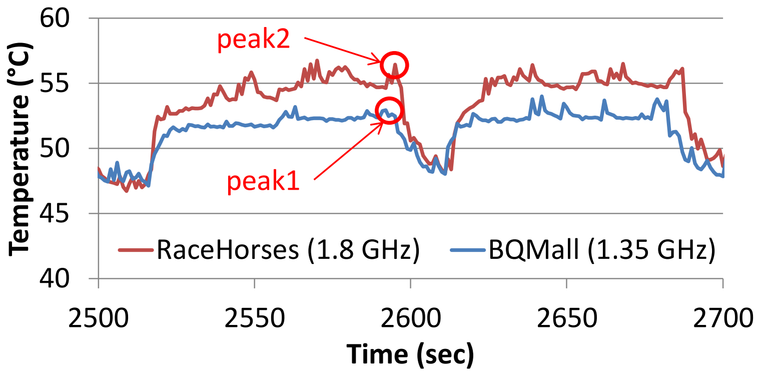
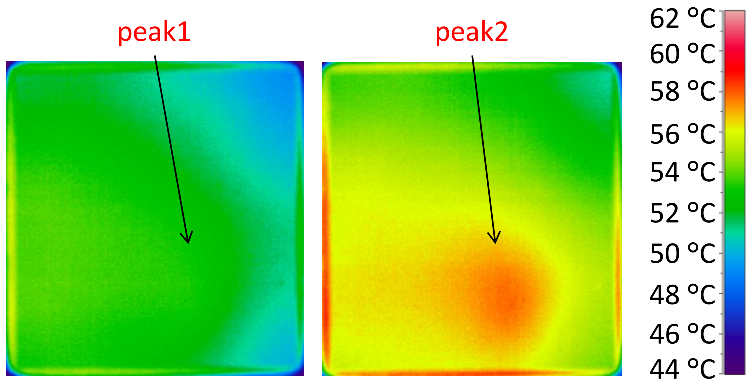


Figure 5.1: Temperature cycles for two sequences with idling core.

Thermal analysis for different video sequences considering frequency scaling: an alternate method to reduce the average temperature is executing the workload at a low frequency while stretching the execution time so that the encoding finishes near the next frame arrival time; otherwise the core is put to the idle mode. Figure 5.2 (a) shows the core temperature over time for the same two video sequences when reducing the operating frequency by 25% of the maximum frequency (1.8 GHz to 1.35 GHz) for the low motion sequence[2]. This results in a decrease in the peak/maximum core temperature from 56.4 ºC to 53.9 ºC. Figure 5.2 (b) shows the thermal map extracted by the infra-red camera[3] at two interesting points in time before going to the idle state (peak1 and peak2). It shows that for the same arrival time the maximum temperature of the low motion sequence is lower than that for the high motion sequence.



(a) Temperature cycles using frequency scaling.



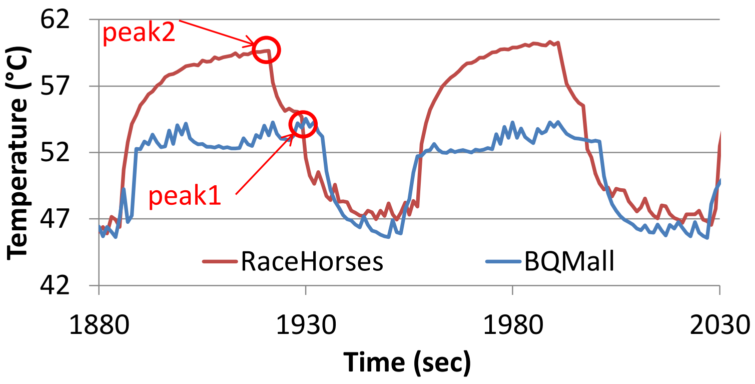
(b) Thermal maps using frequency scaling.

Figure 5.2: Temperature analysis using frequency scaling for low motion sequence.

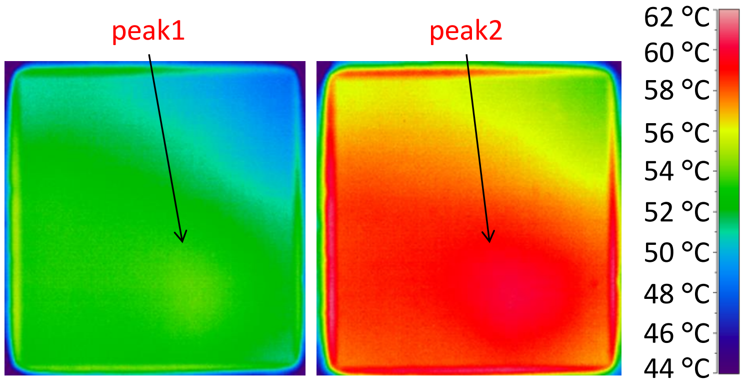
[2] Voltage scaling is not available on our Atom board.

[3] The thermal maps are mirrored with the die floorplan since this is how the infrared camera shows the images.

Thermal analysis for different video sequences considering parallelism in HEVC using Multiple Tiles: another challenging scenario is considering tighter deadlines. In such a case the low motion sequence may finish in time, but the high motion sequence needs more performance to finish the encoding process in time. This can be achieved by using the HEVC's tile-level parallelism, i.e. partitioning the frame into two parts and executing two threads on two cores in parallel. Figure 5.3 (a) shows the temperature over time for two sequences with a tighter deadline. The "*RaceHorses*" sequence is parallelized on two cores using two tiles to achieve high throughput so that both sequences finish their execution at the same time. If this is before the next frame's arrival time, the cores are put to the idle state (see sudden temperature drop). The use of two cores for the "*RaceHorses*" encoding leads to 5 ºC higher temperature when compared to the "*BQMall*" encoding. Figure 5.3 (b) shows the temperature state of the die at the end of frames encoding (peak1 and peak 2) using the thermal maps.



(a) Temperature cycles using two cores.



(b) Thermal maps using two cores.

Figure 5.3: Temperature analysis for high motion sequence with two cores.

Thermal analysis of video content properties: as already mentioned, the efficiency and complexity of advanced video coding tools are usually driven by the properties (e.g. motion/texture) of the input sequence. For instance, the motion estimation algorithms search for similarities among the sequence content across different frames (see chapter 3.2). In case of videos with irregular, textured, and complex moving objects, it is harder for the motion estimation to find a good match. Therefore, video properties such as motion/texture intensity will highly influence the coding complexity and resulting temperature for a given sequence and even for different frames within the sequence. For this thermal analysis of video content properties we classify the frames of a sequence in different complexity classes using the method devised in (SHAFIQUE, ZATT, *et al.*, 2012) (equations 5.1 and 5.2). Without the loss of generality, we use the texture intensity to classify the frame complexity C_f as *low*, *medium* or *high* (see equation 5.2). The texture is calculated using variance v_f of the luminance samples ρ_i in one frame of dimension $n \times m$ as presented in equations (5.1). The threshold values Th_{vi} are statically calculated. Using such a complexity classification, we will be able to observe how these properties influence on temperature behavior.

$$v_f = \frac{1}{n \times m} \sum_{i=0}^{n \times m} \left(\rho_i - \rho_{avg}\right)^2 \tag{5.1}$$

$$C_f = \begin{cases} low & if\left(v_f \le Th_{v1}\right) \\ medium & if\left(Th_{v1} < v_f \le Th_{v2}\right) \\ high & if\left(v_f > Th_{v2}\right) \end{cases} \tag{5.2}$$

Figure 5.4 shows the average temperature distribution for different complexity classes when encoding different sequences, e.g. *RaceHorses* (*medium-to-high* complexity) and *BQMall* (*medium-to-low* complexity. It is possible to observe that sequence properties directly influence temperature behavior of the HEVC video coding process. The *high* complexity frames will lead to higher temperatures while for the *low* and *medium* complexity frames the temperature distribution is relatively lower. The temperature difference between the *high* and the *low* complexity frames can be up to 10 ºC. Another observation from figure 5.4 is that high complexity frames have more "concentrated distribution" since the steady temperature is achieved only when these type of frames are encoded. Moreover, the high temperature distribution is mainly from the consecutive *high* complexity frames of videos like *RaceHorses*. The distribution of *medium*/*low* complexity frames are less concentrated since these types of frames are intercalated between each other in the evaluated sequences. The temperature of encoding theses frames is influenced by the "encoding temperature history".

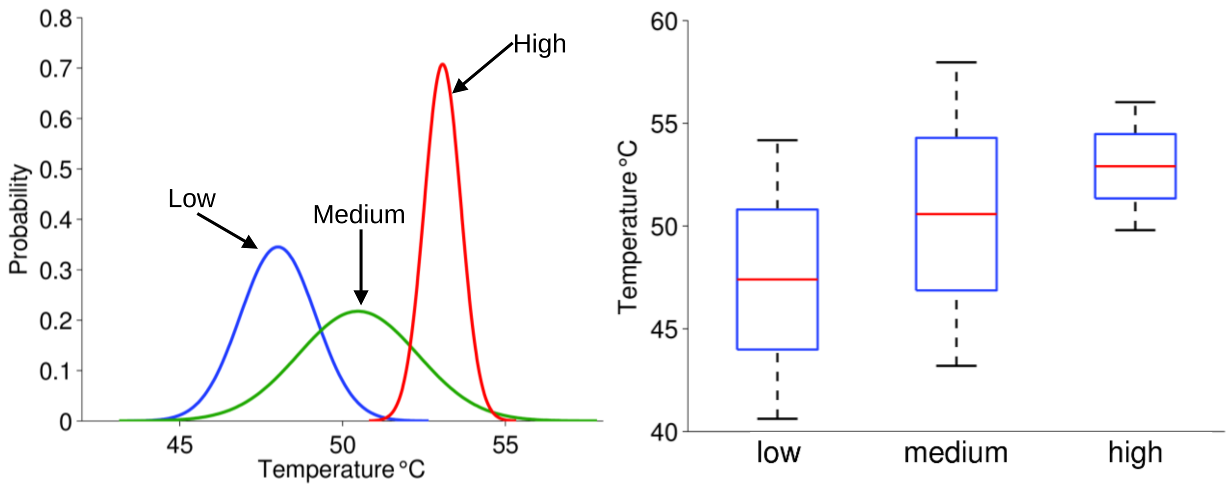


Figure 5.4: Temperature distribution for different complexity frames.

Summary of different sequences analysis: The above thermal analysis demonstrates that video properties (such as motion/texture intensity) directly influence the generated temperature for HEVC encoding considering different scenarios. Also, it is possible to observe that there is potential of applying Dynamic Thermal Management (DTM) mechanisms (e.g. core idling or frequency scaling) in order to have safe operational temperature on video coding systems. *Therefore, one of the key challenges is to leverage the video content properties of sequences to enable application-level temperature management during the HEVC encoding.*

5.1.2 Thermal analysis of different HEVC parameters

In the following, we analyze the impact of key parameters employed by all the advanced video encoder on the CPU temperature and encoding quality (in terms of PSNR and bit rate). The parameters analyzed in this work are the same described in chapter 3.3.2: maximum size of the coding tree unit (CTU), quantization parameter (QP) and number of reference frames (RF) and search area (SA) used for the motion estimation algorithm. Independent parameter analysis is performed to understand their impact on temperature.

Analyzing the impact of QP on temperature: The QP is a parameter used to control bit rate and it consequently influence on the final visual quality. Figure 5.5 shows the average temperature, bit rate, and PSNR values while figure 5.6 shows the temperature over time for four different QP values (22, 27, 32, and 37). It is noticeable that the QP selection can provide average temperature reductions of 2ºC between each value. This provides a hint that QP changing can be used for designing an application-level temperature optimization algorithm. However, this parameter needs a careful selection to avoid significant quality loss as shown in figure 5.5 (white bar). Figure 5.7 shows thermal maps of the steady temperature for the same four QP values.

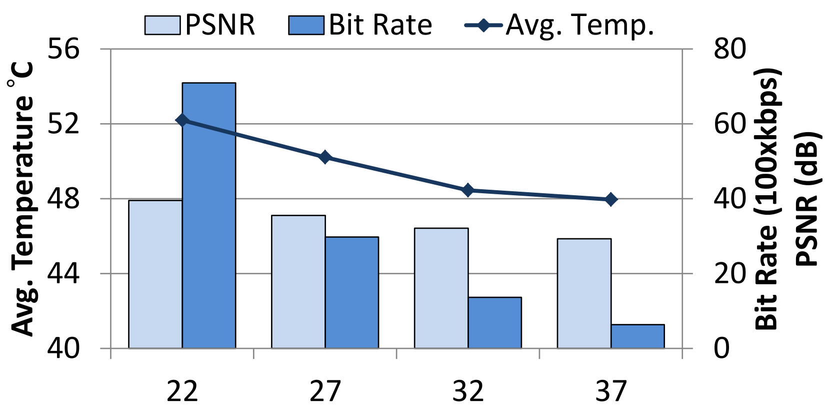


Figure 5.5: Temperature, bit rate and PSNR for different QPs.

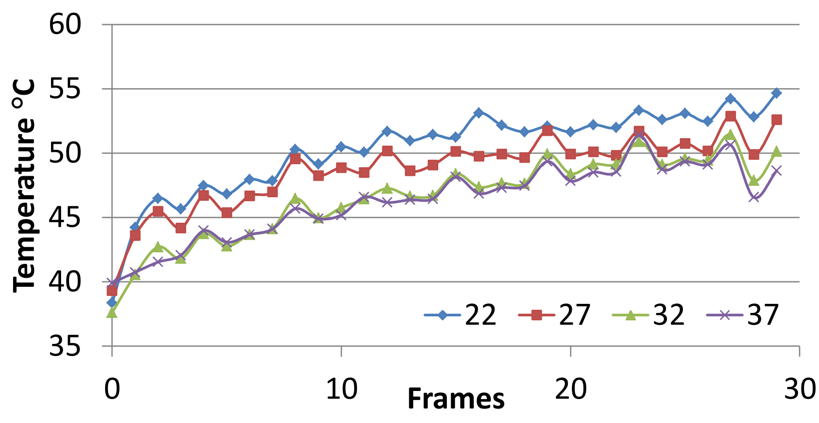


Figure 5.6: Temperature distribution over time for different QP values.

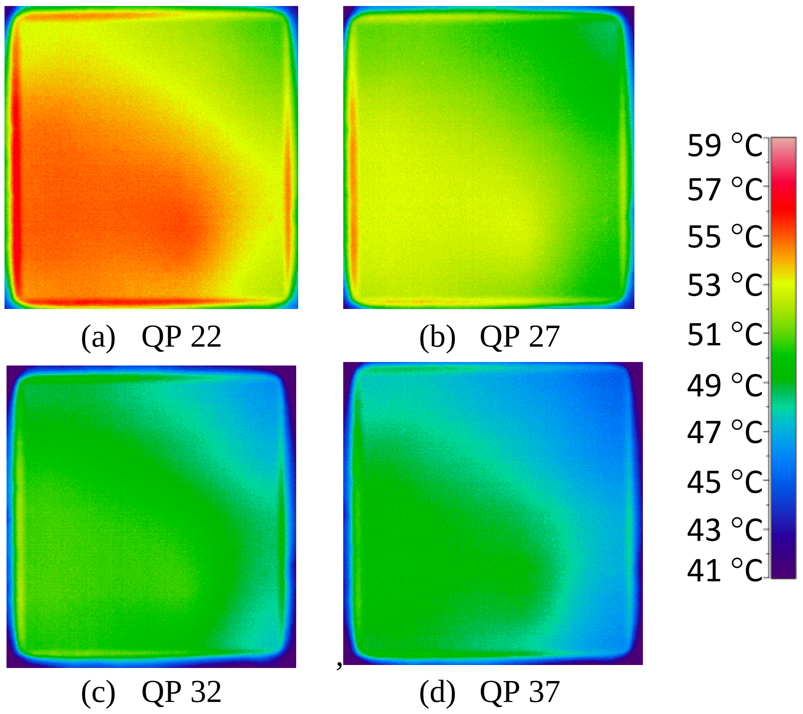


Figure 5.7: Thermal maps for different QPs.

Analyzing the impact of maximum CTU size on temperature: The HEVC standard achieves high compression efficiency using large-sized blocks in a coded tree unit (CTU) structure with different coding unit (CU) sizes. Unlike earlier standards, HEVC allows for selecting the maximum CTU size among 64x64, 32x32 and 16x16 pixels to capture different object sizes for improved compression efficiency. However, different CTU sizes lead to varying thermal profiles. Figure 5.8 shows the average core temperature, bit rate, and PSNR values while figure 5.9 shows the temperature over time for different CTU sizes. Bigger CTUs lead to high encoding effort per video area, since more options are available for block partitioning resulting in high temperatures. When using a CTU size of 32 instead of 64, the average temperature is decreased by 5ºC and the bit rate increases by 2.17%. Figure 5.10 shows an example of thermal maps at the steady temperature when encoding the *RaceHorses* sequence with different CTU sizes.

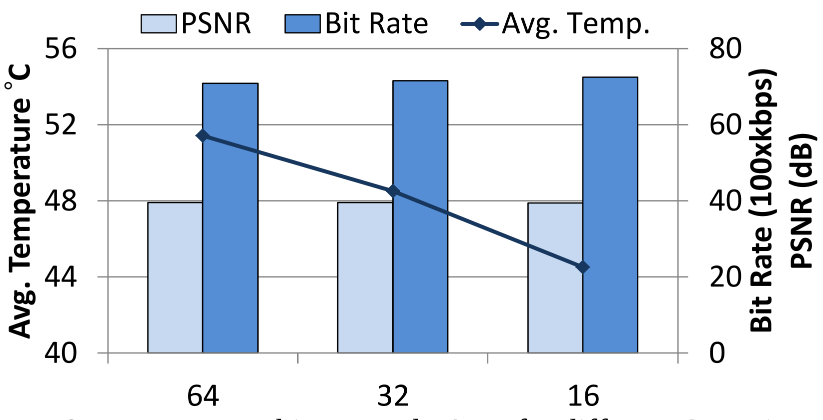


Figure 5.8: Temperature, bit rate and PSNR for different CTU sizes.

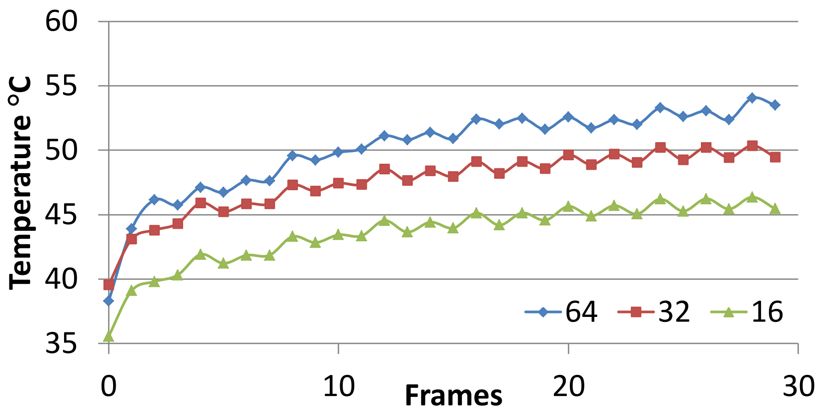


Figure 5.9: Temperature distribution over time for different CTU sizes.

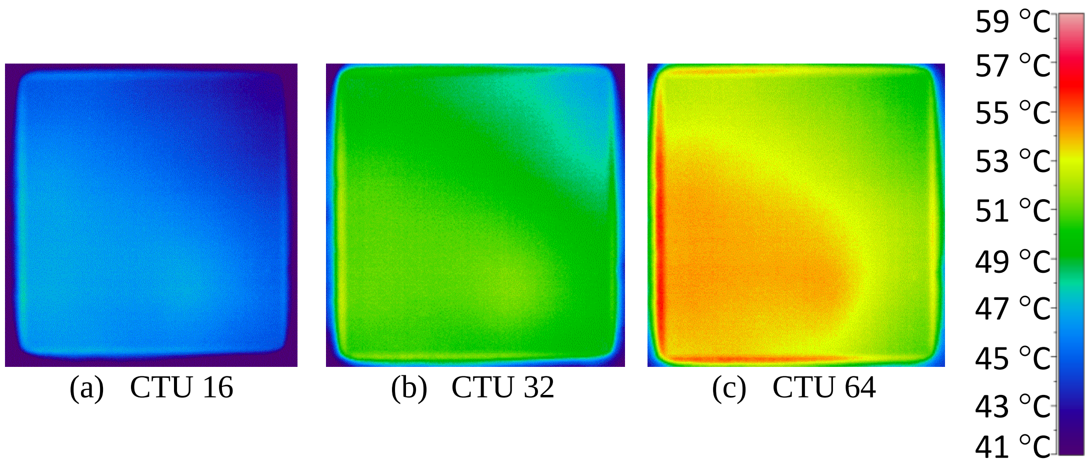
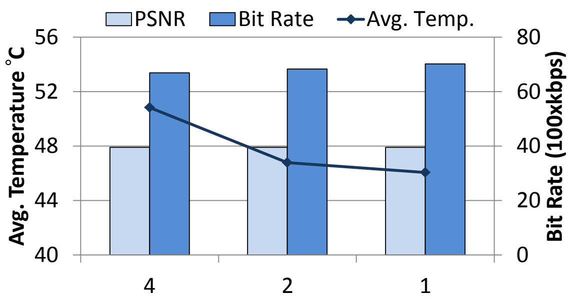
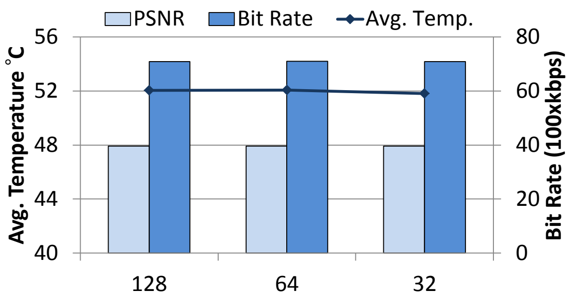


Figure 5.10: Thermal maps for different CTU sizes.

Analyzing the impact of Number of Reference Frames and Search Area on Temperature: the number of reference frames and the search area control the complexity and quality of motion search (especially for objects with repetitive and high motion). However, higher values of these parameters also lead to higher temperature. Figure 5.11 (a) shows the average core temperature, bit rate, and PSNR values for three different selections for the number of reference frames used (1, 2, and 4). When using more reference frames, the temperature of the system is higher. Moreover, the average temperature decreases by 4 ºC when using 2 instead of 4 reference frames at the cost of insignificant quality loss (less than 0.01 dB). Figure 5.11 (b) shows the average temperature, bit rate, and PSNR values for three different search area sizes (128, 64, and 32). Note that for all three different search areas, there is a slight temperature difference (less than 0.5 ºC). This is due to the type of motion estimator used (i.e. EPZS (JCT-VC, 2013) available in the HM software version 11.0) as it does not explore the entire search space and adaptively terminates when a good match is found. Therefore, in this case, changing the search area size does not impact the resulting temperature significantly. However, for a different type of motion estimator, it may change. Figures 5.12 and 5.13 show the thermal maps for the steady temperature considering the different number of reference frames and search area sizes analyzed, respectively.

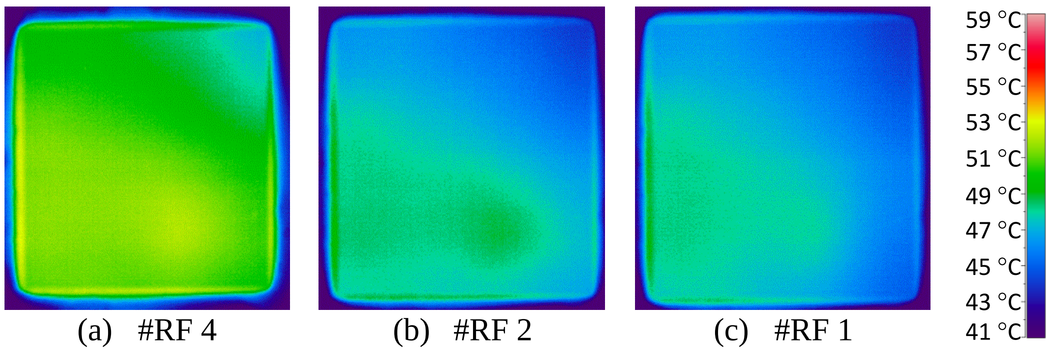


(a) Number of reference frames



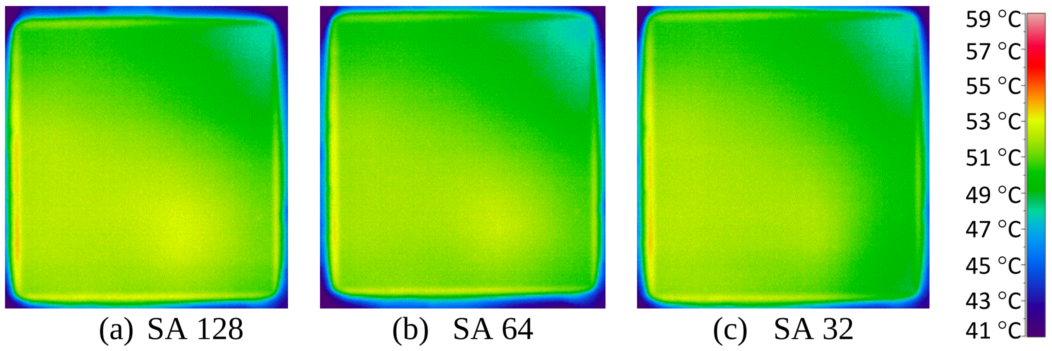
(b) Search area

Figure 5.11: Temperature, bit rate and PSNR for different # RF and SA size.



(a) #RF 4 (b) #RF 2 (c) #RF 1

Figure 5.12: Thermal maps for different number of reference frames.



(a) SA 128 (b) SA 64 (c) SA 32

Figure 5.13: Thermal maps for different search area sizes.

Summary of parameters analysis[4]**:** Encoder parameters determine the thermal behavior (average temperature and temperature variations). The following observations are explored by the first temperature management solution designed in this work.

- Reducing CTU maximum size from 64 to 32 reduces the average temperature by 5 ºC.
- Reducing QP by 5 reduces the temperature by 2 ºC. However, it has a serious impact on the bit rate and PSNR degradation.
- Reducing the number of reference frames from 4 to 2 lowers the temperature by 4 ºC with negligible video quality loss.
- For adaptive motion estimators, changing the search area does not have a significant impact on temperature.

[4] These temperature results are specific for the studied processor, i.e. Intel Atom. For a high frequency processor, the temperature variations between different properties and configuration parameters may even be larger.

5.2 Application-driven dynamic temperature management for video coding

The first temperature solution for video coding proposed in this work is an adaptive thermal management technique for HEVC. Following the main idea of this work the proposed solution raises the abstraction of temperature management to the application level such that video quality degradations can be lowered. This first solution performs (1) application-level temperature prediction; and (2) application-level temperature management. Before explaining these two components of the application-driven solution, we formulate the temperature optimization problem.

5.2.1 Problem formulation

Our temperature solution for video coding accounts for three attributes resulted from the encoding process: (1) T as the CPU temperature in ºC, (2) Q as the video quality in terms of PSNR and (3) B as the resulted bit rate. The main goal of the proposed thermal solution is to minimize QoS (Quality of Service) degradation in terms of bit rate and PSNR, while keeping the current temperature $T_{current}$ under a specified temperature threshold T_{th} as in equation 5.3.

$$T_{current} < T_{th}, \text{ such that, } Max\{Q\} \text{ and } Min\{B\} \tag{5.3}$$

To achieve this goal, the temperature management at the application-level is performed by appropriate selection of the encoding parameters that determines the workload and affects the resulting temperature. In order to perform efficient parameter selection we designed an application-level temperature prediction.

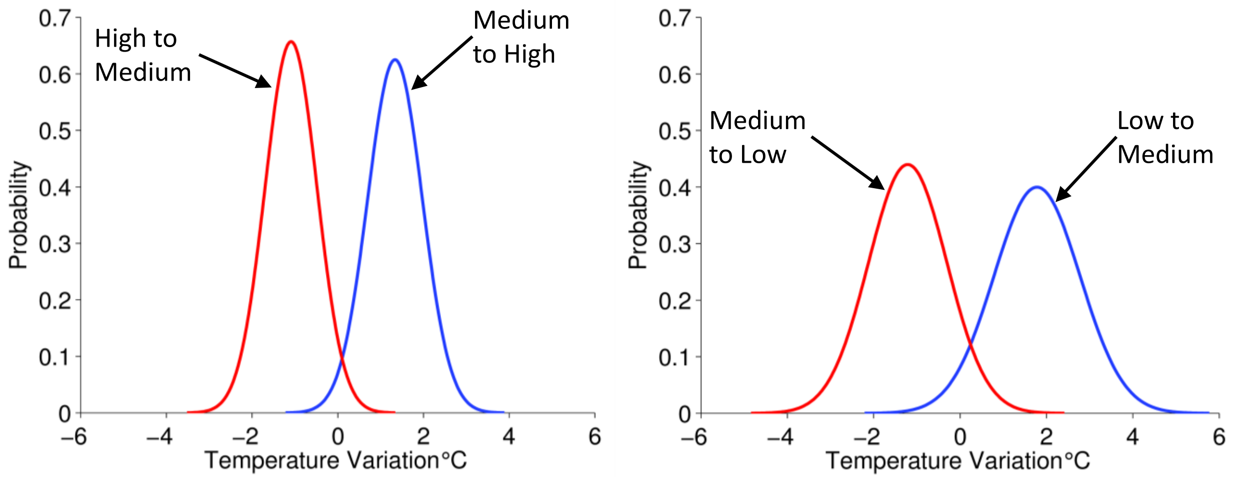
5.2.2 Application-level temperature prediction

Equation 5.4 shows a simple application-level temperature predictor at the frame granularity. For a given time interval (e.g. between the encoding of two frames), the predicted temperature T_{next} (i.e. predicted temperature expected to be obtained after encoding the current frames) can be estimated using: the current temperature $T_{current}$ (temperature at current point in time) and the temperature variation between encoding of current and previous frames.

$$T_{next} = T_{current} + \Delta T \tag{5.4}$$

For obtaining accurate temperature prediction, modeling of the ΔT component in equation 5.4 is the main challenge. As discussed in chapter 5.1.1, the temperature behavior of encoding sequences is highly correlated with sequence content properties. These video properties can be leveraged to obtain hints for temperature prediction. However, using the knowledge of only the current frame may not be sufficient because the temperature distribution is also affected by the content of frames previously encoded (as shown in sub-chapter 5.1.1). Therefore, it is important to account for the complexity class of the current frame and the previous frame, i.e. the complexity difference between two consecutive frames. Considering that, we model the ΔT parameter as a function of the complexity difference between previous and current frames and their respective temperature distributions. Considering the three frame complexity levels (*low*, *medium*, and *high*), we can formulate nine possible temperature variation distributions depending upon the complexity difference between previous and current frames.

Figure 5.14 shows two of the nine possible temperature variation distributions when the complexity between previous and current frame changes (a) from *high-medium-high* and (b) from *low-medium-low*. These temperature variation distributions illustrate that temperature behavior does not follow the sequence complexity structure. It means that if the current frame complexity is higher than that of the previous, the temperature will increase while if the current frame complexity is lower than that of the previous, the temperature will decrease. This complexity variation between previous and current frames provides a good hint for temperature prediction.



(a) high-medium-high (b) low-medium-low

Figure 5.14: Temperature variation distribution between current and previous frames.

Since the standard deviation of theses temperature variation distributions is not zero, using always the same value for ΔT for a given complexity transition at run time can lead to high prediction errors. Therefore, our temperature prediction also uses the error history to improve the prediction accuracy.

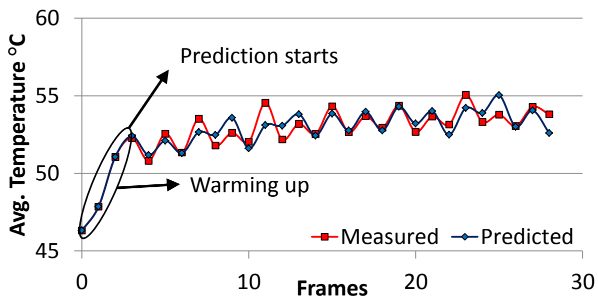
Based on the above discussion, we define our prediction model in equation 5.5. The predicted temperature T_p for the upcoming/current sequence frame is a summation of current temperature $T_{current}$ and temperature variation ΔT. In equation 5.6 the ΔT parameter is calculated using T_v which is the temperature variation between previous and current frame plus the error history. At run time, the T_v parameter is obtained from the probability distribution functions after computing the complexity difference between the current and previous frame (i.e. one out of nine cases as discussed above). The error history is calculated as the mean value of all errors in a time window w (which is the number of previous frames that will be considered to calculate the error) to compensate for the prediction error. Here, the error e is calculated as the difference between the measured temperature T_m (from the temperature sensors) and the prediction temperature T_p (as in equation 5.7).

$$T_p = T_{current} + \Delta T \tag{5.5}$$

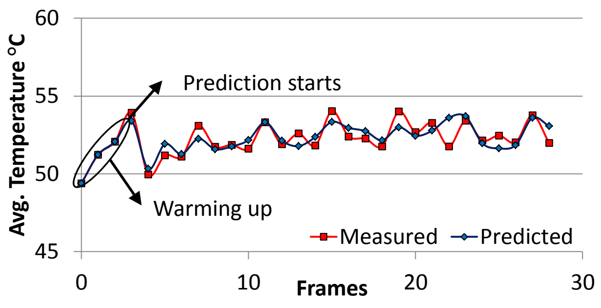
$$\Delta T = T_v + \sum_{i=0}^{w} e_i / w \tag{5.6}$$

$$e = T_m - T_p \tag{5.7}$$

Figure 5.15 shows the accuracy of our application-level temperature prediction for two test sequences *PartyScene* and *Keiba*. For the first three frames the algorithm is not used, since the system is warming up from an idle period, i.e. the temperature increases regardless the complexity level changes between frames. After this warming up period, the prediction starts. It is possible to observe that the predicted temperature (blue line) is indeed close to the measured temperature (red line) for both sequences. In fact the error of our prediction is of only 1.1% on average for the evaluated sequences.



(a) *PartyScene*



(b) *Keiba*

Figure 5.15: Evaluating the accuracy of temperature predictor.

5.2.3 Application-level thermal management

Based on the predicted temperature for the current sequence frame (T_p), our application-driven temperature management solution evaluates if T_p may potentially exceed the safe thermal limit and it reacts accordingly to keep the temperature below the thermal threshold. As discussed in chapter 5.1.2, different sets of encoder parameters will result in different temperature, PSNR and bit rate. Our solution controls the encoder temperature at run time by dynamically selecting an appropriate encoder configuration (i.e. set of encoder parameters) for the current sequence frame, while providing minimum penalties in terms of bit rate and video quality (PSNR) loss. Our solution employs:

(1) *Design-time Pareto analysis of different configurations* for encoding various test video sequences form different complexity classes. The outcome is a set of Pareto-optimal configuration points.
(2) *Run-time configuration selection*: depending upon the predicted temperature, an appropriate configuration is selected from the Pareto-optimal points.

5.2.3.1 Temperature-aware configuration selection

The problem of optimal encoder parameters configuration selection can be solved by Pareto analysis (DAS, 1999) at design-time. Algorithm 5.1 shows how we extract the optimal Pareto curve for each encoder parameter configuration point for a set of test video sequences. For our experiments, only one Pareto curve is provided as input. For each configuration point c_i we encode a test video sequence v_i in order to extract the resulted temperature t_i, bit rate bit_i and PSNR $psnr_i$. Then, with all bit rate and PSNR results mapped into temperature points, we can choose the encoder configuration point that provides the desired temperature d_i maximizing the $psnr_i$ and minimizing the bit_i.

Algorithm 5.1 Extraction of Pareto optimal curve

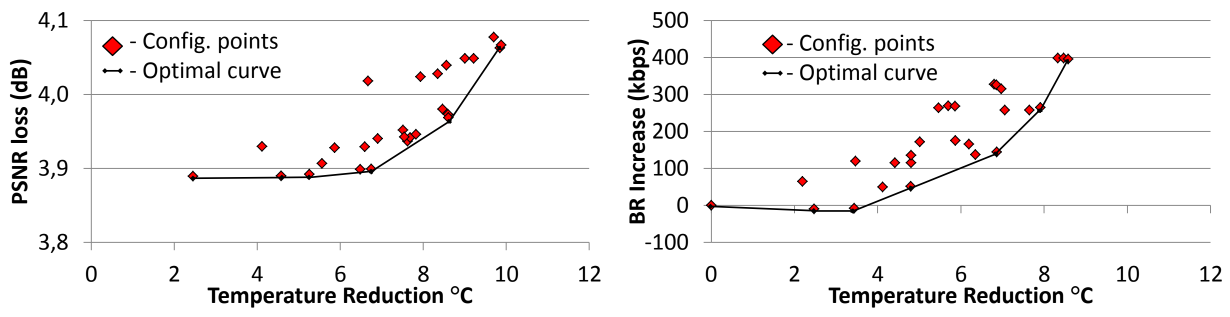
```
Input: Configuration points $\mathbf{C}$, Video Sequences $\mathbf{V}$;
1:  let $\mathbf{T}$ be all temperature points;
2:  for each $v \in \mathbf{V}$ and each $c \in \mathbf{C}$ do:
3:      encode $v_i$ with configuration $c_i$ and get temperature $t_i$;
4:      get $bit_i$ and $psnr_i$;
5:      update point $t_i$ in $\mathbf{T}$ with $bit_i$ and $psnr_i$;
6:  end for;
7:  let $\mathbf{D}$ be the desired temperature points;
8:  for each $v \in \mathbf{V}$ and each $d \in \mathbf{D}$ do:
9:      select $c_i$ while maximizing{$psnr_i$} and minimizing{$bit_i$} to satisfy $d_i$;
10: end for;
```

Since we use real temperature traces and due to time restrictions, we use only a sub-set of the encoder parameters to build our model. Table 5.1 shows the parameter values we adopt. These parameters are well-studied and recommended by the video standardization committee of HEVC (BOSSEN, 2012). Considering the analyzed parameters and all combinations, we are interested in the combinations that optimize temperature reduction with our target attributes (bit rate, PSNR) as shown in Algorithm 5.1.

Table 5.1: Encoder parameters used to build our model.

Parameters	QP	#RF	SA	CTU
Values	22, 27, 32, 37	1, 2, 4	128,64,32	64, 32, 16

Figure 5.16 shows the configuration points space for all combinations of number of reference frames, search area, and maximum size of coding units with a given QP considering average temperature reduction with PSNR loss and bit rate increase for the *RaceHorses* sequence. For other QPs, the curves are scaled. Figure 5.16 (a) illustrates that there are many configuration points decreasing the average temperature but at the cost of high PSNR loss. The same happens in figure 5.16 (b) where two different configuration points can provide the same temperature reduction but with a different bit rate impact. This way, it is possible to achieve good temperature reduction with low penalties in the encoding results by choosing appropriate encoding parameters.



(a) PSNR (b) Bit rate

Figure 5.16: Configuration points for temperature reduction.

5.2.3.2 Run-time adaptive temperature management

Based on the run time predicted temperature T_p for the current frame, our solution dynamically selects appropriate configuration from the design-time Pareto-optimal points. The configuration selection is illustrated in algorithm 5.2. The algorithm starts with initial configuration (line 2) and the current temperature $T_{current}$ is measured from the sensors (line 3). For each frame f, it classifies complexity (line 5) that will be used in the prediction ΔT (line 6-7). After the temperature prediction, it checks if the temperature exceeds the temperature threshold T_{th} (line 8). In case of temperature violation, the algorithm reacts by selecting a new set of configuration parameters c based on the Pareto analysis that ensures temperature reduction with minimum losses in bit rate and PSNR (line 9). Finally, we encode the frame f with configuration c (line 10) updated the current temperature (line 11), the error list (line 13) and the complexity of previous frame (line 14).

It is important to note that, the bit rate and PSNR results are scaled for different sequences but the best configuration points are mainly affected by the encoder parameters. The Pareto-curve is recalculated only if the parameters used to build the model change.

Algorithm 5.2 Video quality-aware Temperature Management

```
Input: Pareto points P, video V, Temperature Threshold T_th;
1:  error_list = [ ];
2:  c = initial configuration;
3:  $T_{current}$ = measure_temperature();
4:  for each frame $f \in$ V do:
5:      $C_{next}$ = classify_complexity($f$);
6:      $\Delta T = T_v(C_{next}, C_{previous})$ + mean(error_list);
7:      $T_p = T_{current} + \Delta T$;
8:      if $T_p > T_{th}$ do:
9:          c = pareto_selection(P, $T_{th}$); //reaction
10:     end if;
11:     encode($f$, c);
12:     $T_{current}$ = measure_temperature();
13:     error = $T_{current} - T_p$;
14:     update_error_list(error);
15:     $C_{previsou} = C_{next}$;
16: end for;
```

5.3 Experimental results and comparison with related works

For evaluating our thermal management solution the input Pareto-curve is obtained from the *RaceHorses* sequence. Therefore, to avoid data biasing, we use four additional sequences (*PartyScene*, *Keiba*, *BasketballDrill* and *BQMall*). Our setup is the same IR-thermal camera based used for the thermal analysis. Figure 5.17 illustrates the distribution of frames in the evaluated sequences considering the proposed complexity classification (*low*, *medium* and *high*). The *PartyScene*, *RaceHorses* and *Keiba* contain *medium-to-high* while the *BasketballDrill* and *BQMall* sequences have *medium-to-low* complexity frames. We evaluate our technique for three threshold temperatures T_{th} 54 ºC, 50 ºC and 46 ºC, as also used by state-of-the-art work (LEE, PATEL and PEDRAM, 2008) showing how our technique adaptively controls the temperature of the video coding system.

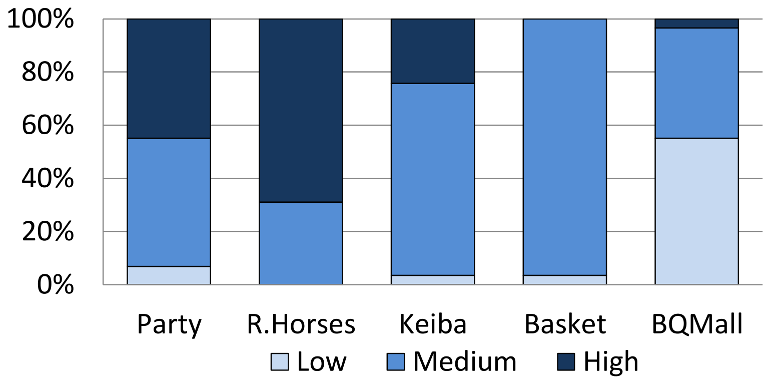
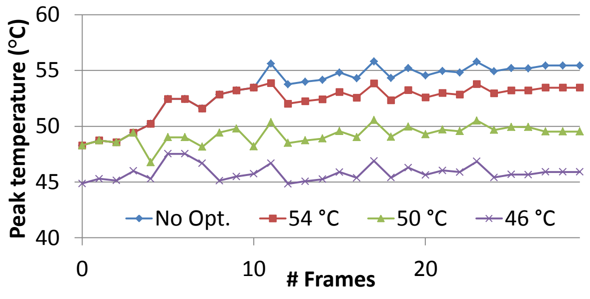
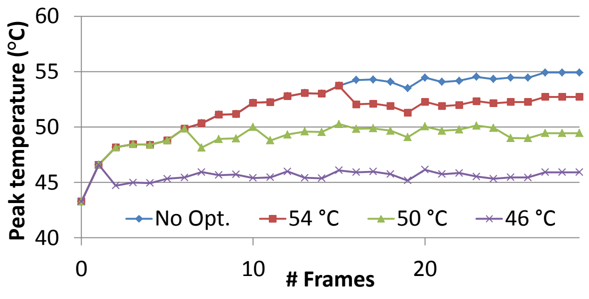


Figure 5.17: Frames complexity distribution of evaluated sequences.

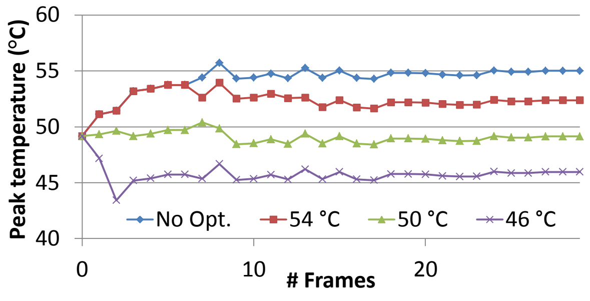
Figure 5.18 (a)-(e) shows the thermal curves during a part of the encoding process for the *PartyScene*, *RaceHorses*, *Keiba*, *Basketball* and *BQMall* sequences in terms of peak temperature-per-frame being encoded to take various scenarios into account. First, we compare our application-driven temperature management technique with the three evaluated temperature thresholds T_{th} values to the "no thermal optimization" case (i.e. the HM default configuration is used in the whole encoding process). It is possible to note that our technique successfully keeps the CPU temperature under the specified T_{th} values by adapting the encoder configurations. As soon as the predicted temperature approaches to T_{th} value, our configuration selection dynamically adapts the encoder parameters in such a way that it decreases the current temperature while minimizing the impact on PSNR and bit rate. For *medium-low* complexity sequences like *BasketballDrill* and *BQMall* it takes more time for increasing the temperature while for medium-high complexity sequences the temperature increases quicker. It means that our temperature management solution will react earlier in case of the high complexity video sequences.



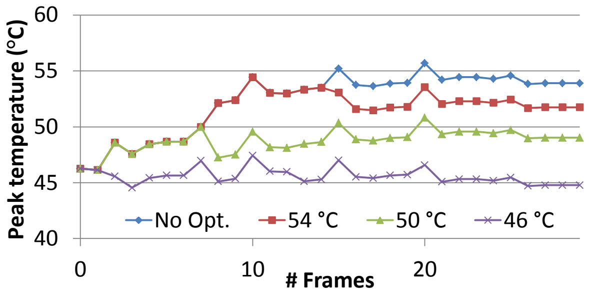
(a) *PartyScene* profile



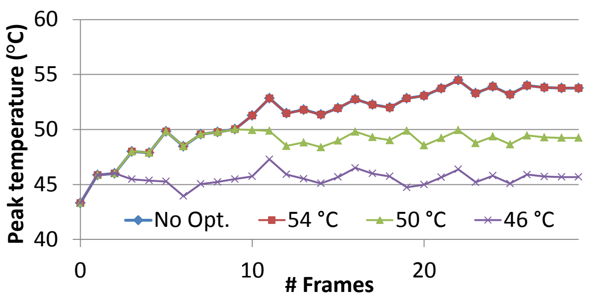
(b) *RaceHorses* profile



(c) *Keiba* profile



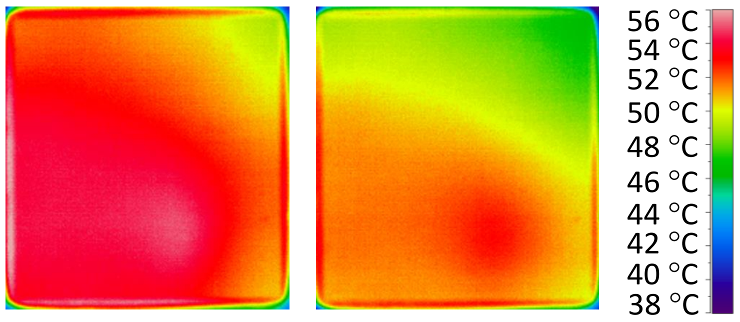
(d) *BasketballDrill* profile



(e) *BQMall* profile

Figure 5.18: Temperature profile encoding different sequences under different thresholds.

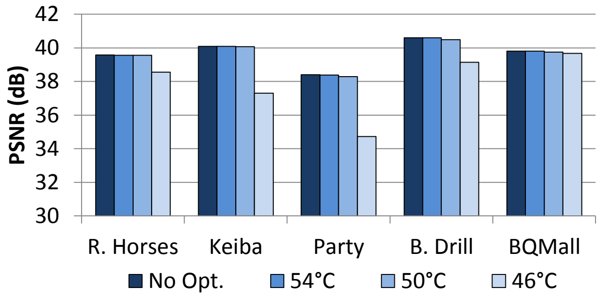
Figure 5.19 compares the thermal maps when using "no thermal optimization" with our thermal management set to 54 ºC for encoding the *RaceHorses* sequence. It is noticeable that when using our solution the temperature of the die decreases since different encoder configuration is chosen to alleviate the computational intensity of the video coding process.

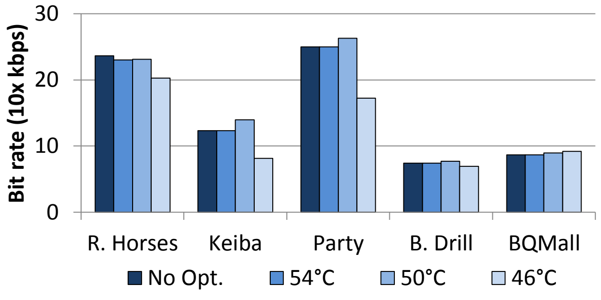


(a) No optimization (b) Our 54 ºC

Figure 5.19: Thermal maps of the die for encoding *RaceHorses*.

Figure 5.20 shows the impact of our temperature management technique in terms of (a) PSNR and (b) bit rate in comparison with "no thermal optimization" for the same five sequences. The degradation in PSNR increases as the T_{th} decreases. However, as our technique is able to perform appropriate encoder parameter selection, the degradation is low. When T_{th} is set to 54 ºC the average PSNR loss is of only 0.007 dB while the bit rate slightly increase 0.99% on average for all sequences. When T_{th} is set to the lowest value of 46 ºC the degradation is higher, but not higher than 1.81 dB in terms of PSNR and 0.84% of bit rate increase on average. Furthermore, for *medium-low* complexity sequences like *BQMall* and *BasketballDrill*, the degradation on coding efficiency is lower for both PSNR and bit rate than for *medium-high* sequences like *PartyScene*, *RaceHorses* and *Keiba*. It occurs, since for reducing the temperature of high complexity sequences, the encoder configuration provided by our solution needs to reduce the workload more than for low complexity sequences.





(a) PSNR (b) Bit rate

Figure 5.20: Coding efficiency results.

We also compare the quality impact of using our temperature management solution with the dynamic thermal management technique proposed by (LEE, PATEL and PEDRAM, 2006) (LEE, PATEL and PEDRAM, 2008). These works use frame drop rates varying from 5.7% to 83.3% depending upon the sequence to keep the temperature under the temperature threshold T_{th}. For comparison purposes we assume that one frame drop implies on using the information of the previous encoded frame in replacement to the dropped frame. For instance, if the frames drop rate is 10% it means that for every 10 frames one is discarded and replaced by the previous frame. This is similar to the state-of-the-art work that employ frame drop on the decoder and copy for display. We establish three frame drop rates, 10%, 20% and 50% to illustrate the negative impact on video quality of using such thermal management technique in comparison with our application-driven solution with threshold set to 54 ºC. Figure 5.21 shows the PSNR for five sequences where our technique achieves better quality results than state-of-the-art for all frame drop rates. The frame drop technique significantly degrades the video quality. On the other hand, our technique optimizes not only the temperature but also the resulted encoder attributes (PSNR and bit rate) through a temperature-aware configuration selection which is a more sophisticated approach compared to naively dropping frames. When the frame drop rate is only 10% the PSNR impact can be of about 12 dB and for 50% the degradation can achieve 20 dB. Such quality degradation is typically intolerable for the users of high-end encoding devices.

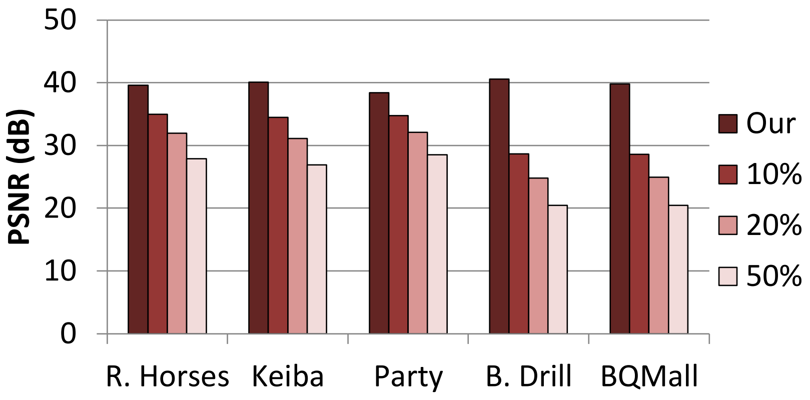


Figure 5.21: Comparison with related works.

Our proposed application-driven dynamic thermal management solution is generic and suitable to any hardware platform, since it is applied at the application-level using video coding characteristics. Since our solution uses information from the encoder itself, such as parameters and luminance samples (to calculate frame complexity), the time overhead in the whole encoding process is negligible. For instance, the texture intensity calculation for the next frame can be performed while the current frame is being encoded.

6 THERMAL OPTIMIZATION USING APPROXIMATE COMPUTING

The second temperature-aware solution proposed in this work is a thermal optimization technique that uses the concept of approximate computing adaptively in order to reduce temperature of HEVC encoding while maintaining good quality results.

Recently, the concept of *approximate computing* has gathered a lot of attention and has been seen as an attractive way to improve performance or power efficiency by compromising the application quality within the tolerable ranges (VENKATARAMANI, CHAKRADHAR, *et al.*, 2015). The basic idea of approximate computing is to explore the application resilience to errors to reduce the total amount of computation at the software and/or hardware level. Several studies have applied approximate computing techniques at different layers of the computing stack, for instance, circuit design (RAMASUBRAMANIAN, VENKATARAMANI, *et al.*, 2013) (GUPTA, MOHAPATRA, *et al.*, 2011), architectures (CHIPPA, VENKATARAMANI, *et al.*, 2014) (CHIPPA, MOHAPATRA and ROY, 2014), and application software (VENKATARAMANI, RANJAN, *et al.*, 2014) (CHAKRADHAR and RAGHUNATHAN, 2010) to reduce power consumption but have not yet explored their impact on the thermal profile optimization. Due to inherent resilience of various functional blocks (like motion estimation) and varying levels of user perception, video coding (HEVC in our case) is a well-suited application for approximate computing and it can tolerate a varying degree of error in the output visual quality after the encoding process.

Earlier works have explored concepts of curtailing mode computations (CORREA, ASSUNÇÃO, *et al.*, 2014), compute effort scaling (CHAKRADHAR and RAGHUNATHAN, 2010), and efficient data management (SHAFIQUE, ZATT, *et al.*, 2012) to trade computational complexity and power consumption with the output visual quality, respectively. However, state-of-the-art works have not yet explored the potential of approximate computing to alleviate the on-chip thermal profiles in complex video coding systems equipped with HEVC.

The main idea of this thermal optimization solution is to adaptively employ approximate computing depending on the video content properties and the application knowledge in order to explore the tradeoffs between on-chip temperatures, computational complexity and visual quality.

6.1 Error tolerance analysis for video coding

We present an analysis of error tolerance for the HEVC encoder application to derive interesting observations that are leveraged for designing an efficient content-aware approximate computing technique for video coding. Imprecise or approximate computing at the application layer can exploit several basic methods to improve power efficiency by selectively eliminating operations/computations at the algorithm and data levels.

6.1.1 Opportunities of approximate computing on video coding

As presented in chapter 3.3.1, the HEVC encoding process is structured as recursive quad-tree partitioning of the basic Coding Tree Unit (CTU – 64x64 pixels). However, the recursive process does not need to go always down to the deepest 8x8 division. Actually, in many cases, depending upon the CTU content properties (for instance, homogeneous blocks with low motion) the 64x64 size is sufficient to have a good coding efficiency. In case, the recursive partitioning stops at the bigger CU size (i.e. it does not go to the deepest level in the quad-tree), the computation complexity of the coding process can significantly be reduced as all other coding steps will only be applied over the generated partitions. However, the coding efficiency-wise best partition may not exist in the set of generated partitions, which may result in degraded coding efficiency.

For the selected partitions, the coding process is performed by exploiting the spatial and temporal correlations through so-called *Intra-* and *Inter-Predictions*. While the inter-prediction processes typically employs matching functions like sum of absolute differences (SAD) (see chapter 3.4), the intra-prediction process employs prediction generation FIR (Finite Impulse Response) filters considering the neighboring pixels. These matching and prediction generation functions are composed of primitive pixel-level operations (like addition, subtraction, and multiplication), which can be approximated at different levels of granularity (ranging from circuit (GUPTA, MOHAPATRA, *et al.*, 2011), loop perforation (SIDIROGLOU-DUOSKUS, MISAILOVIC, *et al.*, 2011) to data approximations (SHAFIQUE, ZATT, *et al.*, 2012)) while still achieving reasonably good prediction results. For instance, figure 6.1 shows different possibilities of applying the data approximations, where the **x:y** notation means that for a set of **y** pixels only **x** are used in the calculation of the matching and prediction functions. Such data approximations can perturb the algorithm and may lead to different prediction values and coding modes, and thereby affecting the coding efficiency results. Aggressive and poor guided data approximations may lead to serious video quality degradation.

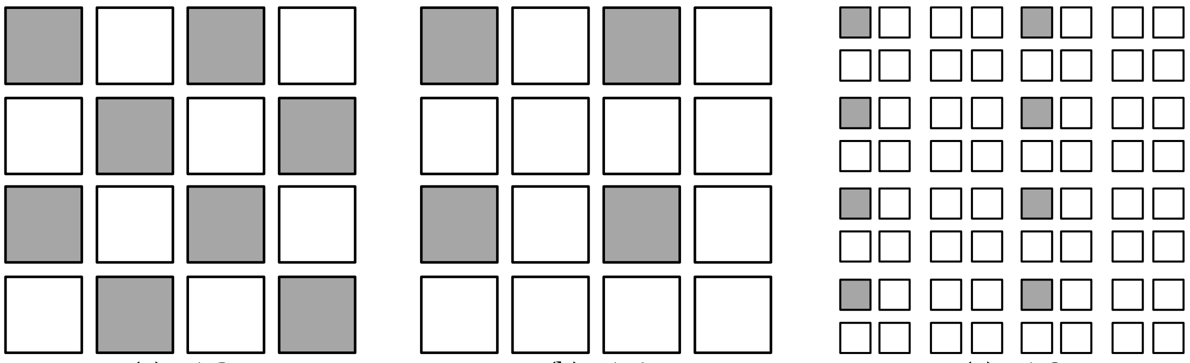


Figure 6.1: Example of possible data approximations (PALOMINO, SHAFIQUE, *et al.*, 2016).

6.1.2 Analyzing the error tolerance of HEVC under different application-level approximations

Without the loss of generality, we employ two basic application-level approximate computing method to the HEVC: (1) *Loop Perforation* (SIDIROGLOU-DUOSKUS, MISAILOVIC, *et al.*, 2011) to prune the quad-tree coding structure; and (2) *data approximation* through pixel sub-sampling during the prediction process. Using this we define four different approximation modes (*AM-0* to *AM-3*) in order to evaluate the error tolerance of video sequence regions. Table 6.1 shows the type of approximations used for each mode. *AM-0* corresponds to no approximation, while *AM-3* corresponds to the most aggressive approximation mode. The approximations at the algorithm level are represented by the maximum quad-tree depth. It limits the depth that the quad-tree coding structure will divide the 64x64 CTU. The approximations at the data level are represented by the pixel sub-sampling from 1:1 to 1:8. Each approximation mode is a composition of a particular algorithm-level approximation and a particular data-level approximation, and it will be applied to every CTU in a frame of a given video sequence. Note, in the generic form, an approximation mode can be defined as a particular combination of different hardware/software-level approximations form a set of basic approximation methods.

Table 6.1: Algorithm and data approximation modes.

Approximate Mode	**Maximum Quad-Tree Depth**	**Data Sub-Sampling**
AM-0	4 (until 8x8)	1:1
AM-1	3 (until 16x16)	1:2
AM-2	2 (until 32x32)	1:4
AM-3	1 (until 64x64)	1:8

Considering the above presented approximation modes, we have performed a set of experiments to analyze the error tolerance of different test video sequences (BOSSEN, 2012). Table 6.2 shows the sequences used in this error tolerance evaluation of HEVC. HEVC test Model (HM) software version 16.0 (JCT-VC, 2015) was used for encoding the test video sequences. The approximation modes are applied CTU by CTU for all video sequences. From these experiments, final visual quality and computational complexity reduction are obtained. The resulting output visual quality is the metric used to illustrate the error tolerance of a given sequence while computational complexity reduction is used as the abstract metric to show the potential for improving the temperature profile, as temperature is a function of the workload.

Table 6.2: Sequences used for error tolerance evaluation.

Sequences	**Resolution (pixels)**
BQMall	832x480
BasketballDrill	832x480
RaceHorses	832x480
BQTerrace	1920x1080
BasketballDrive	1920x1080
Cactus	1920x1080

In the first experiment, we have encoded all sequences using all approximation modes described in table 6.1 for all CTUs in order to check the approximation effects on both visual quality and computational complexity. Figure 6.2 shows the quality results of this experiment in terms of BD-PSNR loss (BJONTEGARD, 2001) (in dB) after applying the approximation modes *AM-1*, *AM-2*, and *AM-3* compared to the *AM-0* mode (i.e. when no approximation are performed).

It is noticeable that the quality loss increases as the approximation modes go further in the quad-tree pruning and in the sub-sampling degrees achieving almost 1.5 dBs of quality loss for the *AM-3* applied to the *RaceHorses* sequence. The algorithm-level approximation in the quad-tree limits the CTU division in region with high motion/texture, where encoding with smaller blocks is essential to get good quality results. Also, the data level approximations imposed by the sub-sampling affects the quality, mainly in heterogeneous regions, where each pixel contributes in a different way to the matching function calculation. Moreover, there is a different ratio in the quality loss between sequences, for example, the *BQTerrace* sequence is less affected by the approximation modes in comparison with the *RaceHorses* sequences. This happens, since different video sequences have different amount of motion/texture properties and different correlation potential (see chapter 5.1.1).

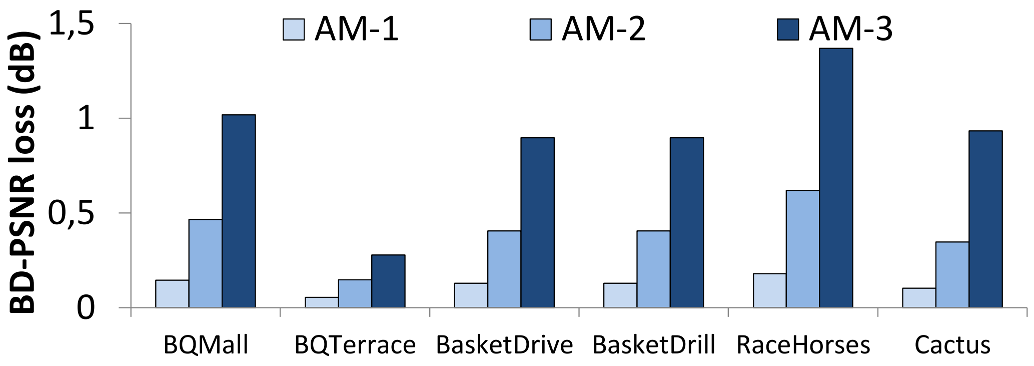
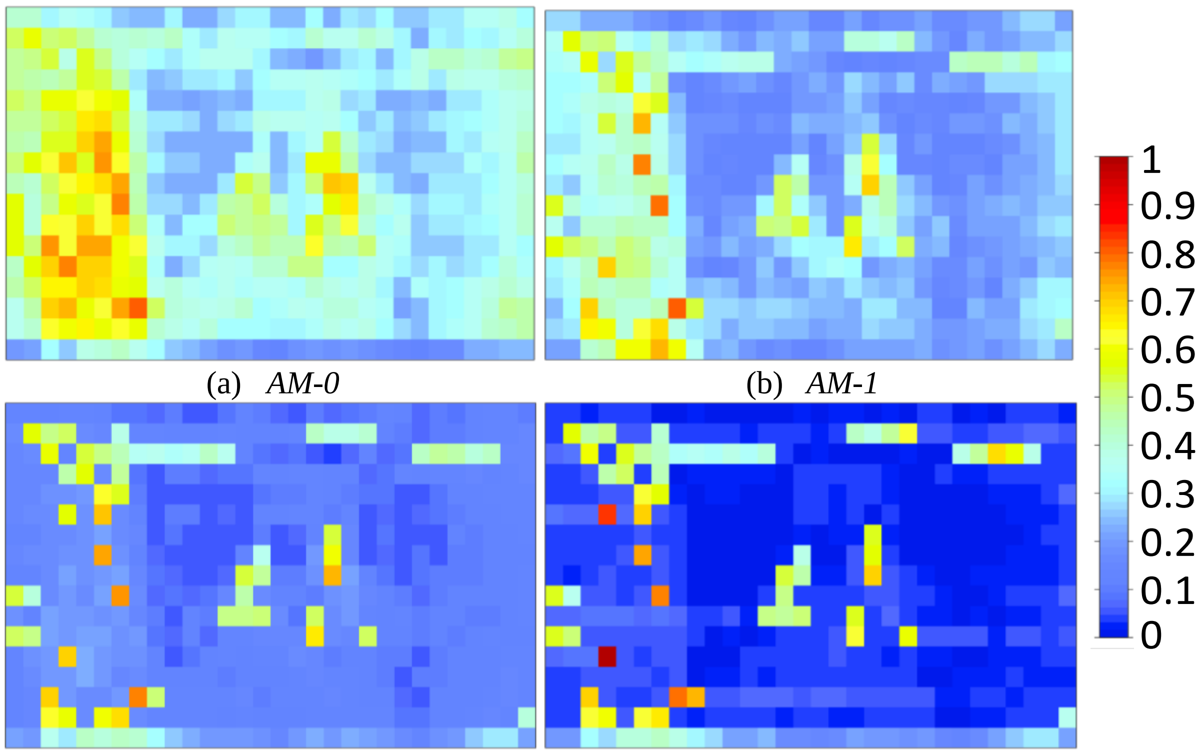


Figure 6.2: Quality results for algorithm/data approximate computing modes.

Figure 6.3 shows the impact of using the approximation modes on the computational complexity of encoding one frame (second frame) of the *BasketballDrive* sequence. The workload map is formed by squares that represent the time (normalized to the range of the CTU times) to encode each CTU in the frame considering all approximation modes.



(a) *AM-0* (b) *AM-1*

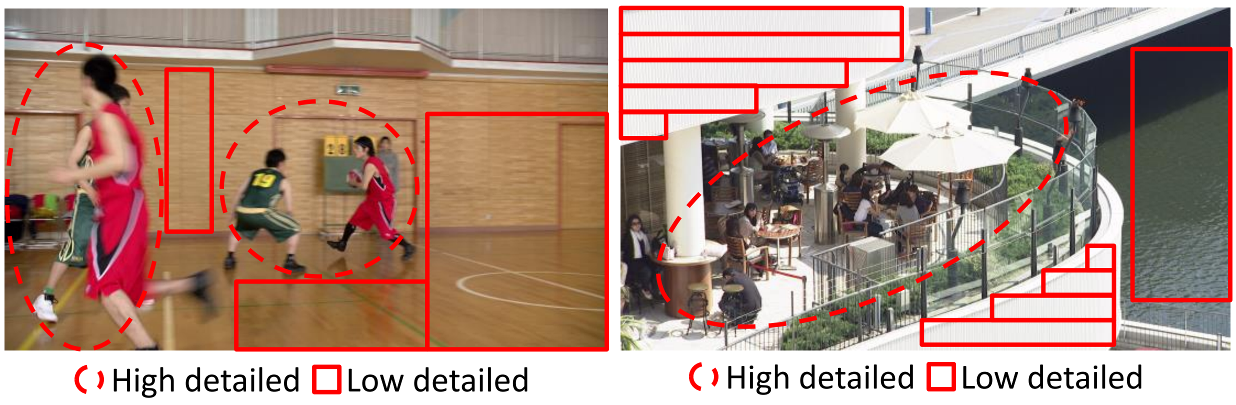
(c) *AM-2* (d) *AM-3*

Figure 6.3: CTUs workload encoding the *BasketballDrive* (second frame) sequence for all Approximation Modes.

Following the quality loss results, the workload reduces as the approximation modes prune more the quad-tree structure and the sub-sampling degree increases. The algorithm-level approximations by eliminating some of the quad-tree depths denote the avoidance of the processing of all coding tools used to evaluate each one of the eliminated blocks. Besides, the sub-sampling is applied to the blocks that were not eliminated by the algorithm-level approximations. It is important to note that even with the approximation modes, the workload of some CTUs do not reduce in the same proportion. In fact, in some cases the CTU workload increases as demonstrated in some of the CTUs when the *AM-3* is applied (figure 6.3 (d)). This happens, since other coding tools can negatively be affected by the quad-tree pruning, and the computational complexity of some algorithm routines may increase (like CABAC Entropy Coding) to achieve better encoding.

Although using the proposed approximation modes for all CTUs seem to be a good way to reduce the workload associated with the video encoding process and consequently to improve the thermal profiles, there are significant quality losses in most of the tested sequences. When the *AM-3* mode is applied, there is about 1 dB of BD-PSNR loss for all sequences except the *BQTerrace* and almost 1.5 dB for the *RaceHorses* sequence. When the *AM-1* mode is applied the BD-PSNR losses are very low. On the other hand, the total workload reduction may not be sufficient to reduce the temperature of the encoding system. Therefore, it is necessary to find a better way to use these approximation modes where good quality results can be achieved together with low temperatures.

An interesting observation can be found in the case of *BQTerrace* quality results in figure 6.2. The BD-PSNR losses are very low even when the *AM-3* mode is applied. This happens since this sequence have more low texture/motion regions than the other sequences, and therefore, using both algorithm and data-level approximation do not impact the output quality much. This denotes that there are regions in a video frame that are more resilient to the approximation errors while there are some other regions that are more sensitive to the approximations. Figure 6.4 shows two frames form two sequences (*BasketballDrive* and *BQTerrace*) where we have identified some low/high detailed areas that can be more or less affected by the approximations explored in this chapter. The low and high detailed regions are selected based on the texture and motion properties of the objects in a frame.



(a) *BasketballDrive* (b) *BQTerrace*

Figure 6.4: High/Low detailed regions (1920x1080 pixels).

In order to support the hypothesis that applying the approximation modes selectively to some characterized resilient video regions can be less harmful to the final output quality while still contributing to the workload reduction, we have performed a second set of experiments.

In these experiments, we selected regions with low texture/motion properties for applying the three approximation modes and encoded two frames shown in figure 6.4.

Figure 6.5 shows the BD-PSNR losses where the approximation modes being applied to all regions in the frames are compared to applying the approximation modes only to those regions characterized as low detailed (i.e. red rectangles in figure 6.4). Note that the BD-PSNR losses are very low (close to the zero value) for both sequences. Additionally, even when the *AM-3* mode is applied, the BD-PSNR losses stay below 0.1 dB, which is almost imperceptible. Our analysis shows that high workload reduction can be achieved at low quality degradation if all three approximation modes can be applied selectively to only the resilient regions in the video sequences. We refer to this as "*content-driven adaptive approximate computing*".

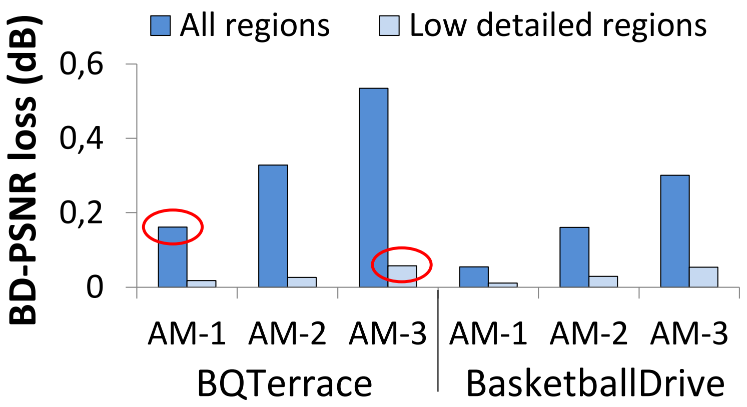


Figure 6.5: BD-PSNR losses comparison.

Figure 6.6 shows the normalized workload reduction of applying the approximation modes only in the low-detailed regions and compares it to the workload of applying the approximation modes to all regions. It shows that even when the approximation modes are applied only to the low-detailed regions, the workload reduction is significant. Considering the *AM-3* mode in the low-detailed regions, the *BQTerrace* sequence shows workload reduction close to the *AM-1* mode (red circles in figure 6.6) with less than half of the BD-PSNR losses (red circles in figure 6.5). Moreover, there is a potential of more workload reduction if all the approximation modes could be applied to all frame selectively depending upon the properties of their different regions.

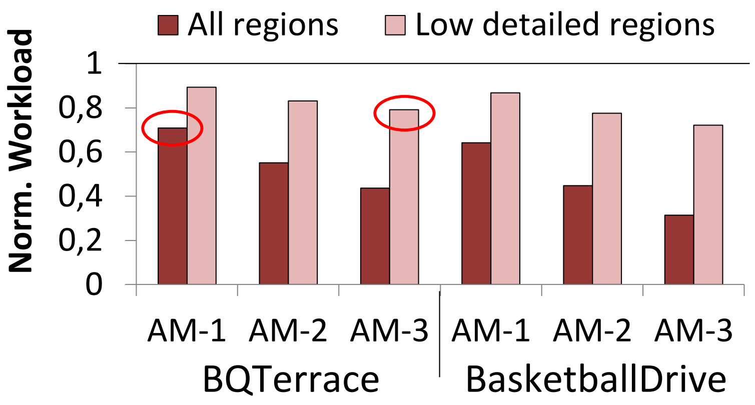


Figure 6.6: Workload reduction comparison.

Summary of analysis: Approximate computing at the algorithm-level by pruning the quad-tree coding structure and at the data-level by sub-sampling the matching function calculation can provide significant workload reductions in the HEVC process. However, in order to improve the tradeoff between workload reduction (and consequently the on-chip temperature) and the resulted visual quality, different degrees of approximations (i.e. different approximation modes) need to be applied adaptively according to the error tolerance/resilience of different video regions. The resilience of a video region can be formulated as a function of its texture and motion properties.

6.2 Thermal optimization through adaptive approximate computing

The main idea of this thermal solution is to adaptively employ different approximation modes to reduce the temperature associated with the HEVC video coding process. The main goal of this technique is to minimize both temperature and application QoS degradation in terms of BD-PSNR. To achieve such a goal, the temperature optimization at the application-level is obtained through a *content-driven approximate computing technique* that exploits video properties to classify the resilience of different regions of the video sequence in order to control the selection of approximation modes during their encoding.

6.2.1 Error Resilience Classification

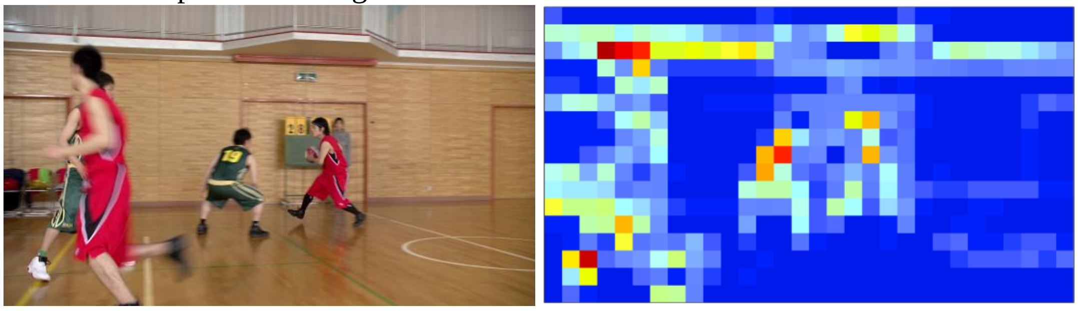
As demonstrated in chapter 6.1, some video regions exhibit higher resilience to approximation than others, i.e. tolerate more error and incur low visual quality loss. It shows that if regions can be characterized in terms of their error resilience, it is possible to obtain high workload reduction at low quality penalties. Our analysis showed that, in general, regions with low motion/texture intensity (low-detailed and homogeneous regions) are the ones with higher resilience to the approximation errors. On the other hand, regions with high motion/texture intensity (high-detailed and heterogeneous regions) are more sensitive to the approximation errors. Therefore, we formulate the resilience of video regions as a function of their texture and motion properties.

Towards this end, we first estimate the texture and motion intensity of different CTUs of the input video sequence. For each 64x64 CTU, we calculate the variance among the luminance samples within the CTU as in equation 6.1 using the model of (SHAFIQUE, ZATT and HENKEL, 2012). Then, we compute the normalized variance of all CTUs within a frame using equation 6.2, since the most important information from the variance calculation is to compare regions in terms of their error resilience level and identify the ones with the high resilience value. It is important to point out that the variance calculation overhead is less than 0.1% (BLUMENBERG, PALOMINO, *et al.*, 2013).

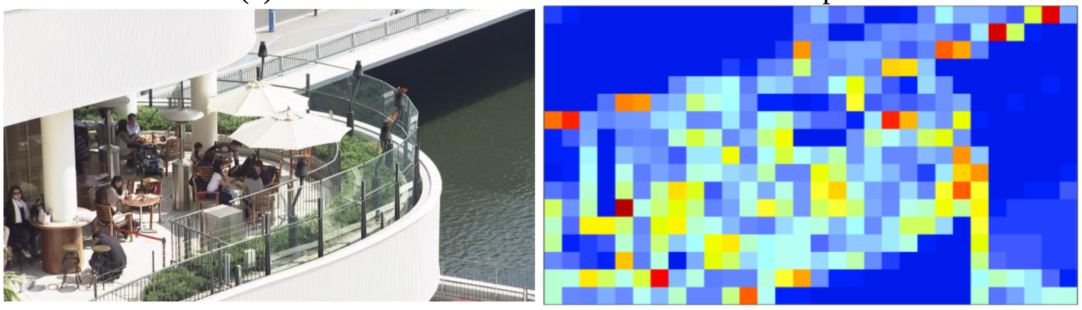
$$v_{CTU} = \frac{1}{4096}\sum_{i=1}^{4096}\left(\rho_i - \rho_{avg}\right)^2 \qquad (6.1)$$

$$norm\_v_{CTU} = \frac{v_{CTU} - minFrame(v_{CTU})}{maxFrame(v_{CTU}) - minFrame(v_{CTU})} \qquad (6.2)$$

Figure 6.7 shows a color map where each square represent the normalized variance value for each CTU for the second frames of *BasketballDrive* and *BQterrace* sequences. It is possible to observe that the low-variance values indicate low texture/motion regions while high-variance values are found in regions where there is more detailed texture/motion information. Note, most of the low-variance regions also match with the selected regions of the second experiment in figure 6.4.



(a) *BasketballDrive* normalized variance map.



(b) *BQTerrace* normalized variance map.

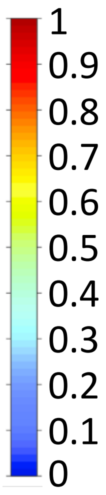


Figure 6.7: Texture/motion color maps using variance.

Based on this CTU-level normalized variance information, we define the following four resilience levels to classify each CTU using the equation 6.3: *Resilient*, *Medium Resilient*, *Medium Sensitive*, and *Sensitive*. This classification will be used for characterizing different CTUs in a video frame and for selecting an approximation mode during the CTU encoding. Without the loss of generalization and for illustrative purposes, we choose four different resilience levels in order to match with our four approximation modes used for the evaluation of the proposed concepts.

$$\Gamma = \begin{cases} Resilient & if\,(norm\_v_{CTU} < Th_{v1}) \\ Medium\ resilient & if\,(Th_{v1} \le norm\_v_{CTU} < Th_{v2}) \\ Medium\ sensitive & if\,(Th_{v2} \le norm\_v_{CTU} < Th_{v3}) \\ Sensitive & if\,(norm\_v_{CTU} \ge Th_{v3}) \end{cases} \tag{6.3}$$

The threshold values in equation 6.3 were obtained through regression analysis (considering a step size of 0.05) using a small subset of test video sequences, which are different from the set of test video sequences used for evaluation, in order to avoid data biasing. For defining the threshold value Th_{v1} corresponding to the level *Resilient*, test video sequences were encoded with the *AM-3* mode in the regression analysis. The threshold selection was performed considering the quality degradation behavior and the workload reduction in the regression tests. Similar methodology was adopted for obtaining the other two threshold values Th_{v2} and Th_{v3}. Table 6.3 shows the final thresholds obtained through the regression analysis providing the best tradeoff between workload reduction and visual quality loss.

Table 6.3: Threshold values used to classify CTU resilience.

Thresholds	Th_{v1}	Th_{v2}	Th_{v3}
Values	0.1	0.2	0.3

6.2.2 Content-driven adaptive approximation management

Based on the proposed resilience classification of CTUs in the HEVC encoding process, our technique uses the obtained threshold values in table 6.3 to define the degree of approximations that will be employed for each CTU encoding. Algorithm 6.1 shows the flow of our light-weight approximate mode selection heuristic. For each frame in a video sequence, the variance of each CTU is extracted and added to a variance list *v_list* (line 1 to 5). Then the normalized variance value is calculated for each CTU to classify the CTU resilience to approximation errors (line 8). According to the classification, one of the four approximation modes is selected for the CTU encoding process (line 10 to 13). For CTUs that are classified as *Sensitive*, no approximations are performed in the CTU encoding.

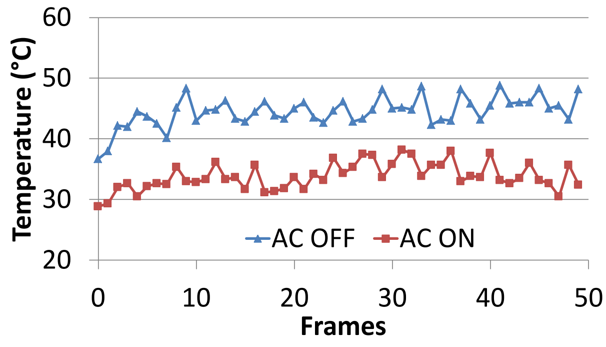
Algorithm 6.1 Approximate mode selection heuristic.

```
Input: sequence S;
1:  v_list = [ ];
2:  for each frame $f$ Є S do:
3:      for each CTU $cu$ Є $f$ do:
4:          $v_{CTU}$ = extract_variance($ctu$);
5:          update_v_list($v_{CTU}$);
6:      end for;
7:      for each CTU $ctu$ Є $f$ do:
8:          norm_$v_{CTU}$ = [$v_{CTU}$ – $min$(v_list)] / [$max$(v_list) – $min$(v_list)];
9:          case norm_$v_{CTU}$:
10:             resilient: encode($ctu$, AM-3);
11:             medium resilient: encode($ctu$, AM-2);
12:             medium sensitive: encode($ctu$, AM-1);
13:             sensitive: encode($ctu$, AM-0);
14:         end case;
15:     end for;
16: end for;
```

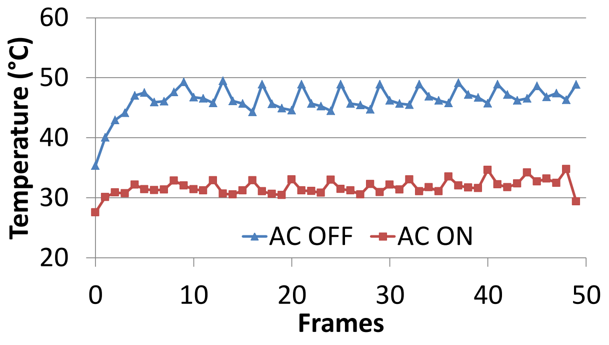
6.3 Experimental results and comparison with related works

For evaluation, we perform thermal simulations when encoding different sequences using HEVC while considering two methodologies. The first methodology uses the digital thermal sensor (DTS) based setup (see chapter 4.3) and the second methodology uses the tool chain setup (see chapter 4.2) to generate thermal maps. All temperature results for both methodologies were collected using the HEVC test Model (HM) software version 16 (JCT-VC, 2015). The variance threshold values were obtained from the *BQTerrace* and *BasketballDrive* sequences. Therefore, to avoid data biasing, we use five additional sequences to evaluate the quality results of our content-driven approximate computing technique and to include a wide range of scenarios considering diverse texture/motion intensity.

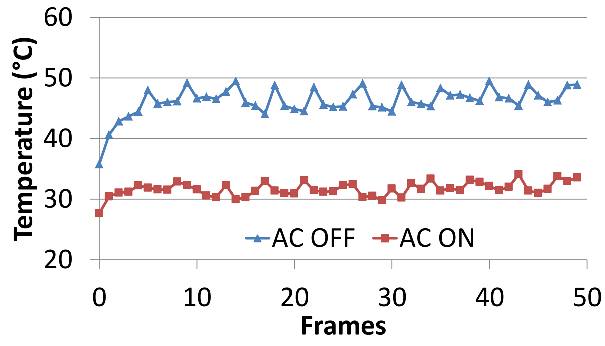
Figure 6.8 shows the thermal profiles extracted using the DTS methodology for encoding four sequences with and without our approximation technique. It can be observed that the temperature values are smaller when our technique is employed. This happens since there is a significant workload reduction in the encoding process provided by our technique. For the *BasketaballDrill* sequence, our technique did not achieve temperature reduction as good as that for the *BasketballDrive*. This can be attributed to the high number of low texture/motion CTUs (i.e. more resilient regions) in *BasketballDrive*, and thereby using more aggressive approximations. In contrast, for *BasketballDrill*, due to the more high-textured CTUs, less aggressive or no approximations were employed when encoding those CTUs.



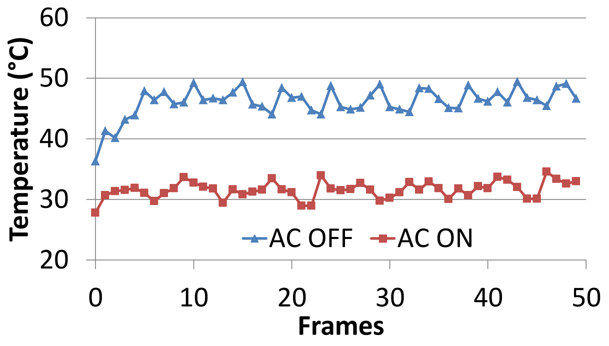
(a) *BasketballDrill*



(b) *BasketballDrive*



(c) *BQTerrace*



(d) *Cactus*

Figure 6.8: Temperature profile for different sequences.

Figure 6.9 the average temperature results for all tested sequences using the DTS methodology. Our content-driven adaptive approximate computing technique achieved low temperatures for all sequences in comparison to the regular HEVC encoding without any approximations. It can be observed that our technique significantly improves the thermal profiles (e.g. 15 ºC reduce temperature in *Cactus*), though there are some sequences where the average temperature reduction is only 5 ºC (*BQMall*). On average, for all sequences, our technique achieved 10 ºC of temperature reduction. The difference in temperature reduction among different sequences is attributed to their diverse distributions of low-resilience CTUs that limit the applicability of aggressive approximations.

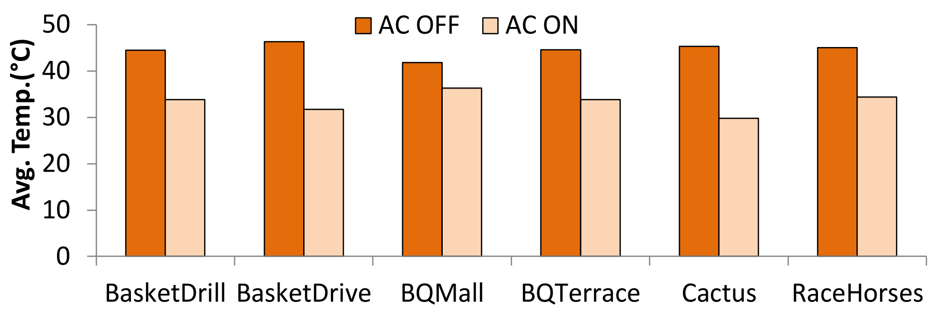


Figure 6.9: Average temperature for all tested sequences.

Figures 6.10 and 6.11 show the thermal maps extracted from the second evaluation methodology based on the tool chain for two sequences. From these figures it can be observed that our technique successfully improves the temperature profile of video coding systems. The left-side hotter maps represent the steady-state temperature when using the regular HEVC encoding without any approximations, while the right-side colder maps represent the steady-state temperature of the HEVC encoding when using our technique, In summary, the temperature reduction for all sequences using the tool chain methodology was also about 10 ºC on average.

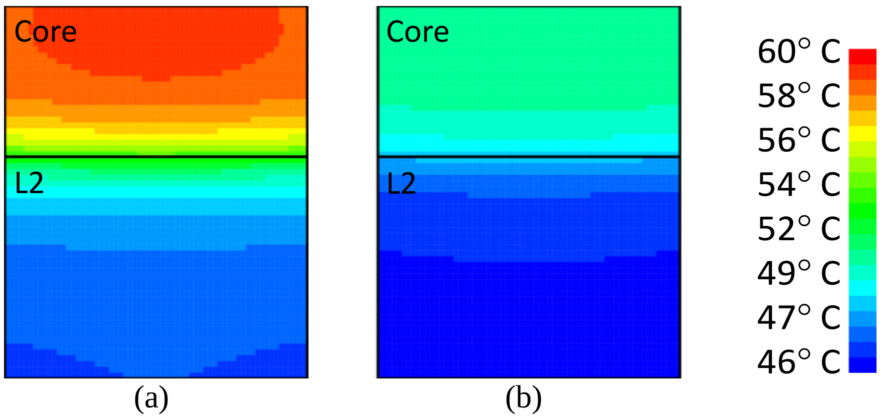


Figure 6.10: Thermal maps of *BasketballDrill* (a) AC OFF (b) AC ON

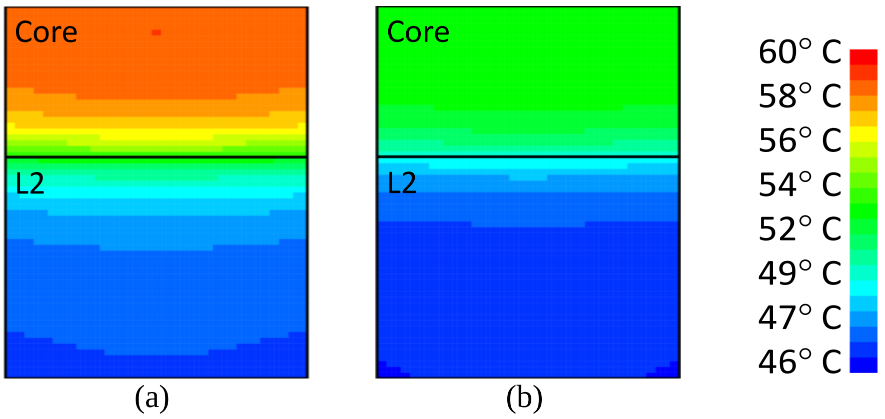


Figure 6.11: Thermal maps of *BQMall* (a) AC OFF (b) AC ON

Figure 6.12 show the impact of our technique on the output quality loss (in terms of BD-PSNR) when compared to the regular HEVC encoding without using our technique. The quality degradation for all tested cases is on average 0.48 dBs (excluding sequences that were used on training the threshold values). Even for the worst-case results (i.e the *RaceHorses* sequence), our technique incurs a maximum BD-PSNR loss of 0.66 dBs, while in the best case (i.e the *Keiba* sequence) it incurs the maximum BD-PSNR loss of only 0.31 dBs. We also compared the BD-PSNR losses of our technique with statically using the *AM-3* mode (the most aggressive approximation mode with the highest workload savings). For all cases our technique presents better output quality results than employing only the *AM-3* mode, which is attributed to the resilience-driven adaptive approximation control.

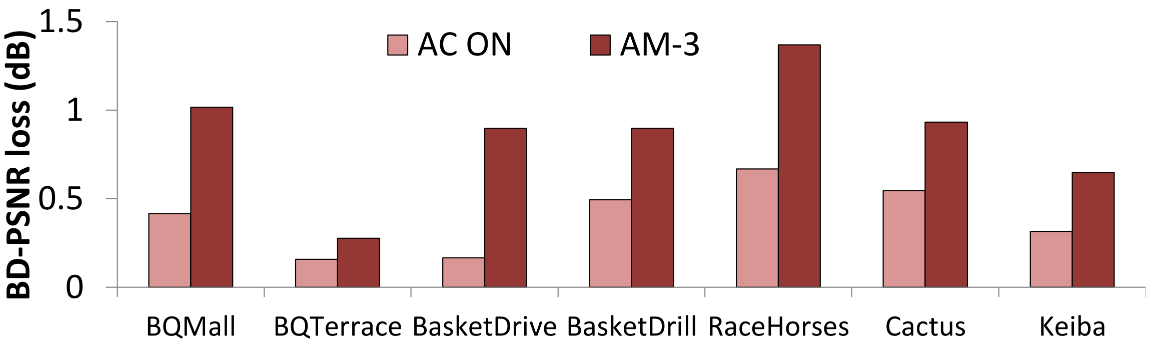


Figure 6.12: Quality results.

We compare quality results of our approximate computing based temperature optimization technique with the dynamic thermal management policy proposed in (LEE, PATEL and PEDRAM, 2006) (LEE, PATEL and PEDRAM, 2008) that use frame drop rates to control temperature. Again, we consider that a frame drop mean that the encoder replaces it by the information of the last-encoded frame and skips the encoding process. We also compare quality results with the application-driven dynamic thermal management proposed in the previous chapter (PALOMINO, SHAFIQUE, *et al.*, 2014a) (PALOMINO, SHAFIQUE, *et al.*, 2014b) that uses run-time encoder configuration to keep the temperature under a defined threshold. The quality results are reported corresponding to the same temperature reduction cases, e.g., 10 ºC reduction.

Figure 6.13 shows the quality comparison in terms of PSNR for encoding four sequences with the HEVC. Our technique is able to outperform previous works in most of the evaluated cases. When comparing with the frame drop based technique, our approximate computing based technique present the best results in all cases. The quality degradation of dropping 20% of the frames leads to a PSNR loss of 8 to 16 dBs. This significant quality loss is usually not tolerable for users on encoding devices. When comparing with thermal management policy that change encoder configurations at run-time, the approximate computing technique present better results for three sequences and similar result for the *BQMall* sequence. This happens because there are some encoder parameters, such as the QP, that can significantly degrade quality. Higher QP values introduce error in the encoding process, generating quality loss. When applied to sequence regions that are too sensitive to errors, it may increase the quality losses. On the other hand, the approximate computing based technique takes into account the texture and motion properties of different sequence regions and adaptively employs varying degree of approximations. Consideration of resilience properties of different regions help in achieving better quality results than other technique that do not account for such information.

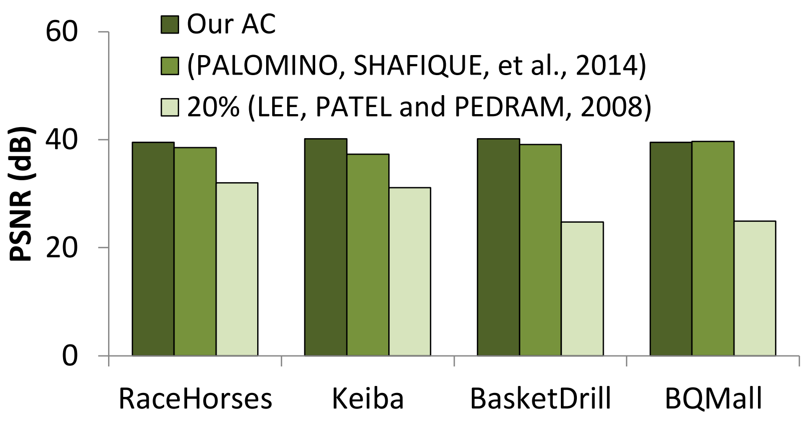


Figure 6.13: Comparison of quality results with related works.

7 APPLICATION-DRIVEN THERMAL-AWARE SCHEDULING

The third temperature solution for video coding proposed in this work is an application-driven thermal aware scheduling technique for on-chip multicore systems executing multi-threaded workloads in order to deal with possible spatial temperature variation that has negative effects on the system reliability (see chapter 2.1).

The main idea of multicore architectures is to provide a hardware platform where applications can run in parallel. The application performance is improved when the total data to be computed is equally split into threads. While threads are completely independent, they can run in different cores at the same time and the speed up provided by the parallel processing should be proportional to the number of threads. However, this is not true for applications where the computation of threads is affected by the characteristics of the data and not only for the amount of data being processed. Video coding is an example of such application where thread's workloads are not proportional to the amount of data. Additionally, this behavior will directly impact in the thermal profiles of a multicore system running the video coding application. In order to measure this workload behavior of the multi-threaded video coding and the thermal impact considering a multicore scenario, we first perform an extensive analysis following the tool chain methodology (chapter 4.2).

7.1 Analysis of threads workloads

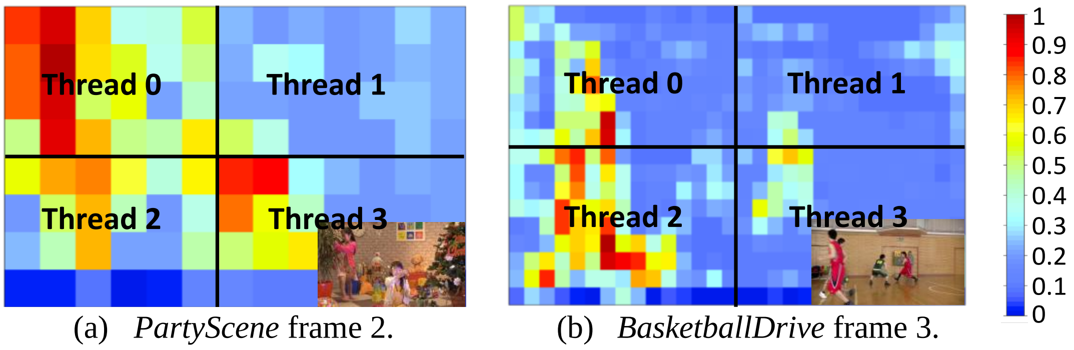
We have performed several analysis related to the unbalance nature of workload between threads for video coding. All analyses were performed considering the HEVC test Model software version 16. The workload associated with each thread is measure in terms of encoding time. Then, we have performed an analysis to measure how the workload unbalance nature of the video coding application impacts on the thermal profiles of multicore systems. We have analyzed three different topologies varying the number of cores (2, 4 and 8 cores).

7.1.1 Thread's workloads in video coding systems

The efficiency and computational complexity of advanced video coding tools are basically driven by the properties of the input video. The main principle of video coding tools is to search for similarities among the sequence content across different frames. The computational complexity associated to encode a particular region of a sequence is highly correlated to properties such as motion and texture intensity. As extensive discussed in previous chapters, regions with low presence of details and low motion intensity between frames usually are less complex to encode than highly detailed regions with high motion intensity between frames. However, this aspect is not considered when splitting data of video sequences for parallel encoding.

In state-of-the-art video coding standards like HEVC, by default, the pixels in one frame are equally split between threads in a matrix fashion (number of rows and columns) for parallel encoding. While the data to be processed is balanced for all threads, the workload associated with encoding each thread may not be due to the video content properties within each thread. This way, the variation of workloads between threads will be highly dependent on the content being encoded. Moreover, this workload variation will affect the thermal profile of the multicore system where the application is running on.

Figure 7.1 (a) and (b) show color maps of normalized workload between threads considering a 2x2 division for two frames of different sequences, frame 2 of *PartyScene* (832x480 pixels) sequence and frame 3 of *BasketballDrive* (1920x1080 pixels) sequence. Each colored square refers to the relative workload for a single coding tree unit (CTU) (64x64 pixels). For the *PartyScene* sequence it is possible to see that the workload of thread 0 is more computational intensive than all other threads, while thread 1 presents the lowest workload. The same behavior is noticeable for the *BasketballDrive* sequence, where the workload of thread 2 is higher than others, while thread 1 is less computational intensive. Even though the number of CTUs is as equally divided as possible among the threads, the workload is not balanced between them. This happens due to the direct relation between data properties and resulted workload. Regions such as thread 0 in sequence (a) and thread 2 in sequence (B) have more motion and/or texture intensity than the other regions. This way, they will demand much more computation from the encoder than the other regions that are easier to encode.

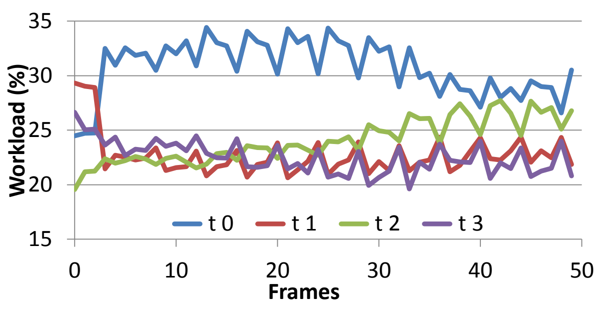


(a) *PartyScene* frame 2. (b) *BasketballDrive* frame 3.

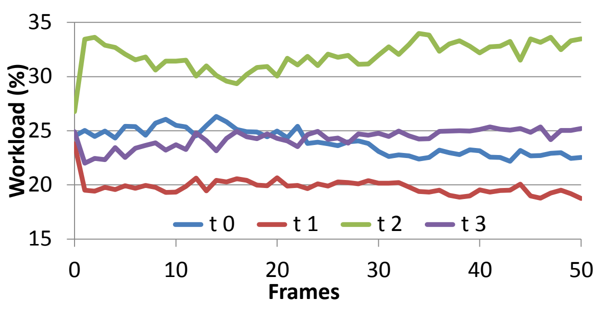
Figure 7.1: Workload maps of threads in the video encoding process.

The workload difference between threads when encoding one frame is noticeable due to the data content in each thread. In real video scenarios only some objects are moving across the sequence frames and this is the characteristic that drive this unbalance workload nature of threads when video are being encoded. This behavior continues along all frames, since motion and texture intensive regions will usually occur in different areas of the sequence frame. We have performed an analysis along 50 frames to show that this workload difference continues happening. Figure 7.2 (a) and (b) show the thread's normalized workload when encoding the same two sequences considering a 2x2 division.

For the *PartyScene* sequence figure 7.2 (a) thread t0 is much more computational intensive than the other threads, while in the *BasketballDrive* sequence, thread t2 presents higher workload. The highest workload region in one frame can change its location according to the sequence scene motion. However, this unbalance workload behavior will continue existing between the threads, since there will always be regions with higher motion/texture intensity than other regions.



(a) *PartyScene* 832x480.

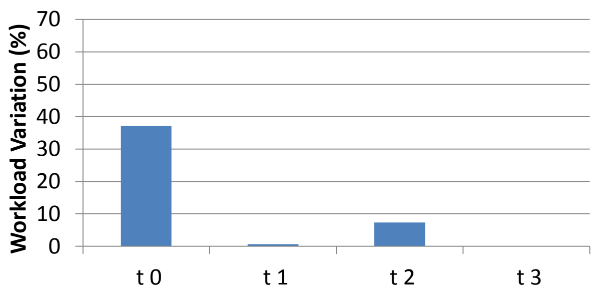


(b) *BasketballDrive* 1920x1080.

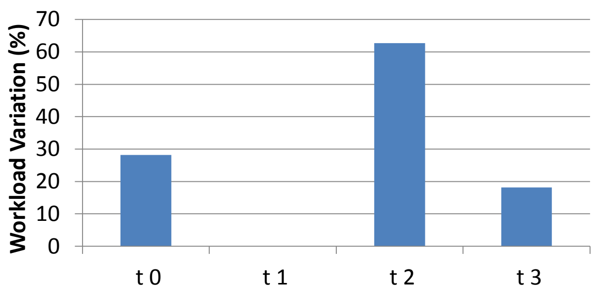
Figure 7.2: Workload behavior of threads for various frames.

Another important aspect is to know how much this thread's workloads can differ between each other. Figure 7.3 (a) and (b) show the workload variation between thread for the same two sequences (*PartyScene* and *BasketballDrive*), where the bar graphs are normalized in the less computational intensive thread (thread t3 for *PartyScene* and thread t1 for *BasketaballDrive*). This analysis was performed considering the same scenario in figure 7.2, i.e. workload different along encoding of 50 frames.

While the most computational intensive thread t0 for the *PartyScene* sequence can be almost 40% more computational intensive than thread t3, for the *BasketballDrive* sequences thread t2 is more than 60% more intensive than thread t1. The workload difference between the threads achieved 60% considering simulation performed with only 4 threads (2x2 split). When more threads are used these workload differences will be even higher, since the computation intensity of high motion/texture regions will be more concentrated within a thread. It is important to note that this workload difference between threads will directly impact the thermal profile of the hardware platform where the video coding application will be running in parallel.



(a) *PartyScene* 832x480.



(b) *BasketballDrive* 1920x1080.

Figure 7.3: Workload variation between threads.

7.1.2 Workload difference impact on thermal profiles

Applications such as video coding have deadlines associated with the tasks. For instance, in a real time video broadcasting, the encoding job must finish before a defined deadline to keep the frame rate (number of frames per second) constant for the video being transmitted. Considering a parallel video coding scenario, the thread's deadlines will be defined considering a proportional part of the time window per frame based on the number of parallel thread and number of processing units being used for the encoding. This means that threads with high difference in the workload intensity will be more or less computational intensive along the deadline window. Moreover, this variation of workload within the deadline window will impact in the thermal profiles of the parallel hardware platform where the application is being processed.

Following the Tool Chain setup, we have evaluated sequences encoding with two (2x1), four (2x2) and eight (2x4) threads for the *BasketballDrive* sequence. Figure 7.4 shows the accumulated workload map for encoding 50 frames using two threads assigned to a two cores chip. The thermal map shows how the workload difference between the threads affects the temperature profile of the chip. It is noticeable the temperature difference between core 0 and core 1. In fact, for the steady temperature this difference is of 10 ºC. Figure 7.5 shows the accumulated workload maps when using four threads for encoding the video sequence mapped to a four core chip. In this case, thread 0 and thread 2 are the most computational intensive. This reflects in the thermal profile of the cores running these threads, since core 0 and core 2 achieve 81 ºC as maximum temperature while core 1 and core 3 achieved 74 ºC and 71 ºC respectively. Finally, Figure 7.6 shows the last evaluated case where the sequence is divided into eight threads that are mapped to an eight core processor. Once more, the most computational intensive threads (thread t0 and t4) generate high temperature differences across the chip. Core 0 and core 4 achieves as maximum temperature 83 ºC and 82 ºC while other cores temperature are below 73 ºC.

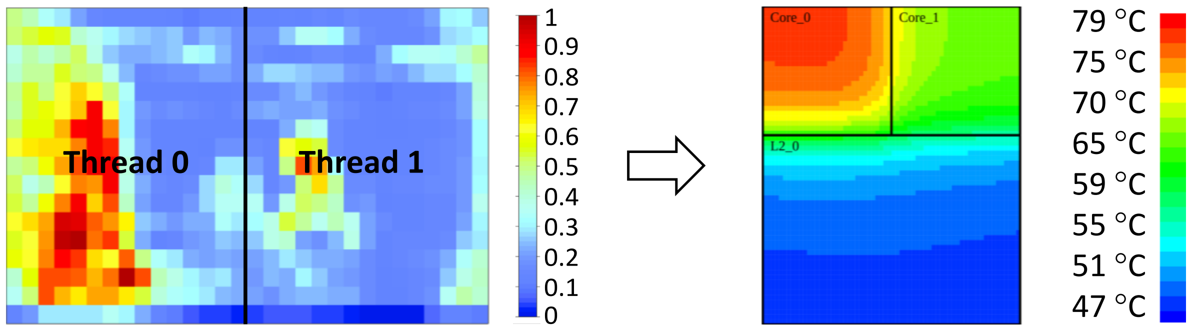


Figure 7.4: *BasketballDrive* two threads two cores.

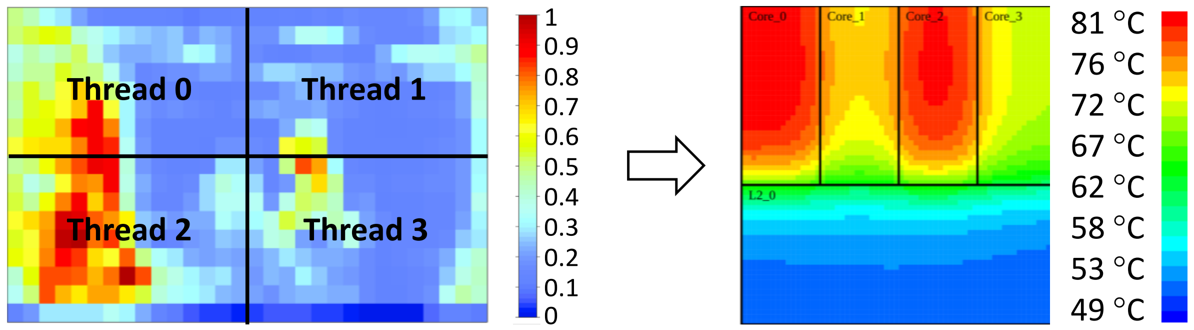


Figure 7.5: *BasketballDrive* four threads four cores.

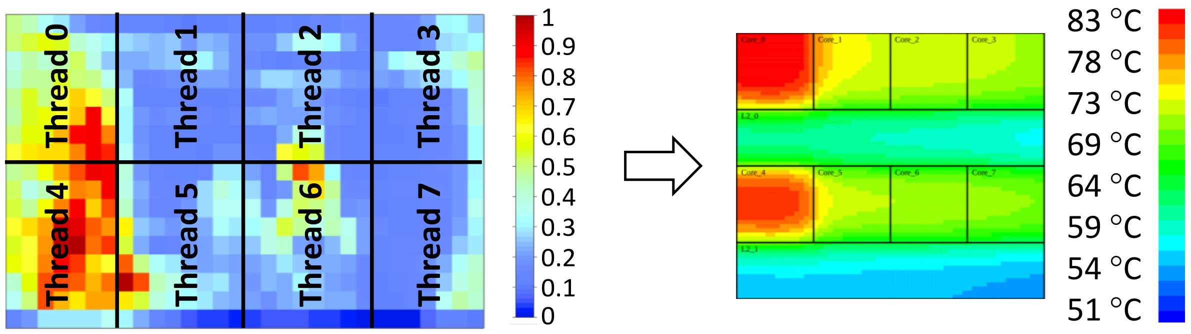


Figure 7.6: *BasketballDrive* eight threads eight cores.

In summary, the unbalanced workload nature of the multithreaded HEVC video coding application directly impact in the temperature profile of the hardware platform running the encoder. Threads that have different workload distribution will generate unbalanced thermal profiles with high spatial temperature gradients that are not desirable when reliability issues are considered. As threads workload distribution are mainly affected by the properties and characteristics of the video sequence, this information can be used to appropriate schedule the threads of new frames to the cores in a way to reduce the spatial temperature gradients and consequently the overall temperature of the chip.

7.2 Application-driven scheduling scheme

In this chapter we present the proposed application-driven temperature-aware scheduling scheme for multithreaded workloads that uses the application knowledge and the current temperature status of cores for assigning threads to cores in a multicore hardware platform. Figure 7.7 provides an overview of the proposed scheduling scheme. Current chips typically contain several thermal sensors, and these sensors data can be continuously read and written into memory to be processed by the operational system. Also, on the application level, the workload associated with the thread is predicted and passed to the application-driven scheduler. Then, the scheduler can use both application knowledge and current thermal status to guide the assignment of each thread in a temperature-aware fashion.

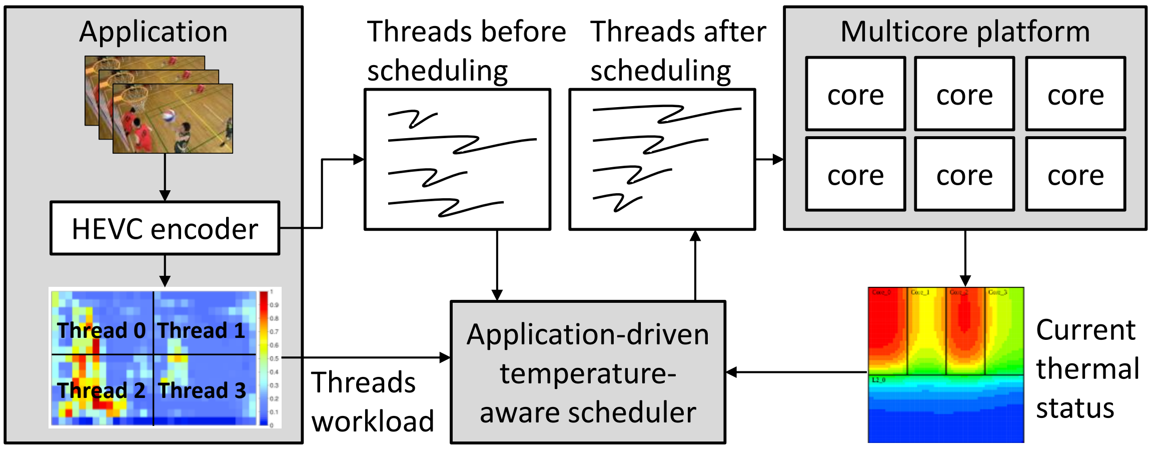


Figure 7.7: Scheduling scheme overview.

In the following, we provide details of the proposed scheduling scheme. First we present the application-level workload prediction and how data and application characteristics can provide potential for a temperature-aware scheduler. Finally, we present our temperature-aware scheduler technique.

7.2.1 Problem formulation

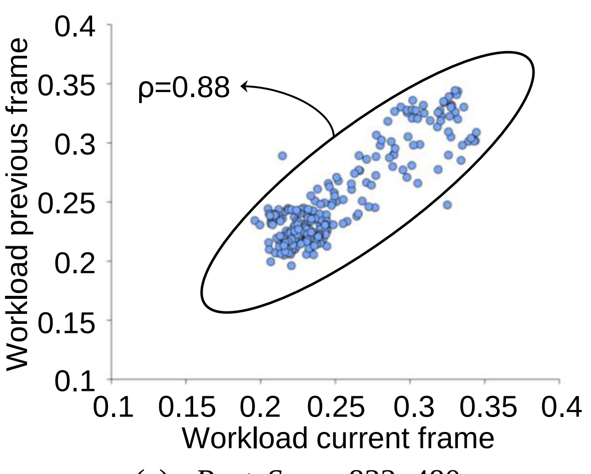
The temperature-aware scheduling problem can be characterizes as follows. Given a set of n cores $C = \{c_0, c_1, \dots, c_n\}$ and a set of m threads $T = \{t_0, t_1, \dots, t_m\}$, the main goal of our scheduling scheme is to assign all threads in T to all cores in C in order to minimize the spatial temperature gradients, i.e. the temperature difference between the hottest and the coldest cores. Also, with this strategy, we expect that the maximum overall temperature of the chip also reduces.

To achieve this goal, the temperature-aware scheduling scheme assigns each thread to each core considering the set of current temperature status of each core as $TS = \{ts_{c0}, ts_{c1}, \dots, ts_{cn}\}$ and also the set of workloads associated with each thread $W = \{w_{t0}, w_{t1}, \dots, w_{tm}\}$ in order to balance the temperature distribution across the chip. The set of workloads W is extracted based on an application-level prediction model that uses video characteristics to accurate predict the workload of each thread.

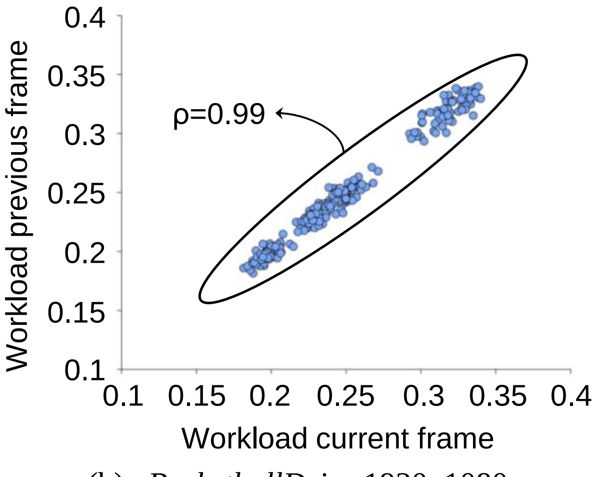
7.2.2 Application-level thread workload prediction

The design of an efficient temperature-aware scheduler requires a good workload predictor, since the workload associated with threads is the most influent aspect to define the thermal behavior of the system. For application such as video coding, the thread's workloads are highly dependent on the video characteristics. Also, this dependence is the main reason for workload differences between threads, which significantly affect the temperature profiles. For instance, the thread's workloads are known only after the encoding process. However, it is possible to predict the workload of each thread looking to the video properties associated with each part of the video being encoded. The main goal of the proposed application-level thread workload prediction is to know how the workload of encoding on frame is distributed across the threads based on the video characteristics and application knowledge to support an appropriate scheduling of threads in the multicore hardware platforms.

Nowadays high resolution videos are captured with 30-60 frames per second. From this characteristic, one important video property arises: two consecutive frames in a sequence are likely to be very similar. Actually, temporal content correlation is one important property that is exploited in most of the video coding tools. Moreover, it means that the information generated after encoding one frame may be useful to predict the next frame encoding behavior. Considering a multi-threaded video coding application, the workload associated with each thread being separately encoded tends to be similar between consecutive frames. We have performed an analysis in order to measure this temporal content correlation in terms of thread workload between two consecutive frames, i.e., the workload of threads collocated in the same position between two consecutive frames are compared. Figure 7.8 shows the results of this analysis for two sequences with four threads per frame, the axis are the normalized measured workload for each thread in the previous frame (axis y) and the measured workload for each collocated thread in the current frame (axis x).



(a) *PartyScene* 832x480.



(b) *BasketballDrive* 1920x1080.

Figure 7.8: Correlation between threads workload of consecutive frames.

It is possible to see that there is a strong correlation between the measured workload of threads considering two consecutive frames. For instance, we have calculated the linear correlation coefficient ρ for five sequences and it ranges from 0.88 for the *PartyScene* sequence to 0.99 for the *BasketballDrive* sequence. This temporal content correlation between two consecutive frames provides a good hint to predict how the workload of one frame is distributed among the threads for the next frame.

Even though there is a strong correlation between the thread's workload of two consecutive frames, it is important to notice that this information is not enough to have a robust workload prediction for each thread. Either objects moving through the scene or the camera moving may change abruptly the workload distribution among each thread of the video being encoded. This way, it is necessary to consider this possible workload distribution variation between frames in order to have an accurate workload prediction for each thread. Also in this case, analyzing video properties variation between frames can lead to a good workload variation prediction.

As previously discusses, texture is one important video property regarding its influence in the workload intensity in the video coding process. Again we deploy the technique proposed by (SHAFIQUE, ZATT and HENKEL, 2012) that uses variance of the luminance samples to extract texture information from video frames. Equation 7.1 shows how the thread texture level tl is extracted of frames being encoded where m and n are the thread horizontal and vertical limits in number of coding tree units (CTUs). Equation 7.2 shows how the variance is calculated for a single coding (64x64=4096 pixels). The texture difference between threads in the current frame and collocated threads in the previous frame is other good hint to track workload variation between thread for two consecutive frames.

$$tl = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} v_{CTU} \tag{7.1}$$

$$v_{CTU} = \frac{1}{4096} \sum_{i=1}^{4096} (\rho_i - \rho_{avg})^2 \tag{7.2}$$

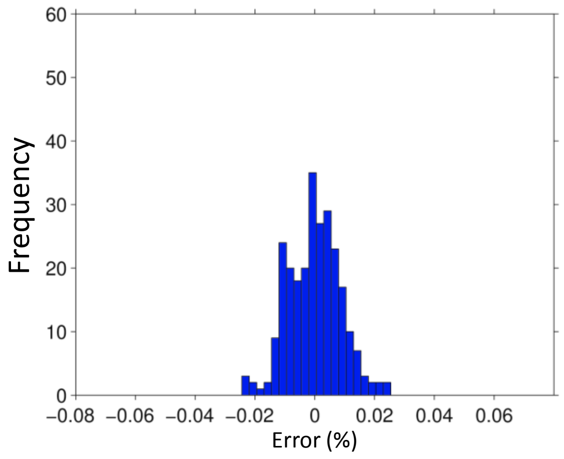
Based on these both premises: (1) there is strong temporal content correlation between two consecutive frames; and (2) measure texture variation between two consecutive frames is a good hint to track workload variation between frames; we define our application-level thread workload prediction as follows.

Given a set of workload measures as the temporal content correlation information $WM = \{wm_{t0}, wm_{t1}, \dots, wm_{tm}\}$ of the previous frame and a set of texture levels of the current frame $TLC = \{tlc_{t0}, tlc_{t1}, \dots, tlc_{tm}\}$ and texture levels of the previous frame $TLP = \{tlp_{t0}, tlp_{t1}, \dots, tlp_{tm}\}$ each wp_{ti} in the set of workload predictions $WP = \{wp_{t0}, wp_{t1}, \dots, wp_{tm}\}$ is predicted using a content dependent function f as in equation 7.3.

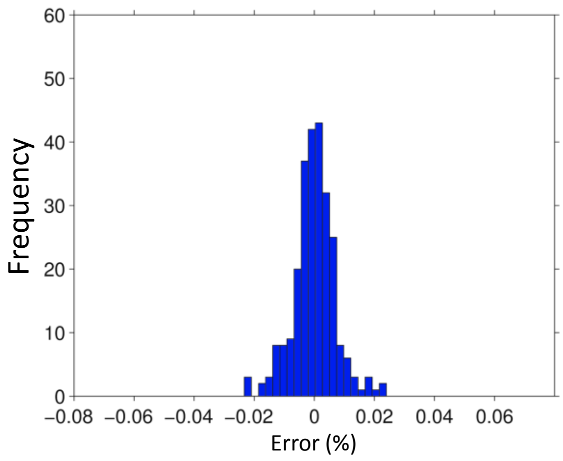
$$wp_{ti} = f(WM, TLC, TLP) = wm_{ti} + \Delta tl_{ti} \tag{7.3}$$

$$\Delta tl_{ti} = tlc_i - tlp_i \tag{7.4}$$

The f function uses temporal content correlation and texture variation between two consecutive frames as in equation 7.4 to predict the workload of each thread in the video coding process. We have performed a simulation with other two sequences (*BQMall* and *RaceHorses*) using our content dependent workload prediction in order to see the error of the prediction using this approach. Figure 7.9 presents the error histograms in percentile when using our workload prediction. The error is measured comparing the predicted workload of each thread in one frame in terms of percentage and the measured workload after the encoding. From the histograms, we can conclude that using temporal content correlation and texture variation between two consecutive frames provides accurate workload prediction considering threads for a parallel video coding application.



(c) *BQMall* 832x480.



(d) *RaceHorses* 832x480.

Figure 7.9: Content dependent workload prediction error.

7.2.3 Temperature-aware scheduler

Based on the prediction of the workload distribution among threads and the current temperature status of the cores, the proposed application-driven temperature-aware scheduler will react to minimize the spatial temperature gradients of the multicore chip. Our scheduler is always monitoring two variables of the encoding system. The first one is the workload distribution among threads being encoded that is given by our application-level temperature predictor and the second one is the temperature status of the cores from the hardware sensors. As our scheduler focuses on minimizing the spatial temperature gradients across the chip, the difference between the hottest and the coldest cores is the attribute we want to minimize. Algorithm 7.1 summarizes our temperature-aware scheduler that works in two main steps.

1) **Monitoring variables:** the video coding process is performed frame by frame. Before on frame encoding, our scheduler is interested on acquiring the workload distribution across the threads that will be used to perform the parallel encoding processing. It means that our content dependent prediction is performed for each thread in a frame to predict the workload intensity wp_{ti} associated with this thread. The content dependent function f uses the set of measured workloads from the previous frame WM, and the texture variation extracted from the current and the previous frame TLC, TLP (line 4-10). Additionally, the cores temperatures are also monitored to check the temperature difference between the core with highest temperature and the core with lowest temperature (line 11-14). This difference is the current spatial temperature gradient $current\_STG$ that is compared to a threshold STG_{th} value to decide whether or not the scheduler should react to this temperature difference (line 15).

2) **Reacting to large spatial gradients:** our scheduler is triggered when the $current\_STG$ is higher than a pre-defined threshold limit. The scheduler, then, uses the accurate workload prediction information of each thread to assign them to the cores in a temperature-aware fashion. The scheduler policy is to send the highest workloads to the coldest cores in order to improve temperature balance among the cores. The threads are sorted considering its

workload intensity in the frame form low to high. Also, the cores are sorted from the hottest core to the coldest core. Finally, the scheduler is responsible to assign each thread to one core based on the thread's workloads and the current temperature status. The scheduler starts assigning the least intensive thread to the hottest core to the most intensive thread to the coldest core (line 16-20).

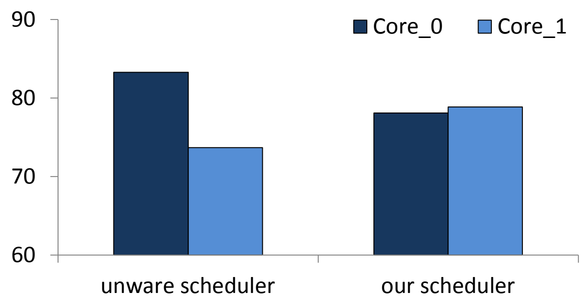
Algorithm 7.1 Application-driven temperature-aware scheduler scheme

```
Input: threads T, cores C, Spatial Gradient Threshold SGT_th;
1:  for each frame f Є V do:
2:      WP = [ ]
3:      TS = [ ]
4:      for each thread t Є T do:
5:          WM.extract()
6:          TLC.extract()
7:          TLP.extract()
8:          $wp_{ti} = f(WM,TLC,TLP)$
9:          $WP$.append($wp_{ti}$)
10:     end for
11:     for each core c Є C do:
12:         $ts_i$ = get_temperature($c_i$)
13:         $TS$.append($ts_i$)
14:     end for
15:     $current\_STG = max(TS) - min(TS)$
16:     if $current\_STG > STG_{th}$ do:
17:         WP.sort()
18:         TS.reverse_sort()
19:         schedule(T,WP,TS)        //reaction
20: end for
```

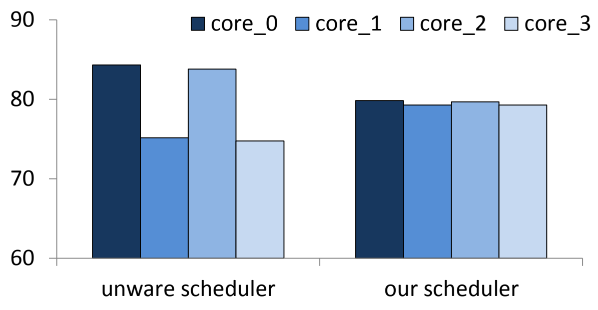
7.3 Experimental results

For evaluating our application-driven temperature-aware scheduler we use the same tool chain based setup as described in chapter 4.2. We have encoded different sequences (*PartyScene*, *RaceHorses*, *BasketballDrive* and *BQMall*) with different content characteristics for evaluating our scheduler and compare the resulted thermal profiles with an application-unaware scheduler. We consider an application-unware scheduler as the ordered mapping of threads to cores, i.e. thread t0 is mapped to core c0 and so on. We also have used the same hardware configurations with 2, 4 and 8 cores and the video frames split into 2, 4 and 8 threads. The spatial temperature gradient threshold STG_{th} was set to 5 ºC.

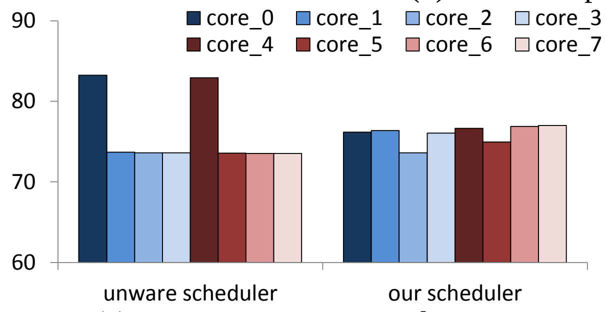
The bar graphs in figure 7.10 (a), (b) and (c) show the maximum temperature achieved for each core considering the *BasketballDrive* sequence. For all architectural configurations, the spatial temperature difference between the hottest and coldest cores is about 10 ºC when an application-unware scheduler is used. However, when using our temperature-aware scheduler the temperature difference drops to less than 1 ºC for two and four cores. Considering the scenario with eight cores, the temperature difference was reduced to 3 ºC. This spatial gradient reduction was possible since our scheduler reacts to high temperature differences between the cores by scheduling the threads with high workload to the coldest cores and the threads with low workloads to the hottest cores.



(a) Max temperature two cores



(b) Max temperature four cores



(c) Max temperature eight cores

Figure 7.10: Maximum temperature results.

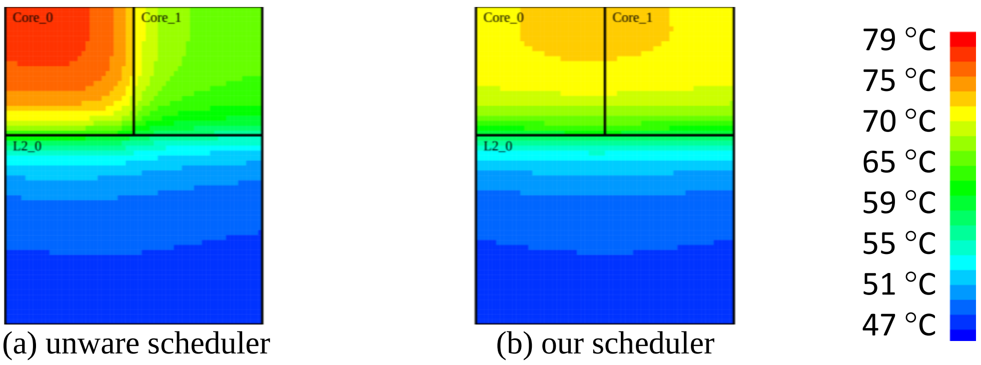
Besides the spatial temperature gradient reduction achieved when using our scheduler, the maximum overall temperature is also reduced. Additionally, the percentage of time that the cores experience high spatial temperature variations was also reduced. Table 7.1 shows the average temperature results for the four evaluated sequences in terms of maximum temperature (in ºC) of the chip and the percentage of time with spatial temperature gradients higher than 5 ºC and 10 ºC for all evaluated sequences. The spatial gradient threshold SGT_{th} was set to 5 ºC.

Table 7.1: Maximum temperature and spatial gradient results.

	unaware scheduler			**Our scheduler (STG_{th} = 5 ºC)**		
#cores	**2**	**4**	**8**	**2**	**4**	**8**
Max temp (ºC)	83.2	84.3	83.2	78.8	79.8	76.9
%time > 5 ºC	43.7	44.3	43.7	0.0	0.0	0.0
%time > 10 ºC	4.6	4.6	5.6	0.0	0.0	0.0

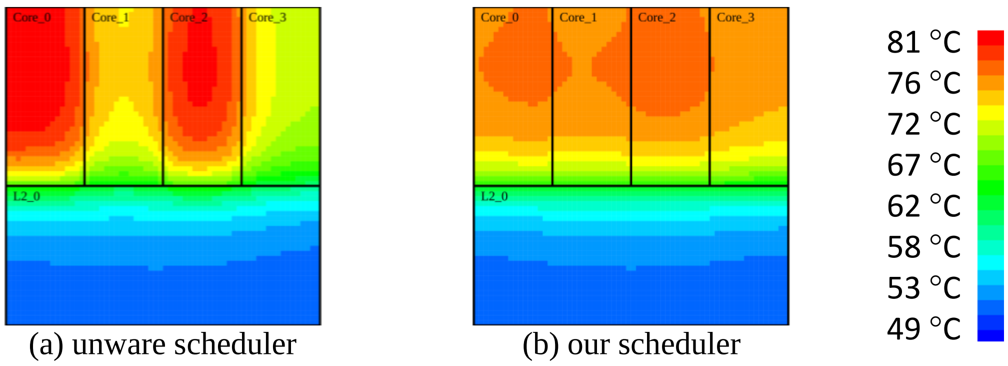
The maximum temperature achieved by the chip was reduced in about 4.5 ºC for the scenarios with two and four cores. For the experiments with eight cores the maximum temperature was reduced in more than 6 ºC. The time that the cores presented spatial temperature gradients higher than 10 ºC was reduced from about 5% for all cases to zero and from more than 40% to zero for spatial temperature gradients higher than 5 ºC. All these results are well demonstrated in figures 7.11, 7.12 and 7.13 where thermal maps are presented when using our application-driven temperature-aware scheduler (right maps) in comparison with an unaware scheduler (left maps). These maps refer to the *BasketballDrive* sequence.

All temperature distributions showed in the maps in the left side are resulted from the unbalanced nature of threads in the video coding application. As there is no scheduling considering the workload of each thread and the multi-threaded video coding application has workload unbalance nature, the temperature distribution is not uniform. Threads with high workloads are constantly assigned to the same cores which generate hotspots and large spatial gradients on the chip. On the other hand, when our application driven temperature-aware scheduler is employed the temperature distribution among the cores is much more spread. As the threads assignment to cores is guided by the spatial temperature gradient threshold STG_{th}, our technique avoids that the difference between the hottest and the coldest core exceeds the threshold. This way, not only the spatial gradients are reduced, but also the temperature distribution is much more uniform in comparison with using an unware scheduler.



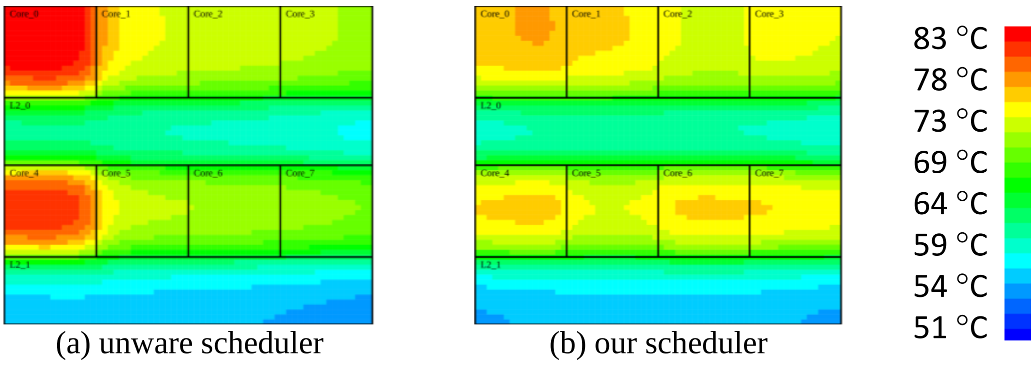
(a) unware scheduler (b) our scheduler

Figure 7.11: Thermal maps two cores for *BasketballDrive* encoding.



(a) unware scheduler (b) our scheduler

Figure 7.12: Thermal maps four cores for *BasketballDrive* encoding.



(a) unware scheduler (b) our scheduler

Figure 7.13: Thermal maps eight cores for *BasketballDrive* encoding.

8 CONCLUSIONS

For multimedia embedded systems under tight constraints, where temperature, performance, and quality cannot be compromised significantly, we introduce a set of application-driven temperature-aware solutions for video coding. The main idea of all presented solutions is to raise the concept of temperature management to the application level where application characteristics and data content properties are leveraged towards improving the temperature profiles of video coding systems with low degradation in the Quality of Service for video end users.

8.1 Summary of thesis main contributions

Application-driven dynamic thermal management for HEVC: we presented an application-level temperature management technique for HEVC. It keeps the temperature within safe operating limits while keeping the video quality degradation to as minimal as possible. Extensive analysis for different video sequences and different encoder configurations provides design hints for developing the proposed thermal management policy. We developed an application-level temperature predictor, a design-time Pareto-optimal analysis of different encoder configuration, and a run-time policy that dynamically selects an encoder configuration depending upon the predicted temperature and specified thermal threshold. Managing thermal behavior of encoders through sophisticated configuration selection enables minimizing the video quality degradation.

Thermal optimization using approximate computing: we introduced a temperature optimization technique using adaptive approximate computing for video coding systems. It employs an adaptive content-driven approximate computing technique that classifies regions of a given video sequence with respect to their resilience properties as a function of their texture and motion characteristics. Based on the resilience level, different approximation modes are employed to reduce the on-chip temperature of the HEVC encoding process. Different approximation modes are realized through both algorithm-level and data-level approximation computing mechanisms. The results illustrate the temperature reduction potential of the proposed adaptive approximate computing technique and benefit of considering the resilience knowledge to lower the output video quality losses compared to state-of-the-art works. As conclusion, approximate computing bears a significant potential for temperature optimization when employed selectively considering the variable resilience properties of the input data sets.

Application-driven thermal-aware scheduler: we designed an application-driven temperature-aware scheduler for multi-threaded video coding application. We investigated the unbalance nature of multi-threaded video coding and its impact on the temperature profiles for different multicore scenarios. We introduce a temperature-aware scheduler that uses application and data characteristics of threads encoding video sequences to reduce the thermal impact of such unbalance workloads on multicore chips. Our solution uses a content dependent workload predictor to map the high workload threads to the coldest cores according to a spatial temperature gradient threshold. As results, we demonstrate through temperature results and thermal maps that our technique is successful on reducing the spatial temperature variations and also the maximum temperature of the multicore chip.

The thesis results demonstrate that thermal management can be performed at the application level successfully. In the proposed techniques, the application adapts itself to thermal issues by using application characteristics and input data content providing thermal-safe profiles without compromising the application quality.

It is important to notice that all solutions described in this thesis were tested only in the hardware platforms described in the temperature methodology section. It is expected that other hardware platforms present different thermal profiles in comparison with those used in the proposed solutions. As the video coding application demands high computational effort, modern hardware platforms usually use a set of dedicated hardware modules for performing the most computational intensity tools of the video coding process. However, the main concept proposed in this thesis (raise the abstraction of thermal management to the application-level) can still be used to improve thermal profiles of these different hardware platforms. For instance, all application-level solutions proposed in this thesis can take as input the current temperature status of different hardware modules (if necessary) to drive the application reaction to thermal emergencies in this dedicated processing units.

As future works, we intend to employ the main concept of this thesis (raise the concept of thermal management to the application level) to other high performance applications starting with other multimedia data intensive applications as they have potential to generate undesired temperature results. We also intend to evaluate jointly hardware/software solutions, where the application can feed the hardware with information to enhance the hardware low-power decisions (DVFS, power gating, clock gating, etc…) providing safe temperature profiles with low overhead.

9 PUBLICATIONS

9.1 Publications resulted from thesis contributions

9.1.1 hevcDTM: Application-Driven Dynamic Thermal Management for High Efficiency Video Coding

Daniel Palomino, Muhammad Shafique, Altamiro Susin, Jörg Henkel.
Design, Automation and Test in Europe (DATE), 2014.

9.1.2 TONE: Adaptive Temperature Optimization for the Next Generation Video Encoders

Daniel Palomino, Muhammad Shafique, Altamiro Susin, Jörg Henkel.
International Symposium on Low Power Electronics and Design (ISLPED), 2014.

9.1.3 Thermal Optimization using Adaptive Approximate Computing for Video Coding

Daniel Palomino, Muhammad Shafique, Altamiro Susin, Jörg Henkel.
Design, Automation and Test in Europe (DATE), 2016.

9.1.4 Application-Driven Temperature-Aware Scheduling of Multi-Threaded Workloads on On-Chip Systems (*submitted*)

Daniel Palomino, Muhammad Shafique, Altamiro Susin, Jörg Henkel.
Transaction on Circuits and Systems for Video Technology (TCSVT), 2017.

9.2 Other publications during Ph.D. course

9.2.1 Energy Evaluation of the HEVC Decoding for Different Encoding Configurations

Douglas Correa, **Daniel Palomino**, Luciano Agostini, Bruno Zatt
Latin American Symposium on Circuits and Systems (LASCAS), 2017

9.2.2 Avaliação do Potencial Máximo de Speedup Usando Tiles para Compressão de Vídeo Paralela Segundo o Padrão HEVC (mention of honor)

Iago Storch, **Daniel Palomino**, Bruno Zatt, Luciano Agostini
Simpósio Brasileiro de Telecomunicações (SBRT), 2016

9.2.3 Adjusting Video Tiling to Available Resources in a Per-frame Basis in High Efficiency Video Coding

Giovani Malossi, **Daniel Palomino**, Cláudio Diniz, Sérgio Bampi, Altamiro Susin
International NEWCAS Conference (NEWCAS), 2016

9.2.4 Speedup-Aware History-Based Tiling Algorithm for the HEVC Standard

Iago Storch, **Daniel Palomino**, Bruno Zatt, Luciano Agostini
International Conference on Image Processing (ICIP), 2016

9.2.5 Fast HEVC Intra Mode Decision Algorithm Based on New Evaluation Order in the Coding Tree Block

Daniel Palomino, Eduardo Cavichioli, Muhammad Shafique, Luciano Agostini, Jörg Henkel, Altamiro Susin
International Picture Coding Symposium (PCS), 2013

9.2.6 Adaptive content-based Tile partitioning algorithm for the HEVC standard

Cauane Blummenberg, **Daniel Palomino**, Bruno Zatt, Sergio Bampi
International Picture Coding Symposium (PCS), 2013

REFERENCES

AJAMI, A.; BANERJEE, K.; PEDRAM, M. Modeling and Analisys of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)**, v. 24, n. 6, p. 849-861, June 2005.

BARTOLINI, A. et al. **A Distributed and Self-Calibrating Model-Predictve Controller for Energy and Thermal Management of High-Performance Multicores**. Design, Automation and Test in Europe (DATE). [S.l.]: [s.n.]. 2011. p. 1-6.

BARTOLINI, A. et al. Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller. **Transactions on Parallel and Distributed Systems (TPDS)**, v. 24, n. 1, p. 170-183, January 2013.

BERKTOLD, M.; TIAN, T. **CPU Monitoring with DTS/PECI**. Intel Corporation. [S.l.], p. 23. 2010.

BERNSTEIN, K. et al. High-performance CMOS variability in the 65-nm regime and beyond. **IBM Journal of Research and Development**, v. 50, n. 4.5, p. 433 - 449, July 2006.

BINKERT, N. et al. The M5 Simulator: Modeling Networked Systems. **Journal IEEE MICRO**, v. 26, n. 4, p. 52-60, July 2006.

BINKERT, N. et al. The gem5 Simulator. **ACM SIGARCH Computer Architecture News**, New York, v. 39, n. 2, p. 1-7, May 2011.

BJONTEGARD, G. **Calculation of average PSNR differences between RD-curves**. ITU - Telecommunications Standardization Sector. Austin, p. 1-4. 2001.

BLUMENBERG, C. et al. **Adaptive Content-Based Tile Partitioning Algorithm for the HEVC Standard**. Picture Coding Symposium (PCS). [S.l.]: [s.n.]. 2013. p. 185-188.

BOHR, M. A 30 Year Retrospective on Dennard's MOSFET Scaling Paper. **Solid-state circuits society newsletter**, v. 12, n. 1, p. 11-13, 2007.

BORKAR, S. et al. **Parameter Variations and Impact on Circuits and Microarchitecture**. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2003. p. 338-342.

BOSSEN, F. **Common test conditions and software reference configurations**. ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC). [S.l.]. 2012.

CES-KIT, C. F. E. S., 2013. Disponivel em: <http://ces.itec.edu.br>. Acesso em: 2013.

CHAKRADHAR, S.; RAGHUNATHAN, A. **Best-effort computing:** Re-thinking parallel software and hardware. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2010. p. 865-870.

CHIP-ARCHITECT. Chip-Architect, 2010. Disponivel em: <http://www.chip-architect.com/news/2010_09_04_AMDs_Bobcat_versus_Intels_Atom.html>. Acesso em: 2015.

CHIPPA, V. et al. **StoRM:** A Stochastic Recognition and Mining processor. International Symposium on Low Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2014. p. 39-44.

CHIPPA, V.; MOHAPATRA, D.; ROY, K. Scalable Effort Hardware Design. **Transactions on Very Large Scale Integration (TVLSI)**, v. 22, n. 9, p. 2004-2016, 2014.

CHO, M. et al. Power Multiplexing for Thermal Field Management in Many-Core Processors. **Transactions on Components, Packaging and Manufacturing Technology (TCPMT)**, v. 3, n. 1, p. 94-104, January 2013.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology, 2012 - 2017**. [S.l.], p. 29. 2013.

CORREA, G. et al. Fast HEVC Encoding Decisions Using Data Mining. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 25, n. 4, p. 660-673, 2014.

COSKUN, A. et al. Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs. **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)**, v. 16, n. 9, p. 1127-1140, September 2008.

COSKUN, A.; ROSING, T.; GROSS, K. **Proactive Temperature Balancing for Low Cost Thermal Management in MPSoCs**. International Conference on Computer-Aided Design (ICCAD). [S.l.]: [s.n.]. 2008. p. 250 - 257.

COSKUN, A.; ROSING, T.; GROSS, K. Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs. **Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)**, v. 28, n. 10, p. 1503-1514, October 2009.

DAS, I. On characterizing the “knee” of the Pareto curve based on Normal-Boundary Intersection. **Structural optimization**, v. 18, n. 2-3, p. 107-115, 1999.

DENNARD, R. et al. Design of ion-implanted MOSFET's with very small physical dimensions. **Journal of solid-state circuits**, v. 9, n. 5, p. 256-268, 1974.

DIAS, I. C. **PYROVIEW 380L compact**, 2013. Disponivel em: <http://www.dias-infrared.com/pdf/pyroview380l_eng.pdf>. Acesso em: 15 Julho 2013.

EBI, T. et al. **Economic Learning for Thermal-Aware Power Budgeting in Many-core Architectures**. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS). [S.l.]: [s.n.]. 2011. p. 189-196.

EBI, T.; FARUQUE, M.; HENKEL, J. **TAPE:** Thermal-Aware Agent-Based Power Economy for Multi/Many-Core Architectures. International Conference on Computer-Aided Design (ICCAD). [S.l.]: [s.n.]. 2009. p. 302-309.

FISHER, N. et al. Thermal-aware global real-time scheduling and analysis on multicore systems. **Journal of Systems Architectures**, v. 57, n. 5, p. 547/560, Maio 2011.

FORTE, D.; SRIVASTAVA, A. **Energy- and Thermal-Aware Video Coding via Encoder/Decoder Workload Balancing**. International Symposium on LoW-Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2010. p. 207-212.

GNAD, D. et al. **Hayat:** Harnessing Dark Silicon and variability for aging deceleration and balancing. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2015. p. 1-6.

GUPTA, V. et al. **IMPACT:** IMPrecise adders for low-power approximate computing. International Symposium on Low Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2011. p. 409-414.

HENKEL, J. et al. **Reliable On-Chip Systems in the Nano-Era:** Lessons Learnt and Future Trends. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2013.

HENKEL, J. et al. **Thermal Management for Dependable On-Chip Systems**. Asia and South Pacific Design Automation Conference (ASP-DAC). [S.l.]: [s.n.]. 2013. p. 113-118.

HUNG, W. et al. **Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture**. International Conference on Computer Design. [S.l.]: [s.n.]. 2004. p. 430-437.

ITU-T. **SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS - Infrastructure of Audiovisual Services - Coding of Moving Video - High Efficiency Video Cdoing**. [S.l.]. 2013.

JCT-VC. HM 11.0 Reference Software, 2013. Disponivel em: <http://hevc.hhi.franhofer.de/>.

JCT-VC. HM 16.0 Reference Software, 2015. Disponivel em: <http://hevc.hhi.franhofer.de/>.

JEDEC, S. S. T. A. **Failure Mechanisms and Models for Semiconductor Devices**. JEDEC publication JEP122C. [S.l.]. 2006.

KUMAR, A. et al. System-Level Dynamic Thermal Management for High-Performance Microprocessors. **Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)**, v. 27, n. 1, p. 96-108, Janeiro 2008.

LEE, W.; PATEL, K.; PEDRAM, M. **Dynamic Thermal Management for MPEG-2 Decoding**. International Symposium on Low Power Electronics and Devices (ISLPED). [S.l.]: [s.n.]. 2006. p. 316-321.

LEE, W.; PATEL, K.; PEDRAM, M. GOP-Level Dynamic Thermal Management in MPEG-2 Decoding. **Transactions on Very Large Scale Integration (TVLSI)**, v. 16, n. 6, p. 662-672, June 2008.

LI, S. et al. **McPAT:** An Integrated Power, Area, and Timing Modeling Gramework for Multicore and Manycore Architectures. International Symposium on Microarchitecture (MICRO). [S.l.]: [s.n.]. 2009. p. 469-480.

LIAN, C. et al. **Development of a Flexible Chip Infrared (IR) Thermal Imaging System for Product Qualification**. Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM). [S.l.]: [s.n.]. 2012. p. 337-343.

LINK, G. M.; VIJAYKRISHNAN, N. **Thermal Trends in Emerging Technologies**. International Symposium on Quality Electronic Design (ISQED). [S.l.]: [s.n.]. 2006. p. 625-632.

MARCU, M.; MILOS, C.; TUDOR, D. **Power-Thermal Analysis of Multimedia Applications**. International Workshop on Thermal Investigations of ICs and Systems (THERMINIC). [S.l.]: [s.n.]. 2010. p. 1-6.

MARTIN, M. et al. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. **ACM SIGARCH Computer Architecture News**, v. 33, n. 4, p. 92-99, November 2005.

MESA-MARTINEZ, F. et al. **Measuring Power and Temperature From Real Processors**. International Symposium on Parallel and Distributed Processing (ISPDP). [S.l.]: [s.n.]. 2008. p. 1-5.

MIRTAR, A.; DEY, S.; RAGHUNATHAN, A. **Adaptation of Video Encoding to Address Dynamic Thermal Management Effects**. International Green Computing Conference (IGCC). [S.l.]: [s.n.]. 2012. p. 1-10.

MOORE, G. E. Cramming more components onto integrated circuits. **IEEE Solid-State Circuits Society Newsletter**, v. 20, n. 3, p. 33 - 35, 2009.

OHM, J.-R. et al. Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1669-1684 , October 2012.

PALOMINO, D. et al. **hevcDTM:** Application-Driven Dynamic Thermal Management for High Efficiency Video Coding. Design, Automation and Test in Europe (DATE). Dresden: [s.n.]. 2014. p. 1-4.

PALOMINO, D. et al. **TONE:** Adaptive temperature optimization for the next generation video encoders. International Symposium on Low Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2014. p. 33-38.

PALOMINO, D. et al. **Thermal Optimization using Adaptive Approximate Computing for Video Coding**. Design, Automation and Test in Europe (DATE). [S.l.]: [s.n.]. 2016. p. 1207-1212.

PECHT, M.; LALL, P.; HAKIM, E. B. The influence of temperature on integrated circuit failure mechanisms. **Quality and Reliability Engeneering International**, v. 8, n. 3, p. 167-176, 1992.

RAMASUBRAMANIAN, S. et al. **Relax-and-Retime:** A Methodology for Energy-Efficient Recovery Based Design. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2013. p. 1-6.

REDA, S. Thermal and Power Characterization of Real Computing Devices. **Jornal on Emerging and Selected Topics in Circuits and Systems (JETCAS)**, v. 1, n. 2, p. 76-87, June 2011.

RICHARDSON, Y. **H.264/AVC and MPEG-4 Video Compression - video Cdoing for Next-Generation Multimedia**. [S.l.]: [s.n.], 2003.

SHAFIQUE, M. et al. **Adaptive power management of on-chip video memory for multiview video coding**. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2012. p. 866-875.

SHAFIQUE, M.; ZATT, B.; HENKEL, J. **A complexity reduction scheme with adaptive search direction and mode elimination for multiview video coding**. Picture Coding Symposium (PCS). [S.l.]: [s.n.]. 2012. p. 105-108.

SIDIROGLOU-DUOSKUS, S. et al. **Managing performance vs. accuracy trade-offs with loop perforation**. SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE). [S.l.]: [s.n.]. 2011. p. 124-134.

SKADRON, K. et al. Temperature-Aware Microarchitecture: Modeling and Implementation. **Transactions on Architecture and Code Optimization (TACO)**, New York, v. 1, n. 1, p. 94-125, March 2003.

SRINIVASAN, J.; ADVE, S. **Predictive Dynamic Thermal Management for Multimedia Applications**. International Conference on Supercomputing (ICS). [S.l.]: [s.n.]. 2003. p. 109-120.

STALLINGS, W. **Arquitetura e Organização de Computadores**. 8^{a}. ed. [S.l.]: Pearson, 2010.

SULLIVAN, G. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1649-1668, October 2012.

TIWARI, A.; TORRELLAS, J. **Facelift:** Hidding and Slowing Down Aging in Multicores. MICRO 41. [S.l.]: [s.n.]. 2008. p. 129-140.

VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1885-1898, October 2012.

VENKATARAMANI, S. et al. **AxNN:** energy-efficient neuromorphic systems using approximate computing. International Symposium on Low Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2014. p. 27-32.

VENKATARAMANI, S. et al. **Computing Approximately, and Efficiently**. Design, Automation and Test in Europe (DATE). [S.l.]: [s.n.]. 2015. p. 48-751.

WEBM. VP9 Video Codec Summary. **VP9 Video Codec**, 2013. Disponivel em: <http://www.webmproject.org/vp9/>. Acesso em: February 2014.

YEO, I.; KIM, E. **Hybrid Dynamic Thermal Management Based on Statistical Characteristics of Multimedia Applications**. International Symposium on Low Power Electronics and Design (ISLPED). [S.l.]: [s.n.]. 2008. p. 321-326.

YEO, I.; LIU, C.; KIM, E. **Predictive Dynamic Thermal Management for Multicore Systems**. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2008. p. 734 - 739.

ZANINI, F. et al. **Multicore Thermal Management with Model Predictive Control**. European Conference on Circuit Theory and Design (ECCTD). [S.l.]: [s.n.]. 2009. p. 711-714.

ZHANG, K. et al. **Minimizing Thermal Variation Across System Components**. International Parallel and Distributed Processing Symposium (IPDPS). [S.l.]: [s.n.]. 2015. p. 1139-1148.

ZHENG, J. et al. **Overview of AVS Broadcasting Standard for High Definition Video**. China Summit & International Conference on Signal and Information Processing (ChinaSIP). [S.l.]: [s.n.]. 2013. p. 250-254.

APPENDIX A – RESUMO – PORTUGUÊS

Soluções para o gerenciamento de temperatura de sistemas de codificação de vídeo

A1. Resumo

Esta tese apresenta soluções para o gerenciamento e otimização de temperatura para sistemas de codificação de vídeo baseados nas características da aplicação e no conteúdo dos vídeos digitais. Diferente dos trabalhos estado-da-arte, as soluções propostas nesta tese focam em técnicas de gerenciamento de temperatura no nível da aplicação e características da aplicação codificação de vídeo e as propriedades dos vídeos digitais são explorados para desenvolver soluções termais para a codificação de vídeo com baixas perdas na qualidade de serviço das aplicações. Diversas análises são realizadas considerando a aplicação de codificação de vídeo para entender o comportamento da temperatura durante o processo de codificação para diferentes sequências de vídeo. Com base nos resultados das análises, soluções com diferentes abordagens são propostas para atenuar os efeitos da temperatura nos sistemas de codificação de vídeo. Gerenciamento de temperatura baseado nas características da aplicação para o padrão de codificação HEVC usa uma técnica de seleção de configuração em tempo de execução para manter a temperatura abaixo dos limites seguros de operação com bons resultados de qualidade de vídeo. Otimização de temperatura baseado em computação imprecisa usa aproximações baseadas em conteúdo para reduzir a temperatura de chips executando o HEVC. Um escalonador de tarefas que usa características da aplicação para guiar o escalonamento de tarefas focando na redução dos gradientes espaciais de temperatura que são resultantes do desbalanceeamento natural de cargas entre as tarefas da aplicação. As soluções propostas são capazes de reduzir em até 10 ºC a temperatura do chip com perdas insignificantes na eficiência de compressão. Os resultados de qualidade objetiva (medida usando PSNR) são de 12 dBs até 20 dBs maiores quando comparados com trabalhos da literatura. Além disso, o escalonador de tarefas proposto é capaz de eliminar os gradientes espaciais de temperatura maiores que 5 ºC para arquitetura multi-cores. Como principal conclusão, esta tese demonstra que as técnicas de gerenciamento de temperatura que usam o conhecimento da aplicação de maneira conjunta com as propriedades dos vídeos digitais tem um alto potencial para melhorar os resultados de temperatura de sistemas de codificação de vídeo mantendo bons resultados de qualidade visual dos vídeos codificados.

A2. Introdução

Serviços baseados em vídeo estão cada vez mais populares nos mercados de consumo e de comunicação. Dispositivos como TVs, *smartphones*, *tablets* são capazes de reproduzir e capturar vídeos digitais. Vídeos digitais utilizam muitos dados para serem representados e por isso tecnologias de compressão de vídeos são essenciais para o sucesso de aplicações que lidam com este tipo de mídia, principalmente com a crescente popularização dos vídeos de ultra alta resolução e também com as tecnologias de imersão como o 3D. Padrões de codificação de vídeo como o HEVC (ITU-T, 2013) são desenvolvidos para lidar com essa grande quantidade de dados, entretanto, para serem capazes de comprimir de forma adequada vídeos de ultra alta resolução os algoritmos presentes nos codificadores de vídeos atuais possuem grande complexidade computacional (VANNE, VIITANEN, *et al.*, 2012).

A alta complexidade dos novos codificadores de vídeo é complementada pelos sistemas de alta capacidade de processamento disponíveis hoje em dias. Esses sistemas se tornaram computacionalmente poderosos devido a duas razões principais: (1) a miniaturização dos transistores que permitiu um aumento da densidade desses dispositivos por área e (2) os avanços nas pesquisas em micro arquitetura que propiciaram o aumento da capacidade de extração de paralelismo a nível de instrução. Entretanto, a miniaturização dos transistores não foi seguida por uma redução na tensão de alimentação desses dispositivos na mesma proporção (DENNARD, GAENSSLEN, *et al.*, 1974), e consequentemente a densidade de potência por área aumentou resultando em altos níveis de temperatura nos circuitos digitais.

Elevadas temperaturas no chip aumentam os custos com resfriamento, que é mais desafiante em sistemas embarcados onde as restrições de área e energia são mais intensas. Além disso, altas temperaturas afetam negativamente os efeitos de *aging* nos circuitos integrados (HENKEL, BAUER, *et al.*, 2013) e gradientes espaciais e temporais reduzem a confiabilidade e o desempenho dos circuitos digitais (PECHT, LALL e HAKIM, 1992).

Existem vários trabalhos na literatura que lidam que lidam com os efeitos da temperatura nos circuitos digitais. Tradicionalmente, a técnica de gerenciamento térmico dinâmico é discutida na maioria dos trabalhos da literatura, onde o principal objetivo é manter a temperatura dentro de níveis adequados com menos penalidades possíveis nas restrições de tempo real e de desempenho das aplicações. Além disso, existem alguns trabalhos que focam em gerenciamento térmico para codificação/decodificação de vídeos.

O objetivo desta tese é atenuar os efeitos de temperatura em sistemas de codificação de vídeo atuando no nível da aplicação. A abordagem desta tese difere dos trabalhos da literatura, pois o conceito de gerenciamento térmico é trazido do nível do *hardware* para ser tratado pelo nível da aplicação, onde a temperatura é gerenciada com base nas características da aplicação e no conteúdo dos vídeos digitais. Como resultado, as técnicas propostas nessa tese demonstram que é possível gerenciar a temperatura no nível da aplicação com baixa perda de desempenho no resultado final da aplicação.

A3. Soluções de gerenciamento térmico para a codificação de vídeo

Nesta tese, três metodologias diferentes foram utilizadas para adquirir os resultados de temperatura.

A primeira metodologia utiliza uma câmera infravermelha que filma um processador rodando a aplicação de codificação de vídeo. A segunda metodologia é baseada em três simuladores (BINKERT, BECKMAN, *et al.*, 2011) (LI, STRONG, *et al.*, 2009) (SKADRON, STAN, *et al.*, 2003), que executam a aplicação de codificação de vídeo com vídeos reais. A terceira metodologia utiliza um sensor de temperatura que está presente nos processadores mais modernos.

Para todas as metodologias, sequências de vídeo são codificadas utilizando o software HEVC *test Model* (HM) (JCT-VC, 2015)seguindo as recomendações da comunidade de codificação de vídeo (BOSSEN, 2012).

Para o desenvolvimento de todas as soluções de gerenciamento/otimização de temperatura propostas nessa tese, várias análises de temperatura foram realizadas. Essas análises tornaram possível o entendimento do efeito das características da aplicação de codificação de vídeo e o conteúdo dos vídeos digitais na temperatura final dos sistemas de codificação de vídeo

A3.1 Solução de gerenciamento térmico dinâmico utilizando parâmetros de configuração

A primeira solução de gerenciamento térmico em tempo real para a codificação de vídeo apresentada neste trabalho utiliza a técnica de frentes de Pareto (DAS, 1999) para escolher o melhor conjunto de parâmetros de configuração para a aplicação buscando manter a temperatura dentro de um limiar previamente definido. O algoritmo A1 mostra as principais etapas dessa solução.

O algoritmo utiliza a temperatura atual e mais um conjunto de características de complexidade dos quadros do vídeo que serão codificados para prever a temperatura futura após a codificação desses quadros. Se a temperatura tende a extrapolar o limiar previamente definido o algoritmo faz uma escolha de novos parâmetros de configuração para a aplicação de codificação de vídeo objetivando a redução da temperatura com a menor perda de qualidade possível. Esse processe se repete para cada quadro do vídeo a ser codificado.

Algoritmo A1 Video quality-aware Temperature Management

```
Input: Pontos de pareto P, vídeo V, Limiar de temperatura $T_{th}$;
1:  error_list = [ ];
2:  c = initial configuration;
3:  $T_{current}$ = measure_temperature();
4:  for each frame $f \in$ V do:
5:      $C_{next}$ = classify_complexity($f$);
6:      $\Delta T = T_v(C_{next}, C_{previous})$ + mean(error_list);
7:      $T_p = T_{current} + \Delta T$;
8:      if $T_p > T_{th}$ do:
9:          c = pareto_selection(P, $T_{th}$); //reaction
10:     end if;
11:     encode($f$, c);
12:     $T_{current}$ = measure_temperature();
13:     error = $T_{current} - T_p$;
14:     update_error_list(error);
15:     $C_{previsou} = C_{next}$;
16: end for;
```

Como resultados, o algoritmo foi capaz de manter a temperatura dentro dos níveis pré-estabelecidos sem comprometer de forma intensa a qualidade final do vídeo codificado. Além disso, os resultados obtidos por essa solução foram melhores quando comparados com soluções da literatura.

A3.2 Solução de otimização de temperatura utilizando computação aproximada

A segunda solução proposta nesta tese utiliza o conceito de computação aproximada para otimizar/reduzir a temperatura de sistemas de codificação de vídeo. A computação aproximada é uma técnica que introduz erros na computação a ser realizada em troca de algum benefício colateral (em geral redução de energia). Neste trabalho, a ideia da computação aproximada é utilizada de maneira adaptativa de acordo com a resiliência a erros das regiões dos vídeos que e serão executados objetivando a redução dos níveis de temperatura em circuitos que executam a codificação de vídeo. As características dos vídeos a serem codificados são utilizadas para definir os níveis de resiliência de cada região do vídeo.

Em seguida, modos de aproximação (MAs) com diferentes intensidades são aplicados de maneira adaptativa na codificação do vídeo. Os modos de aproximação utilizam aproximação de cálculos em nível algorítmico (alguns algoritmos são podados e não são realizados de forma completa) e em nível de dados (alguns dados são ignorados no momento da computação).

O algoritmo A2 resume como essa técnica é utilizada neste trabalho. Cada quadro do vídeo é dividido em regiões. Para cada região é extraído o nível de resiliência a erros dessa região, ou seja, o quanto essa região é sensível à introdução de erros (quanto mais sensível for a região, maior vai ser o impacto negativo de se introduzir erros no resultado de qualidade final da codificação). Após classificar cada região do vídeo de acordo com o seu nível de resiliência, a técnica aplica diferentes níveis de aproximação (utilizando modos de aproximação). Em regiões classificadas como resiliente, modos de aproximação mais intensos são aplicados, ou seja, modos que introduzem mais erros no processo de codificação. Em regiões mais sensíveis, modos de aproximação menos intensos são utilizados. Dessa maneira, é possível reduzir a demanda por computação, e consequentemente a temperatura do sistema de codificação, sem comprometer de maneira intensa a qualidade final do vídeo codificado.

Algoritmo A2 Approximate mode selection heuristic.

```
Input: sequence S;
1:  v_list = [ ];
2:  for each frame f Є S do:
3:      for each CTU cu Є f do:
4:          $v_{CTU}$ = extract_variance(ctu);
5:          update_v_list($v_{CTU}$);
6:      end for;
7:      for each CTU ctu Є f do:
8:          norm_$v_{CTU}$ = [$v_{CTU}$ – min(v_list)] / [max(v_list) – min(v_list)];
9:          case norm_$v_{CTU}$:
10:             resilient: encode(ctu, AM-3);
11:             medium resilient: encode(ctu, AM-2);
12:             medium sensitive: encode(ctu, AM-1);
13:             sensitive: encode(ctu, AM-0);
14:         end case;
15:     end for;
16: end for;
```

Como resultado, a solução de gerenciamento térmico que utiliza computação aproximada foi capaz de reduzir em média 10 °C a temperatura de sistemas de codificação de vídeo em troca de uma baixa perda de qualidade no vídeo codificado. Além disso, quando comparada com trabalhos da literatura, a técnica proposta neste trabalho foi similar ou superior.

A3.3 Esquema de escalonador de tarefas baseado na aplicação

A terceira solução proposta nesta tese foca em reduzir os gradientes espaciais de temperatura em arquitetura multi processadas processando a codificação de vídeo de forma paralela. A intensidade computacional da aplicação codificação de vídeo é principalmente influenciada pelo conteúdo dos vídeos digitais. Isso significa que quando um vídeo digital é igualmente dividido entre unidades de processamento para realizar a codificação de vídeo paralela existe um desbalanceamento natural de carga computacional entre as diferentes unidades de processamento, o que resulta em gradientes de temperatura. Deste modo, esta técnica utiliza m esquema de escalonamento de tarefas para reduzir os gradientes espaciais de temperatura provocados pelo desbalanceamento natural de carga computacional na codificação de vídeo paralela.

O algoritmo A3 resume as principais etapas do esquema de escalonamento proposto. A primeira etapa consiste em um monitoramento de variáveis que são utilizadas pelo esquema de escalonamento. Entre as variáveis estão: os níveis de temperatura de cada unidade de processamento, as características dos quadros sendo processados e a distribuição de carga computacional de quadros anteriormente codificados. Todas essas variáveis são utilizadas para definir o escalonamento das diferentes tarefas geradas pela aplicação. O algoritmo refaz o escalonamento das tarefas toda vez que um limiar de gradiente de temperatura é atingido. Dessa forma, o algoritmo utiliza os resultados de uma etapa de predição de carga computacional para definir qual tarefa será assinalada a cada unidade de processamento. Às unidades de processamento que possuem níveis altos de temperatura são assinaladas as tarefas menos custosas em termos de carga computacional, enquanto que as unidades de processamento com baixos níveis de temperatura são assinaladas com as tarefas mais custosas em termos de carga computacional.

Algoritmo A3 Application-driven temperature-aware scheduler scheme

```
Input: threads T, cores C, Spatial Gradient Threshold $SGT_{th}$;
1:  for each frame f Є V do:
2:      WP = [ ]
3:      TS = [ ]
4:      for each thread t Є T do:
5:          WM.extract()
6:          TLC.extract()
7:          TLP.extract()
8:          $wp_{ti} = f(WM,TLC,TLP)$
9:          WP.append($wp_{ti}$)
10:     end for
11:     for each core c Є C do:
12:         $ts_i$ = get_temperature($c_i$)
13:         TS.append($ts_i$)
14:     end for
15:     current_STG = max(TS) – min(TS)
16:     if current_STG > $STG_{th}$ do:
17:         WP.sort()
18:         TS.reverse_sort()
19:         schedule(T,WP,TS)        //reaction
20: end for
```

Como principais resultados, a técnica de escalonamento de tarefas foi capaz de reduzir a zero o tempo em que gradientes de temperatura eram superiores a 5 °C e 10 °C em sistema de codificação de vídeo multiprocessados. Além disso, foi possível atingir níveis de temperatura menores para o sistema inteiro quando comparado com a não utilização da técnica proposta.

A4. Conclusões

Esta tese introduz um conjunto de soluções de gerenciamento de temperatura para codificação de vídeo aptas a garantir qualidade quando requisitos de temperatura e desempenho são rígidos. A principal ideia utilizada para o desenvolvimento das soluções propostas é trazer o conceito de gerenciamento térmico do nível do hardware para o nível da aplicação, de maneira que a aplicação responde pró ativamente às violações de temperatura que venham a ocorrer em sistemas de codificação de vídeo. As características da aplicação e o conteúdo dos vídeos digitais são utilizados para melhorar os perfis de temperatura com baixa degradação na qualidade dos vídeos. Os resultados demonstraram que o gerenciamento térmico pode ser realizado no nível da aplicação com sucesso.

Como trabalhos futuros, nós pretendemos empregar o principal conceito desta tese em outras aplicações de alto desempenho, iniciando com outras aplicações multimídias que tenham potencial de gerar resultados de temperatura indesejados. Também é planejado avaliar soluções que atuem tanto no nível do software quanto do hardware, onde a aplicação pode alimentar o hardware com informação para enriquecer as decisões para reduzir a temperatura.

A5. Referências

BINKERT, N. et al. The gem5 Simulator. **ACM SIGARCH Computer Architecture News**, New York, v. 39, n. 2, p. 1-7, May 2011.

BOSSEN, F. **Common test conditions and software reference configurations**. ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC). [S.l.]. 2012.

DAS, I. On characterizing the “knee” of the Pareto curve based on Normal-Boundary Intersection. **Structural optimization**, v. 18, n. 2-3, p. 107-115, 1999.

DENNARD, R. et al. Design of ion-implanted MOSFET's with very small physical dimensions. **Journal of solid-state circuits**, v. 9, n. 5, p. 256-268, 1974.

HENKEL, J. et al. **Reliable On-Chip Systems in the Nano-Era:** Lessons Learnt and Future Trends. Design Automation Conference (DAC). [S.l.]: [s.n.]. 2013.

ITU-T. **SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS - Infrastructure of Audiovisual Services - Coding of Moving Video - High Efficiency Video Cdoing**. [S.l.]. 2013.

JCT-VC. HM 16.0 Reference Software, 2015. Disponivel em: <http://hevc.hhi.franhofer.de/>.

LI, S. et al. **McPAT:** An Integrated Power, Area, and Timing Modeling Gramework for Multicore and Manycore Architectures. International Symposium on Microarchitecture (MICRO). [S.l.]: [s.n.]. 2009. p. 469-480.

PECHT, M.; LALL, P.; HAKIM, E. B. The influence of temperature on integrated circuit failure mechanisms. **Quality and Reliability Engeneering International**, v. 8, n. 3, p. 167-176, 1992.

SKADRON, K. et al. Temperature-Aware Microarchitecture: Modeling and Implementation. **Transactions on Architecture and Code Optimization (TACO)**, New York, v. 1, n. 1, p. 94-125, March 2003.

VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1885-1898, October 2012.