Alma Mater Studiorum - Università degli Studi di Bologna

DEIS - DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA

CORSO DI DOTTORATO IN INGEGNERIA ELETTRONICA, INFORMATICA E DELLE TELECOMUNICAZIONI

CICLO XXIII - settore scientifico-disciplinare ING-INF01 Elettronica

DYNAMIC POWER MANAGEMENT: FROM PORTABLE DEVICES TO HIGH PERFORMANCE COMPUTING

Candidato:

ANDREA BARTOLINI

Relatore:

Chiar. mo Prof. Ing. **LUCA BENINI**

Co-Relatore:

Chiar. ma Prof. ssa Ing. **MICHELA MILANO**

Coordinatore:

Chiar. ma Prof. ssa Ing. **PAOLA MELLO**

Esame Finale Anno 2011

Dynamic power management: from portable devices to high performance computing

Andrea Bartolini

Department of Electronic, Computer Science and Systems

University of Bologna

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

March 2011

Acknowledgements

There are so many people I would like to acknowledge for this PhD work that probably I will forgot some. All the nice person that I met during these three years...amazing! Of course in between them there are ones that shine. Prof. Luca Benini for the advise, the opportunity and for keeping us constantly in the path. All the people of micrellab: Martino, Andrea, Paolo, Francesco, Angela, Igor, Mohammad, Giacomo, Davide, Betta, Carlotta, Christine and Marco for the daily help and the fun. The Control System Group, Andrea Tilli and Matteo. The Artificial Intelligence group Prof.ssa Michela Milano and Michele. It has been nice working with you. All the students that help me in my research with projects and thesis. Freescale Semiconductor LTD for supporting my PhD in the first years, in particolar Nigel Drew, Jafar Farougy, Yolanda Prieto. Intel Corp. and Intel Labs Braunschweig for supporting my PhD during the final years. In particular Matthias Gries, Tor Lund-Larsen and all the nice people working in the group. A particular thanks goes to my parents and my brother Fabio. Another special one goes to all my friends for keep me smiling along these years.

To all of you...Grazie!

Abstract

Electronic applications are nowadays converging under the umbrella of the cloud computing vision. The future ecosystem of information and communication technology is going to integrate clouds of portable clients and embedded devices exchanging information, through the internet layer, with processing clusters of servers, data-centers and high performance computing systems. Even thus the whole society is waiting to embrace this revolution, there is a backside of the story. Portable devices require battery to work far from the power plugs and their storage capacity does not scale as the increasing power requirement does. At the other end processing clusters, such as data-centers and server farms, are build upon the integration of thousands multiprocessors. For each of them during the last decade the technology scaling has produced a dramatic increase in power density with significant spatial and temporal variability. This leads to power and temperature hot-spots, which may cause non-uniform ageing and accelerated chip failure. Nonetheless all the heat removed from the silicon translates in high cooling costs. Moreover trend in ICT carbon footprint shows that run-time power consumption of the all spectrum of devices accounts for a significant slice of entire world carbon emissions. This thesis work embrace the full ICT ecosystem and dynamic power consumption concerns by describing a set of new and promising system levels resource management techniques to reduce the power consumption and related issues for two corner cases: Mobile Devices and High Performance Computing.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Historically separated markets such as embedded systems, high performance computing (HPC) and Data Center, are today together the major building blocks of the up-coming cloud computing vision. Indeed as we can see in Fig. 1.1 today Information and Communication Technology (ICT) market is composed at one end by a set of thin mobile clients (ex. Laptop, Smarthphone, ebook,..) handled by the final users and at the other end by clusters of data and service centers. This two end points are interfaced through the internet communication layer.

Why power matter? Power consumption of the entire ICT ecosystem has grown importance during the last decade and it has been shown to contribute significantly at the world wide carbon footprint (1, 2). Moving from the 2% of 2007 toward the projected 10% in 2020. Indeed PC ownership will quadruple between 2007 and 2020 to 4 billion devices, and emissions will double over the same period, with laptops overtaking desktops as the main source of global ICT emissions (22%). Mobile phone ownership will almost double to nearly 5 billion accounts by 2020, but emissions will only grow by 4%. Data centers trends suggests the world will be using 122 million servers in 2020, up from 18 million today. In 2002, the global data center footprint, including equipment use and embodied carbon, was 76 MtCO2e and this is expected to more than triple by 2020 to 259 MtCO2e making it the fastest-growing contributor to the ICT sectors carbon footprint.

Power consumption does not effects only the sustainability of our future world, but it is also a very sensitive key market player both in high performance computing and embedded industry.

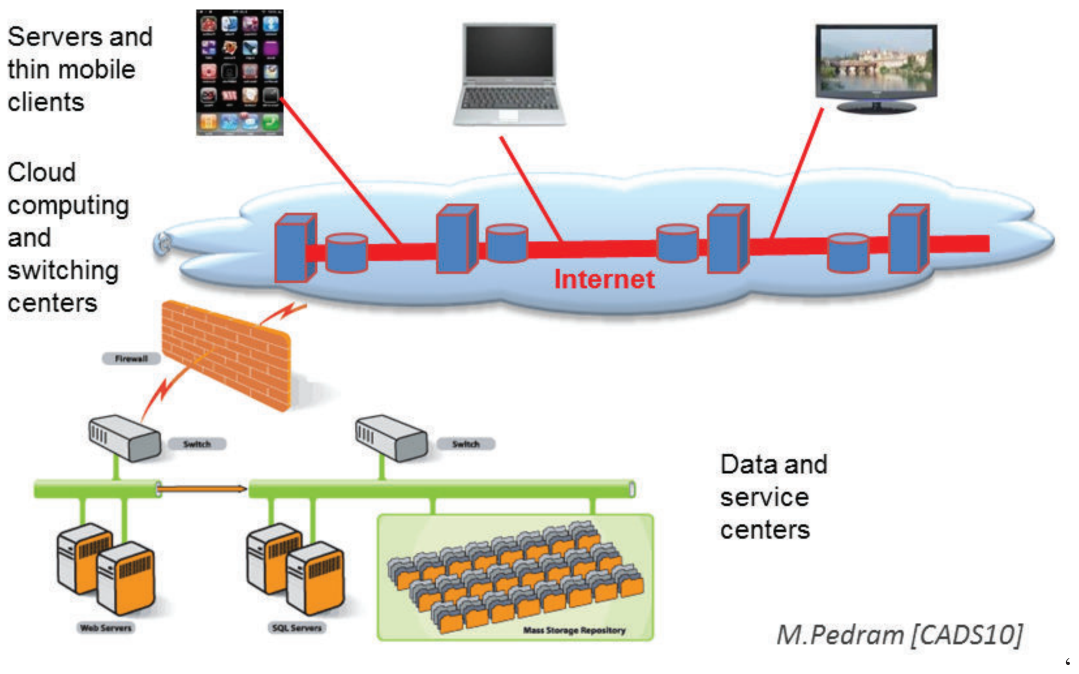


Figure 1.1: ICT ecosystem

Indeed as customers every one of us knows the importance of longer battery life time in our portable electronic devices. Whereas regarding high performance computing, in 2006 the US Environmental Protection Agency reported that the US used 61 billion kilowatt-hours of power for data centers and servers in 2006. This accounted as the 1.5 percent of all US electricity use, and it costs the companies that paid those bills more than $4.5 billion(3). Moreover 50% of the energy consumed by data centers is used for powering the cooling infrastructure. The remaining energy is used for computation and causes the temperature ramp-up. Thus power consumption is not only a concern for today ICT systems but a key issue to tackle for future society improvements.

By keeping the focus on the two corner cases of the ICT ecosystem, mobile clients and high performance computing, in Fig. 1.2 are reported the portion of the carbon footprint for different mobile clients. It can be noticed that the usage power accounts for half of the entire life cycle emissions. Whereas in Fig. 1.3 is reported the world-wide servers market cost trend. It can be noticed that whereas money spent yearly in new server is stable the cost for maintenance (power and cooling) is increasing along the years and now is comparable with the investment in new hardware.

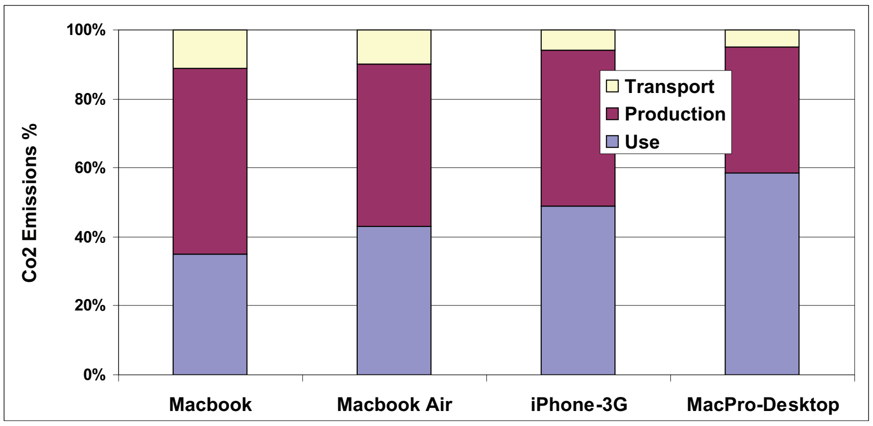


Figure 1.2: Mobile clients life cycle carbon emission

Thus we can clearly estimate that the run-time power consumption accounts roughly for the half of the costs and environment impacts of the entire ICT market.

Higher power effort does not always translate in higher system performance and final quality of service (QoS). Indeed the limited capabilities of interaction and differences in human usage habits in mobile clients and shared memory resources in high performance computing, due to the large amount of collaborating computational resource, deteriorate the system performance and final quality of service regardless the power effort. Thus system design for peak performance causes power waste during lower performance requirements. Under these circumstances the same QoS can be achieved by a lower system performance and power effort. Techniques that reduce the overall power consumption by adapting at run time the system performance to variation in application and usage requirements take the name of dynamic power management.

Even thus this concept is rather clear in HPC where recent processors allow core frequency and supply voltage to dynamically adapt the system performance and the power consumption to workload requirements, in mobile devices the same concept of performance adaptation can be use to tune the QoS offered to the human capabilities of recognize it.

Power management techniques for mobile clients The first task of mobile clients devices is to interact with the final users and thus the interfaces play a crucial role on its behavior. Due to human habits the visual one is the most important and this can be seen directly from the power consumption, where the display system accounts for the majority of the total device

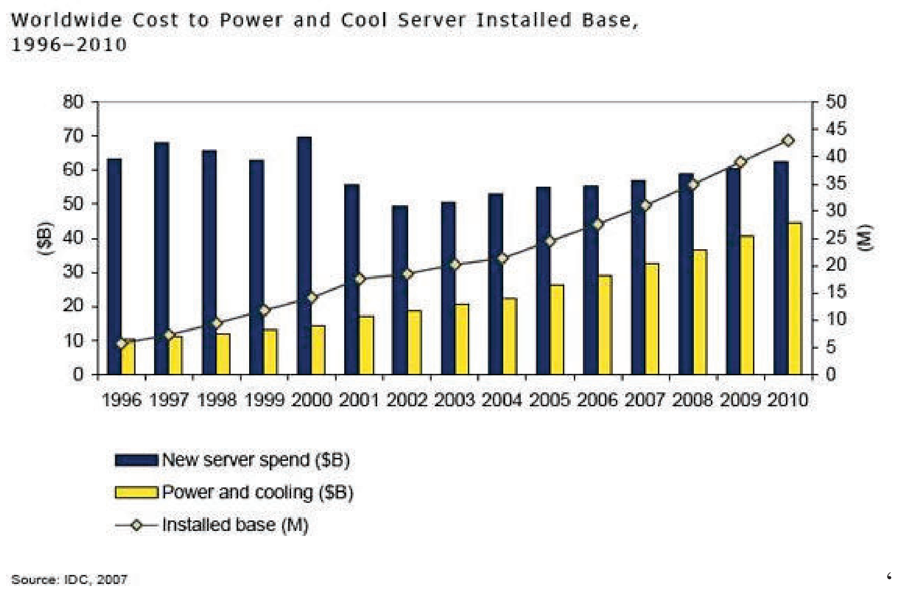


Figure 1.3: Worldwide IT spending on servers, power and cooling

power consumption. Indeed despite the technological improvements in Liquid Crystal Display's (LCD) technology, LCD power consumption is still one of the major limitations to the battery life of mobile appliances, such as portable media players, smart phones, navigation and gaming devices (6). Pushed by the market of multimedia applications, portable devices need to render high definition videos and pictures on larger and larger screens. Since LCD power consumption depends on backlight and pixel matrix driving circuits, which are both proportional to the panel area, LCD contribution to the total system power consumption is going to be considerable even in future mobile devices.

Several techniques have been proposed to face the issue of LCD power. The most promising ones achieve substantial savings by dynamically adapting backlight intensity levels while compensating for the ensuing visual quality degradation with image pixel transformations(7, 8, 9). Due to the display system non-idealities, this techniques unfortunately produces a quality degradation in the rendered images. The entity of this quality loss depends on several factors: the pixel transformation, the final backlight intensity level, the LCD display electrical and optical properties, and finally the human visual system (HVS) features. Indeed, due to the high non-linearities and complexity of the HVS, the same distortion level of physical quantities, such as luminosity or chromatic differences, can be perceived in different ways (4).

Unfortunately there is not a clear assessment of quality of service in the state-of-the-art techniques and this is the main limiters for the applicability of these techniques in today devices.

Power management techniques for high performance computing At the other end of ICT market, high performance computing and data server platforms only cares about performance. The evolution of these platforms along the Moore's law had encountered a series of technological wall, namely power, memory and now thermal wall. Mainly with technological scaling and clock frequency increases, the power densities on the silicon die surface has dramatically increased, as reported in Fig. 1.4.

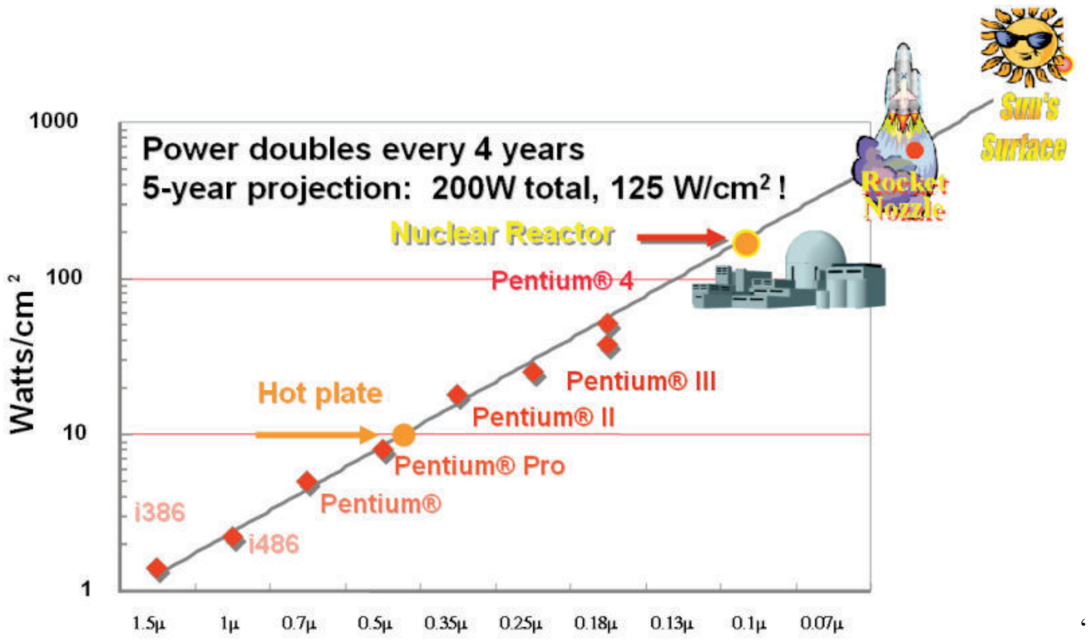


Figure 1.4: Silicon power density trend

As direct consequence of that die temperature abruptly increases along years and new challenges on future design comes. The consequent stop on clock frequency grow forces a paradigm shift in processors architectures to still accomplishing the performance demand coming from the society. Indeed performance is now scaled by parallelizing execution on multiple cores: the whole industry and research community has enthusiastically embraced the multi-core revolution. Multi-core architectures can effectively leverage Moores law, by packing more cores in the same silicon area, thereby achieving higher parallel computational density as silicon technology keeps improving with the pace dictated by Moores law. Unfortunately this trend cannot continue indefinitely: increasing performance per unit area (or volume) comes unavoidably with increasing power density, which becomes heat, leading to degradation, acceleration of chip aging and increase in cooling costs. We are at the brink of a thermal crisis: cooling and heat management are rapidly becoming the new major limiters for high performance processors.

The research community and leading electronics companies had invested significant effort in developing thermal control solutions for computing platforms, overcome the overhead imposed by worst case thermal design in both cost and performance. On-chip silicon performance and usage show highly spatial and temporal variability; this reflects in the inefficiency of static approaches to tackle thermal issues, causing HW damage, reliability loss and cooling costs overhead. Thus the main objectives of dynamic thermal control is to adapt at run-time the resource usage to achieve an optimal trade-off between performance, QoS, power consumption and cooling effort, while at the same time ensuring safe working temperature under all working conditions. Todays multicore processors include hardware support for dynamic power and thermal management, based on introspective monitors, sensors and performance knobs. This infrastructure provides the sensors and the actuators for feedback control management policies. Threshold-based thermal control techniques are the most widely used today in both HW and SW layers (10, 11). To avoid chip damage, microprocessors automatically shut down with a hardware trigger when core temperature crosses a safe temperature limit. In addition, operating systems exploit the HW knobs (core voltage and frequency and power gating) and temperature sensors readings with threshold-based feedback control to avoid emergency thermal shutdown. These techniques have major drawbacks, particularly for multi-scale systems such as many-core. Hardware-triggered temperature capping brings major performance degradation or even application failure, while OS-based capping cannot safely bound the run-time temperature and it has been shown to worsen the thermal cycles and system reliability. To overcome these limitations, classic feedback controllers have been proposed to adjust hardware knobs smoothly (11, 12). However simple feedback controllers, such as PID (proportional integral derivative) controllers are not sufficiently flexible and powerful for the complex multimodal dynamic behaviour of many-core heterogeneous SoCs (13). Recently Model Predictive Controllers (MPC) have been proposed, which rely on a system model to predict the future temperature while finding the optimal control action by solving a constrained optimization problem for one or more control steps in the future. Provided that a thermal model is available, MPCs can guarantee a reliable temperature capping in any working condition, minimizing the performance penalty at the chip level (13, 14, 15).

1.2 Thesis Contributions

In this thesis I carry out a thorough study of system level power management techniques starting from mobile clients toward high performance computing.

On the first part, mobile clients has been studied, with particular effort on the display subsystem that accounts for the majority of the entire device power consumption. Significant saving can be achieved by adopting dynamic backlight scaling techniques (DBS), unfortunately state-of-the-art techniques cannot keep final visual QoS under control since they policy are scarcely correlated to display image quality.

The first contribution of this work has been the development of a new framework to asses the visual quality performance of backlight scaling algorithms. Our framework considers both the real embedded video chain non-idealities and the human visual system (HVS) insights. The real embedded video chain non-idealities are accounted through an accurate display modelling performed on a real embedded platform for multimedia application (5). The HVS peculiarities are considered in the implemented image quality metric. This metric compares the original and the DBS distorted images, thus producing an index which is proportional to the perceived degree of similarity between the two images (4).

This framework has been validated and utilized to analyze the behavior of state-of-the-art DBS techniques. Experimental results show that none of the DBS techniques available today is fully capable to provide a controlled, constant quality loss, and that there is significant room for improvement in this direction.

Second main contribution has been the definition of a novel on-line technique for DBS which limits the degradation of the final rendered image in a robust fashion. Our approach is HVS and image structure-aware. We provide a real implementation of the proposed framework on a Freescale application development board based on the i.MX31 processor. We carried out a full characterization of the overall system power consumption versus QoS. Experimental results validate our technique and highlight the robustness of our approach in terms of performance, energy efficiency and image quality trade-offs.

In the second part of the thesis, instead, I analyze the opposite ICT scenario by exploring system power management techniques in High Performance Computing and Data Center to highlight room for improvement.

The first main contribution as been the development of a framework to analyze the performance and power consumption of parallel benchmarks and application in a server-like multicore real platform. The second main contribution was to combine the information learn from this analysis, such as power model, thermal model and functional behavior with a multicore simulation environment to create a complete virtual platform with the following key features:

- High-level functional modelling, where corresponding power and thermal models are included, allowing fast, but "physically accurate" simulations of the complete multicore platform.
- Integration with a suitable framework to perform co-simulation of multicore SoCs with control algorithms for rapid prototyping and to achieve flexibility and effectiveness in the design, testing and review of control strategies, exploiting control-theory tools.

First the virtual platform as been used to test state-of-the-art thermal management techniques such as classical control approach. Then we used it do develop a novel solution based on more advance and predictive control policies. This solution is line breaking technology and represent the third main contribution of this thesis. It is distributed, and combines energy minimization, MPC based thermal capping and thermal model self-calibration. It has the following key features:

- First according to the incoming task workload characteristics, each local node first selects the minimum frequency (f_{EC}) that preserves the performance within a tolerable overhead.
- Second, if necessary each local MPC controller trims the frequency to ensure a safe working temperature. Local controllers jointly optimize global system operation by exchanging a limited amount of information at run-time on a neighbourhood basis.
- Third we address model uncertainty by self-calibration: each thermal controller node extracts automatically the local thermal model by applying a set of training stimuli and monitoring the thermal response of the neighbourhood area. The distributed controller strategy combined with the distributed thermal model calibration phase allow us to take advantage of the parallelism of the underlying multi-core platform by running different instances of the controller and self-calibration routine in parallel.

1.3 Thesis Overview

We describe in this section the organization of the remainder of this thesis. Fist mobile devices power consumption and display technology are analyzed in chapter 2. Dynamic luminance scaling (DBS) techniques are first introduced in chapter 3. Then in the same chapter a novel framework for evaluating the visual performance of the state-of-the-art one is presented, highlighting their limitations. Chapter 4 describes a novel and innovative DBS techniques to override them. Secondly high performance computing platform and thermal and power management state-of-the-art solution are presented in chapter 5. In the same chapter, section 5.2 describe the developed framework for accessing at the performance management knobs and sensors of modern server platforms. The enabled functionalities are used in section 5.3 to characterize future parallel benchmarks for a target multicore machine. Section 5.4 describe a characterization phase done to extract power and temperature empirical models suitable for the following chapter. Indeed all these analysis and data converge in chapter 6 and 7. The first one describes the novel virtual platform developed to fast simulate and prototype novel thermal and power management techniques, whereas the second describes a novel distributed and self-calibrating thermal and energy management technique. Finally in chapter 9 final remarks are drawn whereas chapter 8 contains the list of publications.

Bibliography

[1] www.greenpeace.org, Make IT Green, http://www.greenpeace.org/international/-Global/international/planet-2/report/2010/3/make-it-green-cloud-computing.pdf 1

[2] www.smart2020.org, SMART 2020: Enabling the low carbon economy in the information age. http://www.smart2020.org/_assets/files/01_Smart2020ReportSummary.pdf 1

[3] Environmental Protection Agency, EPA report to Congress on server and data center energy efficiency (Aug. 2007). http://www.energystar.gov/ia/partners/prod_development/-downloads/EPA_Datacenter_Report_Congress_Final1.pdf 2

[4] Wang Z., Bovik A. C., Sheikh H. R. and Simoncelli E. P. Image quality assessment: From error visibility to structural similarity IEEE Trans. on Image Process., Vol. 13, no. 4, pp. 600-612, Apr. 2004. 4, 7

[5] i.MX31: Multimedia Applications Processor http://www.freescale.com 7

[6] Findlay Shearer Power management in mobile devices Wiley ed. 4

[7] Iranli A., Lee W. and Pedram M. HVS-Aware Dynamic Backlight Scaling in TFT-LCDs. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 14, N.10, 2006. 4

[8] Cheng L., Mohapatra S., El Zarki M., Dutt N. and Venkatasubramanian N. Quality-based backlight optimization for video playback on handheld devices. Adv. MultiMedia 2007, 1 (Jan. 2007), 4-4. 4

[9] Cheng W., Hsu C. and Chao C. Temporal vision-guided energy minimization for portable displays. In Proceedings of the 2006 international Symposium on Low Power Electronics and Design, ISLPED ’06. 89-94, 2006. 4

[10] S. Gunther, F. Binns, D. Carmean, J. Hall. Managing the Impact of Increasing Microprocessor Power Consumption. Intel Technology Journal, Q1, Feb. 2001. 6

[11] K. Skadron, T. Abdelzaher, M. R. Stan, Control-theoretic techniques and thermal-RC modelling for accurate and localized dynamic thermal management. IEEE High-Performance Computing Architecture, 2002. 6

[12] M. Kadin, S.Reda, A. Uht. Central vs. distributed dynamic thermal management for multi-core processors: which one is better?. ACM Great Lakes symposium on VLSi, 2009. 6

[13] Y. Wang, K. Ma, X. Wang. Temperature-Constrained Power Control for Chip Multiprocessor with Online Model Estimation. ACM International symposium on Computer architecture, 2009. 6, 7

[14] X. Wang, M. Kai, Y. Wang. Adaptive Power Control with Online Model Estimation for Chip Multiprocessors. IEEE Transactions on Parallel and Distributed Systems, 2011. 7

[15] F. Zanini, D. Atienza, L. Benini, G. De Micheli. Multicore Thermal Management with model predictive control. IEEE European Conference Circuit Theory and Design, 2009. 7

Chapter 2

Portable Device Display Technologies

2.1 Overview

In this section first is analyzed the mobile device power distribution (Section2.2). A significant portion of it is used by the LCD display system to render visual content. An LCD display system is composed of an LCD panel, a frame buffer memory, an LCD controller, and a backlight inverter and lamp or light-emitting diode (LED). High-resolution, high-color LCDs require large LCD panels, high-wattage backlight lamps, and large-capacity frame buffer memories, which together lead to high-power consumption. The backlight concept and technology is described in Section2.3 whereas display technology are described in Section2.4.

The processor and the memory are in power-down mode during the slack time, but the display components are always active mode, for as long as the display is turned on. This makes the LCD backlight the dominant power consumer, with the LCD panel and the frame buffer coming a second and third in power consumption. A modern mobile device requires a lot of computing power. With interactive applications, such as a video telephony or an assisted GPS, an even higher portion of the energy will be consumed by the display system.

2.2 Mobile Device Power Distribution

Figures 2.1, 2.2, 2.3 indicate pie charts that illustrate the mobile device power distribution ranging from a legacy mobile device (voice only) to a smartphone/multimedia mobile device to a gaming targeted mobile device. Convergence of features is driving new application processing and visual display requirements.

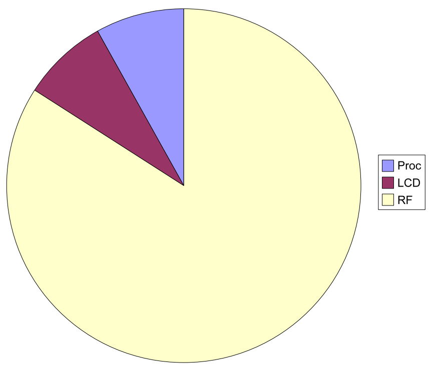


Figure 2.1: Legacy Handset Power Distribution Radio Frequency (RF) Dominated Power Consumption in Legacy (Voice Only) Handsets

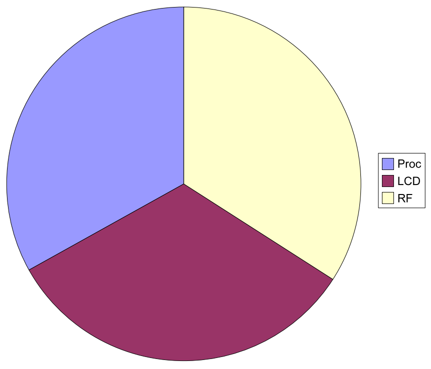


Figure 2.2: Feature Rich Handsets Power Distribution. More Equitable Power Consumption Distribution in Smartphone/Multimedia Mobile Devices

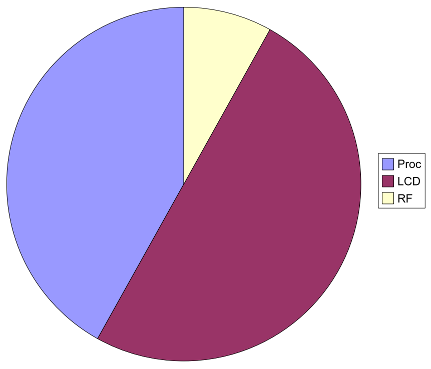
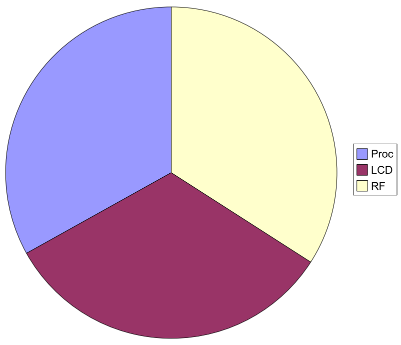
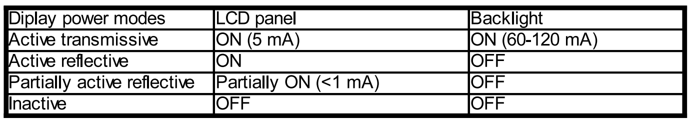


Figure 2.3: Game Oriented Phone Power Distribution. Power Distribution for Single Game Players, Dominated by the Display and Processing





Diplay power modes	LCD panel	Backlight
Active transmissive	ON (5 mA)	ON (60-120 mA)
Active reflective	ON	OFF
Partially active reflective	Partially ON (<1 mA)	OFF
Inactive	OFF	OFF

Figure 2.4: Most Power is Consumed When the Backlight is On

2.3 Backlights

Backlight mechanical design has become very sophisticated, allowing very few LEDs to be used with highly complex optical light pipes/light spreads which create uniform illumination of the LCD. Power consumption of the backlight is a critical issue. Now that white LEDs have replaced cold-cathode fluorescent lamps (CCFLs) for backlighting mobile device displays (and LEDs will soon replace CCFLs in laptop also), the next major innovation is the RGB LEDs, which will significantly enlarge the color gamut of the backlight, and therefore the display itself. It is claimed that an image that has higher color saturation (larger color gamut) looks brighter than a lower color saturation image. Engineers will take advantage of this effest to lower backlight power consumption. The RGB LED backlight will be a great benefit to mobile device displays.

Many mobile devices are equipped with color thin-film transistor (TFT) liquid-crytal displays (LCDs). A quality LCD system is now the default configuration for handheld embedded system. An LCD panel does not illuminate itself and thus requires a light source. A transmissive LCD uses a backlight, which is on of the greatest consumer of power of all system components. A reflective LCD uses ambient light and a reflector instead of the backlight. However, reflective LCD is not suitable for quality displays, and complementary use f ambient light and the backlight, named transflective LCD, is used for small handheld devices. When the backlight is turned off, a transmissive LCD displays nothing but black screen; even transflective screens are barely legible without the backlight.

Most useful applications require the backlight to be on. Figure 2.4 indicates the display power modes and the current drawn when the backlight is turned on (1).

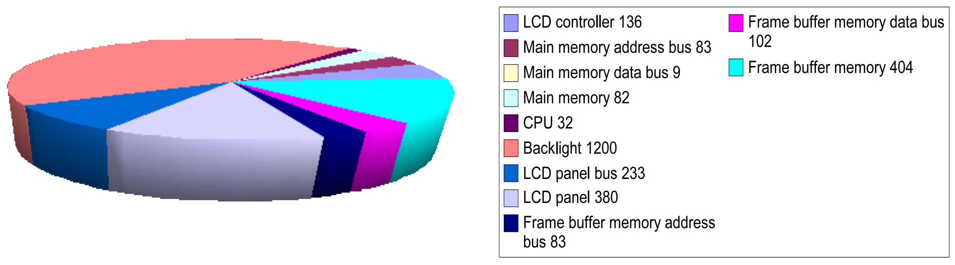


Figure 2.5: System Wide Power Consumption (2)

Figure 2.5 indicates the system level components contribution to the power consumption. Note the power consumed by the backlight.

There are many techniques employed to conserve energy consumed by a display system. Ambient luminance affects the visibility of LCD TFT panels. However, by taking account of this, backlight auto regulation (3) can also reduce the average energy requirements of the backlight. Simultaneously brightness and contrast scaling (4) further enhances image fidelity with a dim backlight, and thus, permits an additional reduction in backlight power. Even more aggressive management of the backlight can be achieved by modification of the LCD panel to permit zoned backlighting (5). Additional energy conserving techniques include dynamic luminance scaling (DLS), dynamic contrast enhancement (DCE), and backlight autoregulation. These techniques will be discussed later in this chapter.

2.4 Display Technologies

Display technologies such as backlight LCDs, reflective LCDs, electroluminescent (EL) displays, organic LEDs (OLEDs), and electrophoretic displays (EPD) objective is to achieve paper-like viewing displays.

There are four primary approaches to flat-panel displays. Three are illustrated in Figure 2.6:

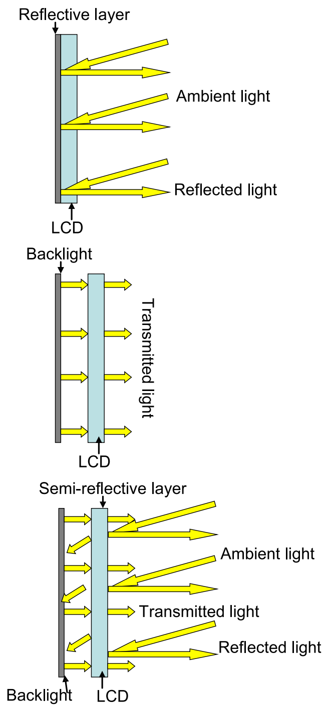


Figure 2.6: Approaches to Displays

1. Transmissive displays work by modulating a source of light, such as a backlight, using an optically active material such as a liquid-crystal mixture.

2. Emissive displays such as OLEDs make use of organic materials to generate light when exposed to a current source.

3. Reflective displays work by modulating ambient light entering the display and reflecting it off a mirror-like surface. Until recently, this modulation has typically been accomplished using liquid-crystal mixtures or electrophoretic mixture.

4. Transflective displays are a hybrid combination of a transmissive and reflective display. This technology was developed to provide sunlight viewability for transmissive displays. Being a compromise however, this type of display technology offers a compromised viewing experience.

Reflective displays were invented primarily to address the shortcomings of transmissive and emissive displays, namely power consumption and poor readability in bright environments.

Existing Technologies

Parameter	Units	Reflective STN	Reflective TFT	Backlight TFT	EPD	OLED
Color		B&W	Color	Color	Limited Color	Color
Resolution		Low	Good	Good	Good	Good
Sunlight readable	80% contrast	Yes,good	Yes,good	No	Yes,excellent	Yes,poor
Video response speed	ms	100	20	20	500	1/1,000
Static power including drivers	mW	15	20	200	0	200
Thickness	mm	5	6	9	1.25	3
Environmental	Temperature Range (°C)	10 - 30	10 - 30	10 - 30	0 - 50	10 - 50

Figure 2.7: Comparison of Display Technologies

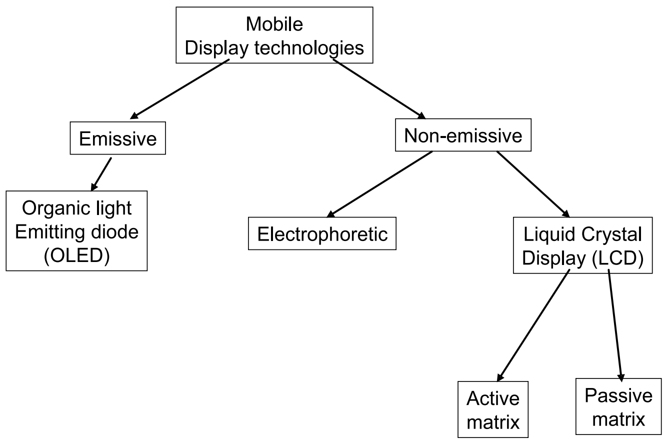


Figure 2.8: Classification of Mobile Device Displays

Since transmissive LCDs require a power-hungry backlight and emissive OLEDs require a constant power source to generate light, it makes it difficult for designers of these technologies to reduce power consumption without changing the image content. This is especially important for battery-powered portable devices such as mobile phones, PDAs, digital music players, digital cameras, GPS units, and mobile gaming devices. With efficient use of ambient light, reflective displays eliminate the backlight unit and offer both significant power savings and a thinner display module (2.7).

Mobile device display technologies can be separated into emissive and non emissive classes. This classification is expanded and illustrated in Figure 2.8.

2.4.1 TFT LCD

There are three types of TFT LCD panels. In transmissive LCDs, a backlight illuminates the pixels from behind. Transmissive LCDs offer a wide color range and high contrast, and are typically used in laptops. They perform best under lighting conditions ranging from complete darkness to an office environment.

Reflective LCDs are illuminated form the front. Reflective LCD pixels reflect incident light originating from the ambient environment or a frontlight. Reflective LCDs can offer very low-power consumption (especially without frontlight) and are often used in small portable devices such as handheld games, PDAs, or instrumentation. They perform best in a typical office environment or in brighter lighting. Under a dim lighting, reflective LCDs require a frontlight.

Transflective LCDs are partially transmissive and partially reflective, so they can make use of environment light or backlight. Transflective LCDs are common in devices used under a wide variety of lighting conditions, from complete darkness to sunlight.

Transmissive and transflective LCD panels use very bright backlight sources that emit more than 1,000 cd/m2. However, the transmittance of the LCD is relative low, and thus the resultant maximum luminance of the panel is usually less than 10% of the backlight luminance.

Theoretically, the backlight and the ambient light are additive. However, once the backlight is turned on, a transflective LCD panel effectively operates in the transmissive mode because the backlight source is generally much brighter than the ambient light.

As stated earlier, backlighting for LCDs is the single biggest power draw in portable displays. This is especially true in bright environments where the backlight has to be switched to the brightest mode. Given how difficult it is to view a typical transmissive LCD in a sunlit environment, LCD developers have been working diligently on reflective LCDs.

Currently there are a number of portable devices using transflective LCDs. The transflective display was invented to improve the performance of the transmissive LCD outdoors, where bright ambient light quickly overpowered the LCD backlight, making the display hard to read. It was also configured to address the shortcomings of a purely reflective LCD in a dark environment. The transflective display employs a reflector that lets some light through from a backlight. Using such an element, the display can be used in the dark where the backlight provides illumination through the partly transmissive reflecting element. In the bright outdoors, the backlight can be switched off to conserve power and the mirrored portion of the reflector allows the LCD to be viewed by making use of the ambient light. Theoretically, the transflective display appears to fix the shortcomings of the purely reflective and transmissive displays. But in reality, this approach is a compromise and offers poor viewing experience.

Figure 2.9 shows the complexity of an LCD. The extensive use of optical films such as polarized and color filters, as well as the TFT element which itself requires several process step fabricate. Since LCDs work with polarized light, the necessity of using a polarizer limits

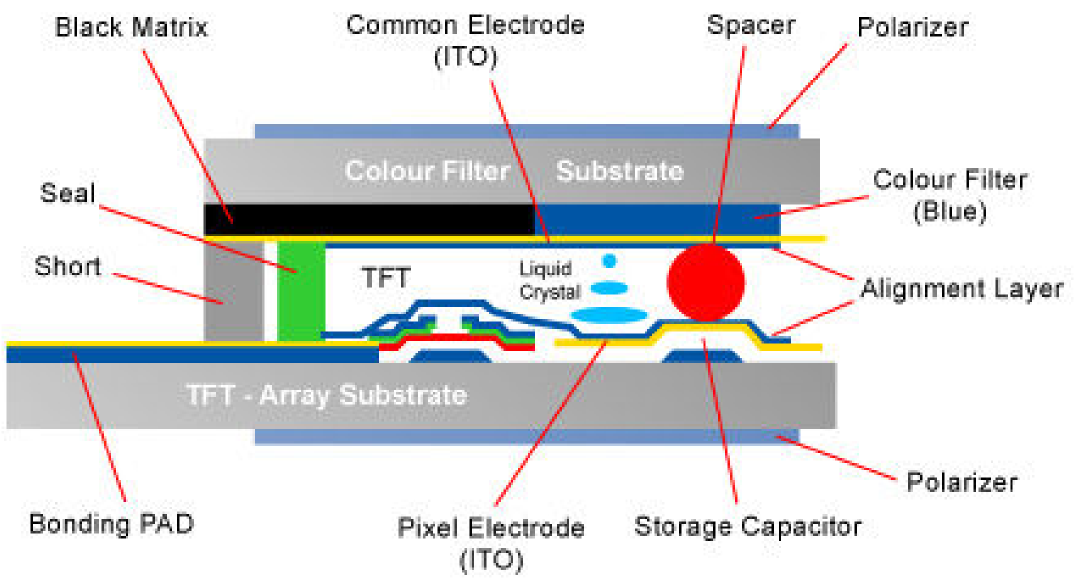


Figure 2.9: LCD Structure

the amount of light that is reflected or transmitted from the display. The additional layers, such as the color filter, reduce light even further. Consequently, today's LCDs require brighter backlight in order to be readable, whether in total darkness or in the bright sunlight. These brighter backlights lead to greater power consumption.

Despite the ever increasing advantages in LCD's technology, their power consumption is still one of the major limitations to mobile. There is a clear trend towards the increase of LCD size to exploit the multimedia capabilities of mobile devices that can receive and visualize high deifnition video and pictures. Multimedia applications running on these deivces impose LCD screen sizes of 2.2-3.5 in. and more to display video sequences and pictures with the required quality.

2.4.2 OLED

Similar to LCDs, OLEDs can be constructed using a passive or active matrix. The basic OLED cell structure is comprised of a stack of thin organic layers that are sandwiched between a transparent anode and a metallic cathode. When a current passes between the cathode and anode, the organic compounds emit light (see Figure 2.10). Unlike LCDs, passice matrix

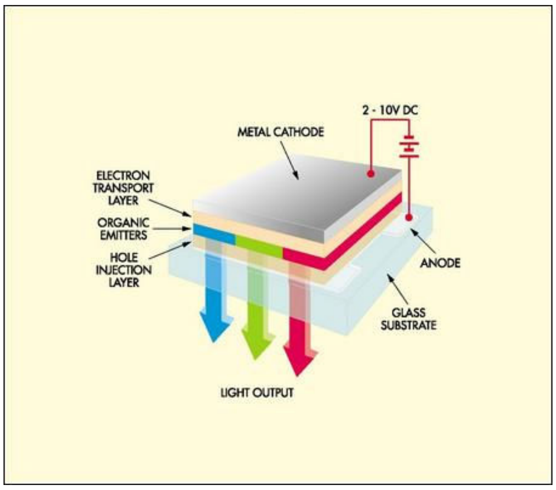


Figure 2.10: OLED Structure

OLEDs does not suffer from lower contrast or slower response time. However, OLEDs offer several advantages over LCDs.

The obvious advantage is that OLEDs are like tiny light bulbs, so they do not need a backlight or any other external light source. They are less than one-third of the bulk of a typical color LCD and about half the thickness of most black-and-white LCDs. The viewing angle is also wider, about 160. OLEDs also switch faster than LCD elements, producing a smoother animation. Once initial investments in new facilities are recouped, OLEDs can potentially compete at an equal or lower cost than incumbent LCDs.

Despite these advantages, OLEDs have a relatively short lifespan and as power/brightness is increased the life is reduces dramatically. This is especially true for the blues, which lose their color balance over time. In addition, only low-resolution OLED displays can use passive matrix backplanes and higher resolutions require active matrices, which need to be highly conductive since OLEDs are current driven. Typically, low temperature poly silicon (LTPS) backplanes are used which adds cost and complexity. These conductors are also highly reflective requiring the OLED designers to add a circular polarizer on the front if the display reducing the efficiency if the display and increasing the cost. Finally, as is the case with all emissive displays, OLED displays have poor readability in environments such as the bright outdoors.

Bibliography

[1] T.Botzas. PenTile RGBW display technology for meeting aggressive power budgets in high resolution multimedia mobile applications. In International Wireless Industry Consortium. 2005. 15

[2] I.Choi. J2ME LBPB (Low Power Basis Profile of the Java 2 Micro Edition) Computer Systems Laboratory, 2002 vii, 16

[3] F.Gatti and A.Acquaviva and L.Benini and B.Ricco. Low power control techniques for TFT LCD displays. In Proceedings of the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems, 2002 16

[4] W.C.Cheng and Y.Hou and M.Pedram. Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling In Proceedings of DATE 04, 2004 16

[5] J.Flinn and M.Satyanarayanan. Energy-aware adaptation for mobile applications. In Proceedings of the Symposium on Operating Systems Principles, 1999. 16

Chapter 3

Visual quality analysis for dynamic backlight scaling in LCD systems

3.1 Overview

With the trend toward high-quality large form factor displays on high-end handhelds, as introduced in previous chapter2 LCD backlight accounts for a significant and increasing percentage of the total energy budget. Substantial energy savings can be achieved by dynamically adapting backlight intensity levels while compensating for the ensuing visual quality degradation with image pixel transformations. Several compensation techniques have been recently developed to this purpose, but none of them has been fully characterized in terms of quality losses considering jointly the non-idealities present in a real embedded video chain and the peculiar characteristics of the human visual system (HVS). We have developed a quality analysis framework based on an accurate embedded visualization system model and HVS-aware metrics. We use it to assess the visual quality performance of existing dynamic backlight scaling (DBS) solutions. Experimental results show that none of the DBS techniques available today is fully capable to keep quality loss under control, and that there is significant room for improvement in this direction.

3.2 Introduction

Despite the constant evolutions in Liquid Crystal Display's (LCD) technology, LCD power consumption is still one of the major limitations to the battery life of mobile appliances, such as portable media players, smart phones, navigation and gaming devices (14). Pushed by the market of multimedia applications, portable devices need to render high definition videos and pictures on larger and larger screens. Since LCD power consumption depends on backlight and pixel matrix driving circuits, which are both proportional to the panel area, LCD contribution to the total system power consumption is going to be considerable even in future mobile devices.

Several techniques have been proposed to face the issue of LCD power. The most promising ones achieve substantial savings by dynamically adapting backlight intensity levels while compensating for the ensuing visual quality degradation with image pixel transformations. Due by the display system non-idealities, this kind of techniques unfortunately produces a quality degradation in the rendered images. The entity of this quality loss depends on several factors: the pixel transformation, the final backlight intensity level, the LCD display electrical and optical properties, and finally the human visual system (HVS) features. Indeed, due to the high non-linearities and complexity of the HVS, the same distortion level of physical quantities, such as luminosity or chromatic differences, can be perceived in different ways (9).

Two main approaches have been developed to control the visual quality loss in DBS techniques. The first is image dependent, as depicted in Fig.3.1.a: it computes on-line the amount of backlight scaling on every frame using a simple image distortion metric as a constraint. Techniques following this approach aim at keeping a constant distortion by maintaining the frame-by-frame computed quality performance metric as constant as possible (1), (2), (7), (8). Unfortunately the simplified quality metrics adopted do not account for both the HVS peculiarities and the display system non-idealities. Hence there is no formal relationship between the adopted distortion metrics and the real perceived quality degradation.

Accounting both the HVS peculiarities and the display system non-idealities in the on-line visual quality distortion metric is very hard, due to their computational complexity. Therefore, image independent approaches have been developed, in which the relationship between perceived image distortion and DBS transformation is statistically analysed off-line (Fig.3.1.b). These approaches consider image degradation as dependent only on the applied DBS transformation. Authors in (3), (4), (5), (6) use a set of benchmark images to calculate an image-independent empirical function which relates the image degradation levels to the transformation parameter applied to the image itself. The distortion degradation level is calculated off-line through a quality metric which considers the HVS peculiarities. This approach does not consider the perceived image distortion as a function of the image itself, in other words it implicitly assumes that a constant backlight reduction would produce, after compensation, a constant

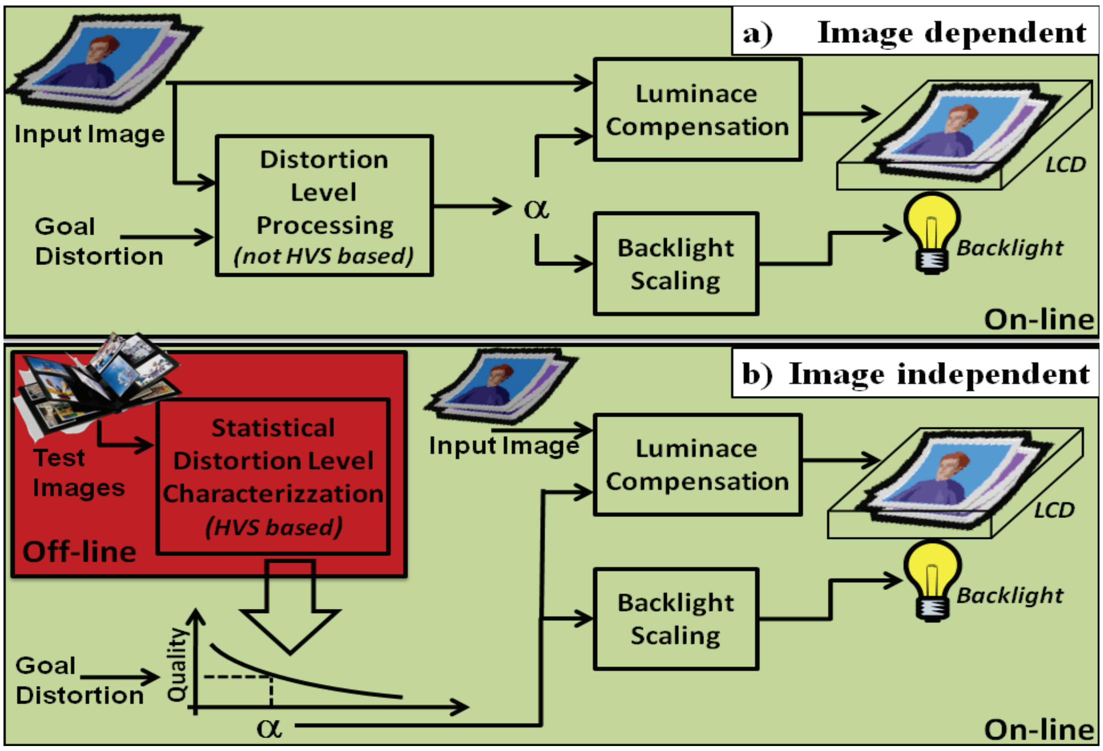


Figure 3.1: DBS technique approaches : image dependent (a) vs image independent(b)

quality degradation. This assumption has not been accurately assessed and validated: it is not clear if this methodology produces different perceived distortion levels over different images while aiming to keep it constant.

The main contribution of this work is the development of a new framework to asses the visual quality performance of backlight scaling algorithms. Our framework considers both the real embedded video chain non-idealities and the HVS insights. The real embedded video chain non-idealities are accounted through an accurate display modelling performed on a real embedded platform for multimedia application (11). The HVS peculiarities are considered in the implemented image quality metric. This metric compares the original and the DBS distorted images, thus producing an index which is proportional to the perceived degree of similarity between the two images (9).

This framework has been validated and utilized to analyse the behaviour of state-of-the-art DBS techniques. Experimental results show that none of the DBS techniques available today is fully capable to provide a controlled, constant quality loss, and that there is significant room for improvement in this direction.

3.3 Framework for Visual Quality Performance Evaluation

The evaluation of the visual quality performance for dynamic backlight scaling algorithms is not trivial due to LCD display non-idealities and HVS peculiarities. We developed a complete Matlab-based framework to overcome these difficulties. Fig.3.2 shows the framework block diagram.

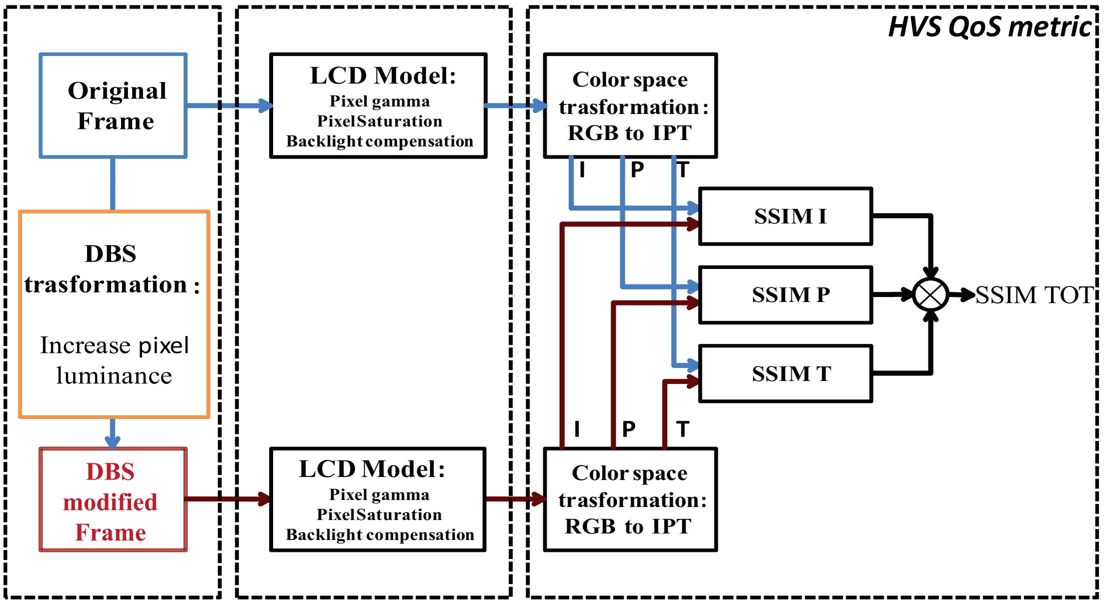


Figure 3.2: QoS matlab framework block diagram.

We can identify three main blocks:

1. DBS transformation: this block processes the original image or video frame with the DBS image transformation. The outputs are a modified image with increased pixels luminance and the related new backlight scaled value.

2. The LCD model: this block models the behaviour of a real embedded LCD panel and its non-idealities. From the RGB image and the backlight level produced by the DBS transformation block, it generates an image representing how the target image itself will look like on the LCD panel. This step is done for both the images: original and DLS transformed one.

3. HVS QoS metric: this block implements the HVS peculiarities and evaluates the differences between the two images from the LCD model block.

3.3.1 LCD Model

We can consider the luminance emitted from a x pixel of the LCD panel equal to:

$$L(x) = f(BL) * g(x) \tag{3.1}$$

Intuitively, this equation shows that the luminance emitted from a pixel depends on functions *f()* of the backlight (BL) and *g()* of the pixel value itself, which sets its transmittance. These two functions are related to the LCD panel used, and generally are non-linear.

3.3.1.1 LCD display characterization

This section describes our LCD characterization methodology focused on quantifying the LCD optical properties and to evaluate the non-linearity effects existing between real displayed quantities and digital values descriptions. More in detail, to analyse:

1. The relationship between pixel digital values in RGB space and emitted light intensities, *g()*.
2. The relationship between backlight values and emitted light intensities, *f()*.

As reference case we apply this general methodology to a TFT LCD display (Sharp LQ-035Q7DB02). We displayed a set of images on the LCD and measured the emitted light with a light sensor. To get a set of consistent measurements, the ambient light contribution was eliminated performing the tests within a dark room.

For the first test, we used a photodiode IPL 10530DAW as light sensor. The light sensor produces as output a voltage linearly proportional to the intensity of the incident light, emitted by the LCD. We measured the emitted light from the LCD displaying a monochromatic image which has only one RGB component which varies from 0 to 255, and the remaining components set to 0 (i.e. (RGB=X,0,0), (RGB=0,X,0), (RGB=0,0,X)).

Fig.3.3 shows the normalized light intensities on the y-axes and the normalized value of the three RGB components (X/255) on the x axes. The plot shows that the light emitted by pixels is non-linear with the digital RGB value, but it matches a polynomial law. We fitted the measured value (the dots in the plot) with the function:

$$L(x) \propto offset + K * (R,G,B)^{\gamma_{R,G,B}} \tag{3.2}$$

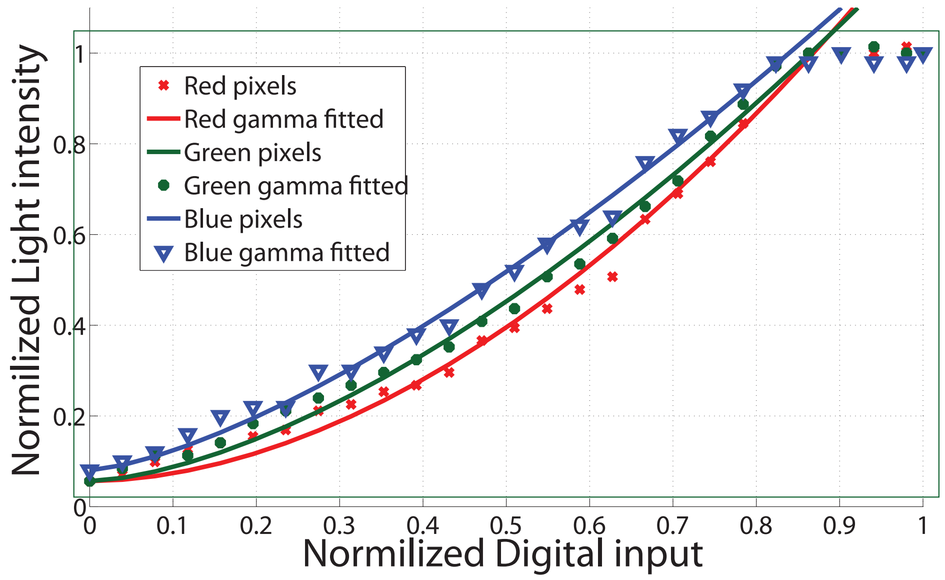


Figure 3.3: Light intensities for R,G,B pixels and relative gamma fit.

The backlight effect has been quantified by measuring its contribution on emitted light using the photodiode as light sensor. The results of this characterization are shown in Fig. 3.4: the dots show the measured data and the line shows the fitted equation.

The curve was obtained with the power fit:

$$L(x) \propto K * (Backlight)^{\theta} \tag{3.3}$$

3.3.1.2 LCD Model

We modeled the LCD panel behaviour using the results from the LCD characterization phase (see Section.3.3.1.1). The model takes as input an RGB image and a backlight value, and it produces as output a trichromatic image which has each pixel component proportional to the light intensity produced by the component itself.

The model can be split in three main blocks (see Fig.3.5):

1. The incoming image is transformed by the set of Equations 3.2, which account for the non-linearity of LCD light transmittance.

2. The input backlight is used to obtain a new value which considers the non-linearity showed in Eq.3.3.

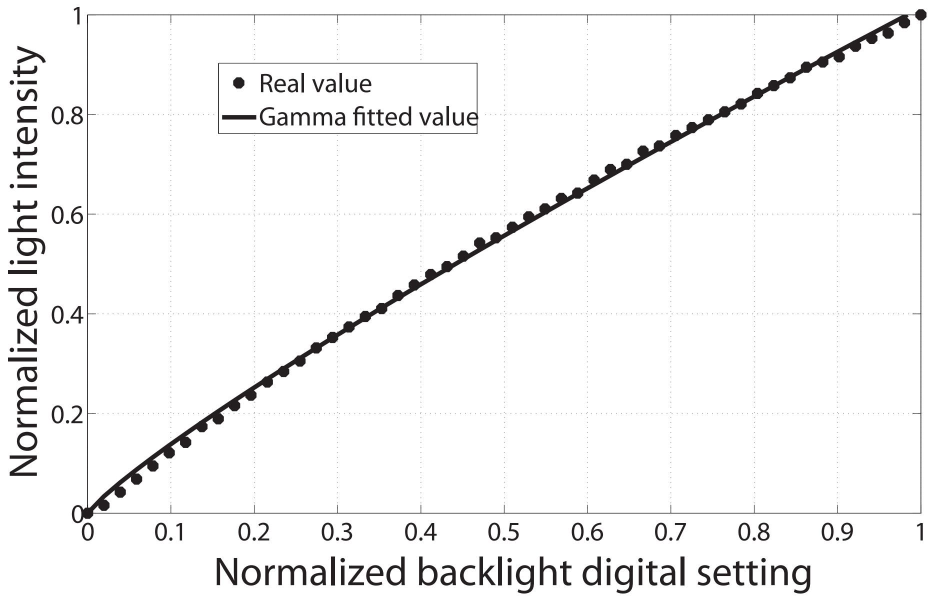


Figure 3.4: Normalized light intensity vs. normalized digital backlight value.

3. The model combines the transformed image and the new backlight value to compute the simulated displayed image.

3.3.2 DBS transformation

The overall goal of a DBS transformation is to dim the backlight while at the same time scaling the pixel transmittance in order to re-equilibrate the target pixel luminance. In the proposed framework we adopt a DBS technique that aims to preserve the pixel luminance and chroma, considering the display non-linearities highlighted in Section.3.3.1.1. The DBS transformation presented is a linear transformation of the pixels value, and can be applied through a pixel matrix operation that is suitable for an implementation on a real embedded system. From Eq.3.1, 3.2 and 3.3, we can formulate:

$$\begin{cases} L_R(x) \propto (R)^{\gamma_R} * (Backlight)^{\theta} & (3.4) \\ L_G(x) \propto (G)^{\gamma_G} * (Backlight)^{\theta} & (3.5) \\ L_B(x) \propto (B)^{\gamma_B} * (Backlight)^{\theta} & (3.6) \end{cases}$$

If we dim the Backlight value of a scaling factor β, we need also to scale each RGB pixel component by a set of α factor in order to obtain the same target luminance:

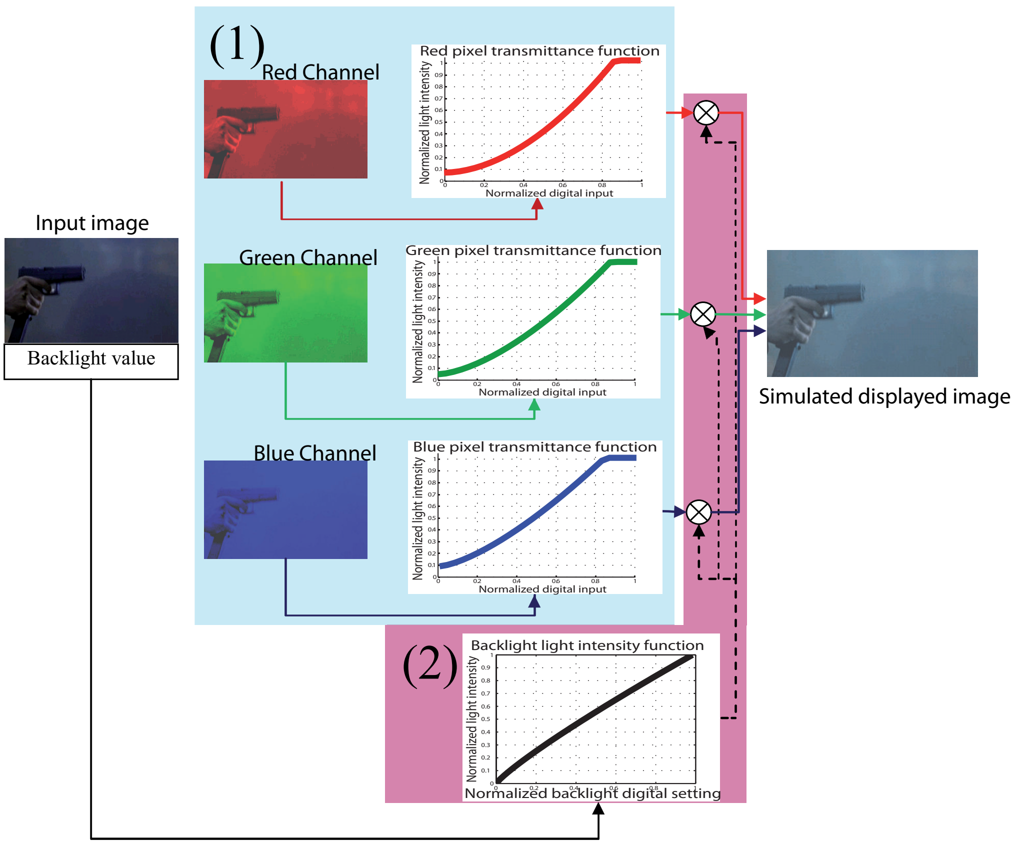

(1)
Red Channel
Red pixel transmittance function
Green Channel
Green pixel transmittance function
Blue Channel
Blue pixel transmittance function
Normalized light intensity
Normalized digital input
Input image
Backlight value
Simulated displayed image
(2)
Backlight light intensity function
Normalized backlight digital setting

Figure 3.5: LCD model schematic.

$$\begin{cases} L'_R(x) \propto (\alpha_R R)^{\gamma_R} * (Backlight/\beta)^{\theta} & (3.7)\\ L'_G(x) \propto (\alpha_G G)^{\gamma_G} * (Backlight/\beta)^{\theta} & (3.8)\\ L'_B(x) \propto (\alpha_B B)^{\gamma_B} * (Backlight/\beta)^{\theta} & (3.9) \end{cases}$$

From 3.4, 3.5, 3.6 and 3.7, 3.8, 3.9 we obtain:

$$\begin{cases} {\alpha_R}^{\gamma_R} = \beta^{\theta} \Rightarrow & \alpha_R = \beta^{\theta/\gamma_R} & (3.10)\\ {\alpha_G}^{\gamma_G} = \beta^{\theta} \Rightarrow & \alpha_G = \beta^{\theta/\gamma_G} & (3.11)\\ {\alpha_B}^{\gamma_B} = \beta^{\theta} \Rightarrow & \alpha_B = \beta^{\theta/\gamma_B} & (3.12) \end{cases}$$

The last equation shows that, for a given β backlight scaling factor, we need to compensate each component by a α_X[1] factor. Due to the bounded digital range [0..255] for each component, this operation is not able to compensate the luminance in the top end values, thus introducing a

[1]For simplicity in the following sections we will use only α_B,and we will call it α.

saturation effect. This can produce both a luminance dynamic range compression and a color distortion. The proposed framework considers all these distortion effects.

3.3.3 HVS QoS metric

This block evaluates the perceived quality degradation introduced in the final rendered image by the DBS technique. We used a well established visual QoS index : the structural similarity index metric (SSIM). This index has been demonstrated to be able to quantify the HVS perceived differences between two images (12) (9).

SSIM is based on the assumption that the human visual system is efficient in extracting structural information from the viewing field. The structural information of an image is an attribute which describes the structure of objects in the scene independently from average luminance and contrast. In Fig.3.6 we can see the schematic implementation of the SSIM metric. The average luminance, contrast and structural information are computed for both target and sample images and then combined together to produce the quality index.

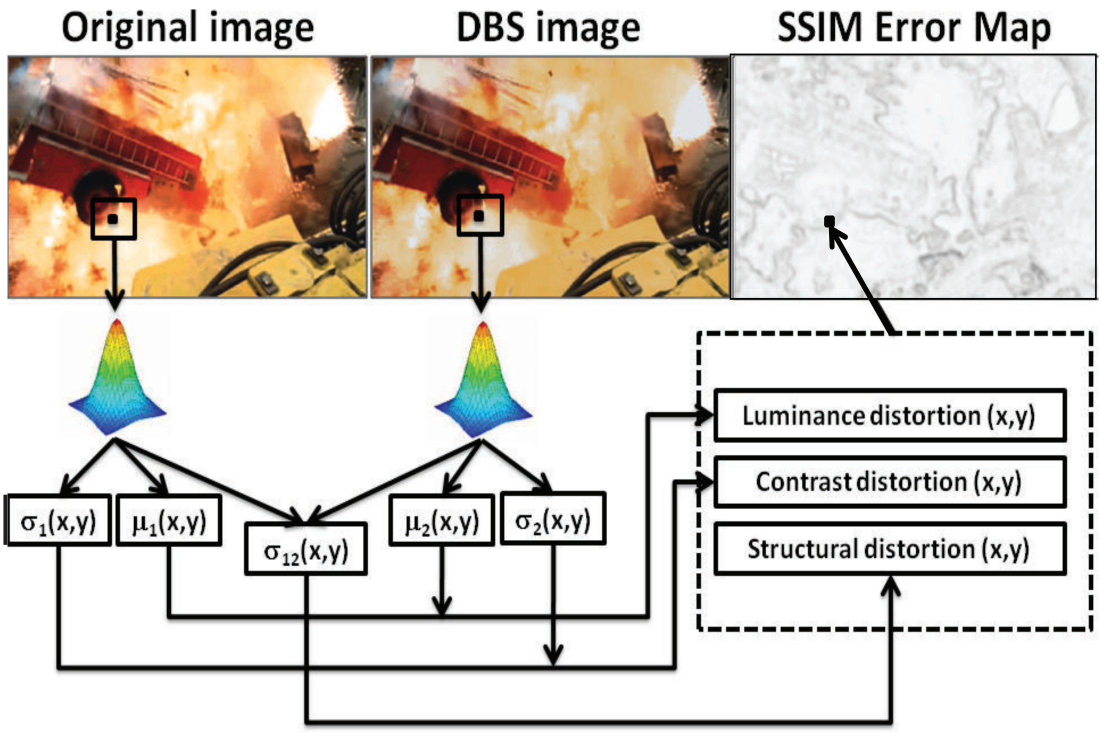

Original image
DBS image
SSIM Error Map
Luminance distortion (x,y)
Contrast distortion (x,y)
Structural distortion (x,y)
σ1(x,y)
μ1(x,y)
σ12(x,y)
μ2(x,y)
σ2(x,y)

Figure 3.6: Diagram of (SSIM) structural similarity measurement system.

SSIM index for monochromatic images is made of three similarity indexes, accounting for luminance, contrast and structural similarity, $l(X,Y)$, $c(X,Y)$ and $s(X,Y)$ respectively.

$$SSIM(X,Y) = l(X,Y) * c(X,Y) * s(X,Y) \tag{3.13}$$

To apply SSIM to color images, we first transformed the LCD model output images from the trichromatic (RGB) color space to the IPT color space (10) (13). In IPT each quantity is correlated to perceive lightness, chroma and hue. Then the SSIM index is computed for each I,P,T, component ($SSIM_I$, $SSIM_P$, $SSIM_T$). Finally we combine them together:

$$SSIM = SSIM_I * SSIM_P * SSIM_T \tag{3.14}$$

3.4 EXperimental Results

3.4.1 QoS framework performance

The presented framework has been tested to verify how our QoS index is sensitive to visual distortion. We asked to a set of human users to asses the visual similarity, in a side by side comparison, between original and distorted images at different SSIM quality levels. Results lead two important findings: first a constant SSIM index for different images means constant perceived distortion. Second, we correlated human observer scores with the SSIM index results. This helped indentifying four macro-regions in the SSIM index range: each region specifies a quality level. Table.3.1 shows the quality levels and their SSIM ranges. Fig.3.7 shows a visual example of the four quality regions for three test images.

SSIM Range	Quality Level
1 - 0.98	High quality
0.98 - 0.96	Medium quality
0.96 - 0.94	Low quality
≤ 0.94	Unacceptable

Table 3.1: QoS framework output quality regions.

3.4.2 Image independent DBS techniques

One approach to solve the DBS problem adopted in literature (3) (4) (5) (6), considers the distortion introduced by luminance dynamic range reduction ($\propto \alpha$) independent from the image

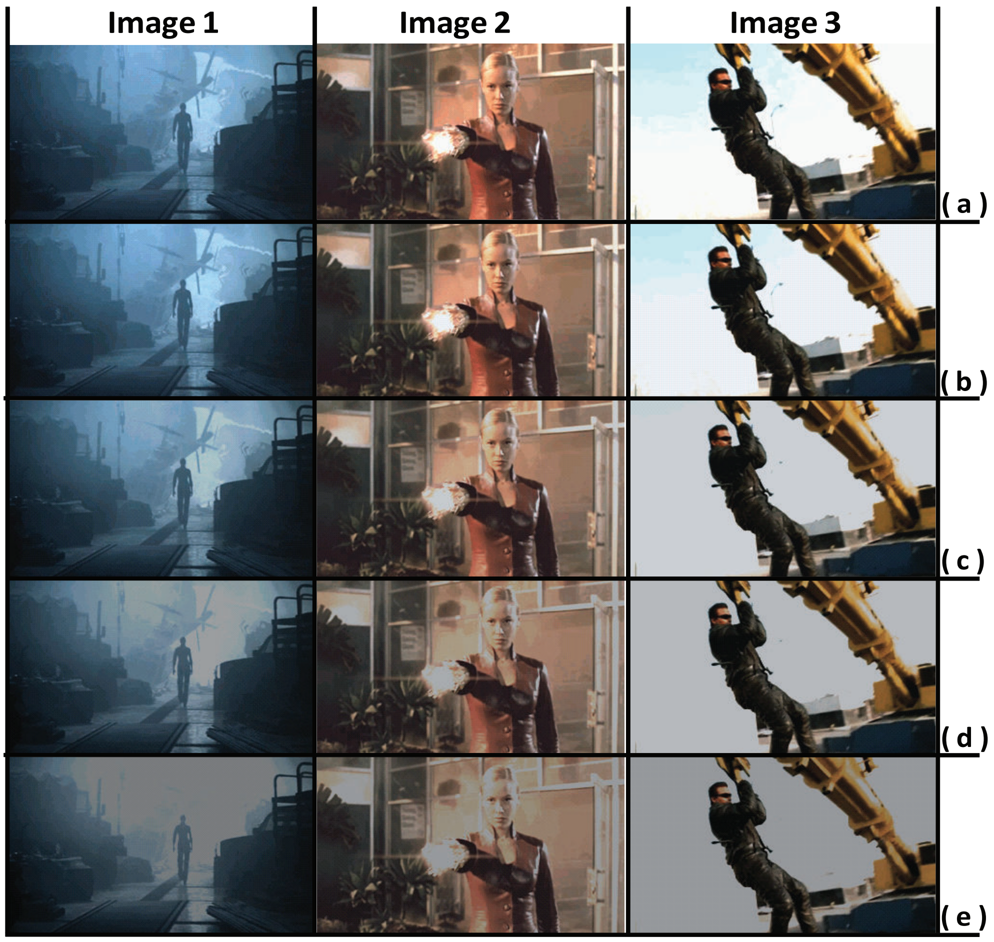


Figure 3.7: Images at different distortion levels: a) Original images; b) High quality (SSIM = 0.98); c) Medium quality (SSIM = 0.96); d) Low quality (SSIM = 0.94); e) Unacceptable (SSIM 0.90).

content and only related to α (see Section.3.3.2). As already show in Fig.3.1.b this is done in two steps:

- off-line: it consists of characterizing the relationship between different image quality loss and different compensation levels, for an image benchmark set by means of a HVS aware distortion metric. The relationship is obtained as statistical fitting function of the test results.
- on-line: the relationship in the off-line step is then used to univocally select the proper compensation value for the target tolerated distortion level in a not-content image aware fashion.

The main hypothesis behind this approach relies to the assumption that applying the same compensation factor to different images produces a constant visual quality loss. We validated and analysed this approach exploiting our framework. In order to do that, we applied the same compensation level to each frame of the video test benchmark chosen (i.e. Terminator 3 movie trailer).

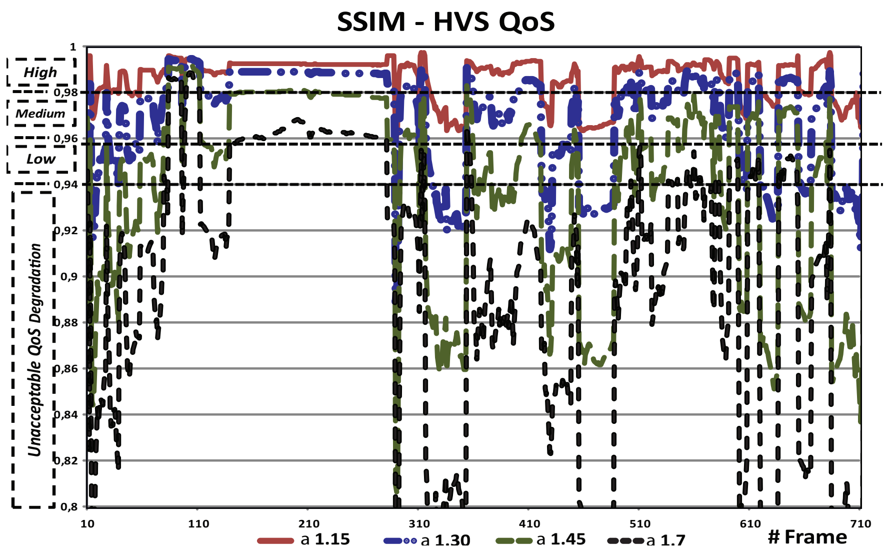


Figure 3.8: Image independent DBS technique: QoS framework output vs frame index.

Fig.3.8 shows the output SSIM index generated by the proposed QoS framework (y-axes) vs. the video frame index (x-axes). In the plot each line refers to a specific compensation level (α). Since each of these lines is not straight, but it crosses different quality regions, we can state that an invariant compensation level applied to every image does not lead to a constant perceived distortion level.

Image independent DBS techniques are incapable to keep the perceived QoS constant. By the way the plot shows that it is possible to achieve a constant distortion level by dynamically adapting the compensation level (α) frame by frame while considering the image spatial properties.

3.4.3 Image dependent DBS techniques

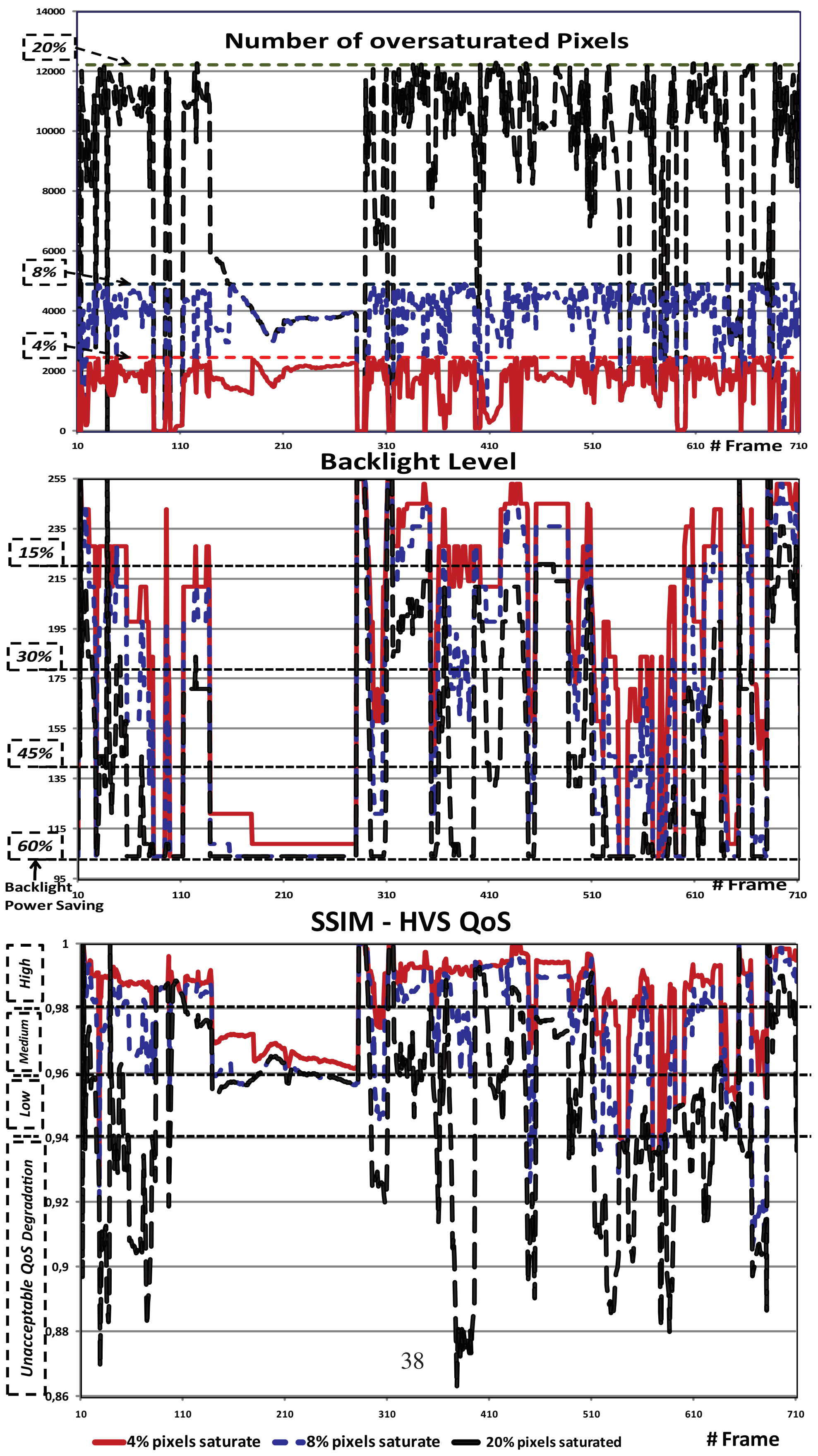
A different approach is to consider the distortion introduced by DBS proportional to image dependent key features, as shown in Fig.3.1.a . The most used technique considers the final perceptual degradation level proportional to the amount of oversaturate pixels on the compensated image (1) (2) (8). According to this approach, a constant perceived degradation level for different images can be obtained by keeping constant the saturated pixel percentage.

We use our framework to evaluate if this approach really lead to a constant perceived image degradation. For each frame of the video test benchmark we generate the R,G,B pixel value histograms. We scan them in order to find the maximum compensation factor (α_X) that keeps the percentage of pixel saturated below a selected level. Finally we select the most conservative α value between them. We then apply the compensation factor to evaluate the QoS SSIM index for the target images. In the plots in Fig.3.9 on the x-axes is represented the video frame index and on the y-axes are reported the saturated pixel generated by the DBS transformation, the backlight level and the SSIM QoS index associated for each frame. On each plot, each line refers to a different maximum level of saturated pixels, namely 4%, 8%, 20%.

We can see from the central plot that the backlight scales dynamically to keep the number of saturated pixels constantly below the specified levels for all the video frames, as reported in the top plot. But this does not lead to a constant visual distortion level, since in the bottom plot each line is not straight but cross different quality regions. This means that controlling the percentage of pixels saturated is not enough to maintain constant the perceived quality distortion. This bad behaviour is due to the evidence that the applied distortion level processing is not HVS aware.

3.4.4 Conclusions

In this chapter we proposed a framework to measure the perceived distortion for rendered images on LCD when a DBS technique is applied. We then apply this framework to analyse the QoS performance for the two most up to date DBS techniques. The first technique considers the perceive distortion level directly proportional to the dynamic range and not dependent to image content. We demonstrate that this approach is not capable to keep the perceived distortion level under control, because the image spatial information cannot be neglected. The second technique instead considers the perceived distortion level directly proportional to the number of saturated pixels and not related to the spatial distribution of them. We demonstrate that keeping constant the percentage of saturated pixels does not lead to produce constant perceived image distortion. Both these results suggest that in order to obtain a constant image degradation while saving the maximum allowed power, DBS solutions need to account the image pixels locality information and HVS peculiarities.


Number of oversaturated Pixels
14000
12000
10000
8000
6000
4000
2000
0
20%
8%
4%
10
110
210
310
410
510
610
710
# Frame
Backlight Level
255
235
215
195
175
155
135
115
95
15%
30%
45%
60%
Backlight
Power Saving
SSIM - HVS QoS
1
0,98
0,96
0,94
0,92
0,9
0,88
0,86
High
Medium
Low
Unacceptable QoS Degradation
4% pixels saturate
8% pixels saturate
20% pixels saturated

Bibliography

[1] Chang N., Choi I. and Shim H. DLS: dynamic backlight luminance scaling of liquid crystal display. In Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, ISSN:1063-8210, VOL.12, NO.8, PAGE.837–846 (2004) 26, 37

[2] Shim H., Chang N. and Pedram M. A Backlight Power Management Framework for Battery-Operated Multimedia Systems. In IEEE Des. Test, ISSN:0740-7475, VOL.21, NO.5, PAGE.388–396 (2004) 26, 37

[3] Iranli A. and Pedram M. DTM: dynamic tone mapping for backlight scaling. In Proceedings of the 42nd Annual Conference on Design Automation. DAC '05, 612-617. 2005. 26, 34

[4] Iranli A., Fatemi H. and Pedram M.. HEBS: Histogram Equalization for Backlight Scaling. Design, Automation and Test in Europe, 2005. Proceedings, pp. 346-351 Vol. 1, 2005. 26, 34

[5] Iranli A., Lee W. and Pedram M. Backlight dimming in power-aware mobile displays. In Proceedings of the 43rd Annual Conference on Design Automation. DAC '06. 2006. 26, 34

[6] Iranli A., Lee W. and Pedram M. HVS-Aware Dynamic Backlight Scaling in TFT-LCDs. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 14, N.10, 2006. 26, 34

[7] Cheng L., Bossi S., Mohapatra S., El Zarki M., Venkatasubramanian N. and Dutt N. Quality Adapted Backlight Scaling (QABS) for Video Streaming to Mobile Handheld Devices. citeseer.ist.psu.edu/722345.html 26

[8] Cheng W. and Chao C. Perception-guided power minimization for color sequential displays. In Proceedings of the 16th ACM Great Lakes Symposium on VLSI, 290-295, 2006. 26, 37

[9] Wang Z., Bovik A. C., Sheikh H. R. and Simoncelli E. P. Image quality assessment: From error visibility to structural similarity IEEE Trans. on Image Process., Vol. 13, no. 4, pp. 600-612, Apr. 2004. 26, 27, 33

[10] Ebner F. and Fairchild M. D. Development and testing of a color space (ipt) with improved hue uniformity 6th IST/SID Color Imaging Conference, pp 8-13, 1998. 34

[11] i.MX31: Multimedia Applications Processor http://www.freescale.com 27

[12] The SSIM Index for Image Quality Assessment http://www.ece.uwaterloo.ca/ z70wang/research/ssim/ 33

[13] Bonnier N., Schmitt F., Brettel H. and Berche S. Evaluation of Spatial Gamut Mapping Algorithms 14th IST/SID Color Imaging Conference, 2006. 34

[14] Findlay Shearer Power management in mobile devices Wiley ed.
26

Chapter 4

HVS-DBS: Human Visual System-aware Dynamic Luminance Backlight Scaling

4.1 Overview

As result of previous chapter 3 we demonstrate that none of the state-of-the-art DBS algorithms is really capable of ensuring a target image quality while achieving major power savings. In this chapter we propose a novel on-line technique for dynamic backlight scaling. The presented approach is HVS (i.e. Human Visual System) and image structure-aware. We also provide at the end of the chapter a fully operational implementation of the proposed framework by which we carried out a full characterization of the overall system power consumption versus QoS.

4.2 Introduction

As already described in Chapter 2 the main contribution to LCD subsystem power consumption comes from the backlight (1).

Previous Chapter 3 results highlight that most techniques rely on metrics that none of the DBS techniques presented in literature is really capable of ensuring a target image quality while achieving maximum power savings. In fact, most techniques rely on metrics that are not well-related to degradation perceived by human eyes to meet frame-rate imposed constraints (17), (12), (5), (10), (11). Advanced DBS solutions need to account for information on pixels locality and Human Visual System (HVS) characteristics to maintain a constant image degradation level - while saving the maximum possible power (6), (7), (8), (9). However, accounting on-line for information on pixels locality and HVS peculiarities is very energy expensive, due to the computational complexity of image processing and HVS model (16).

The main contribution of this chapter is the definition of a novel on-line technique for DBS which limits the degradation of the final rendered image in a robust fashion. The presented approach is HVS and image structure-aware. We provide a real implementation of the proposed framework on a Freescale application development board based on the i.MX31 processor. We carried out a full characterization of the overall system power consumption versus QoS. Experimental results validate the our technique and highlight the robustness of our approach in terms of performance, energy efficiency and image quality trade-offs.

The rest of the chapter is structured as follows: Section 4.3 describes the target embedded multimedia processor; in Section 4.4 the main DBS problem model is analytically formulated; Section 4.5 introduces our main HVS-DBS approach, while Section 4.6 describes its implementation; finally we present our experimental results in Section 4.7 and conclusions in Section 4.8.

4.3 i.MX31 Multimedia Processor

The Freescale i.MX31 is an ARM-11 based Application Processor targeted for portable multimedia devices (13). The i.MX31 processor is augmented with an Image Processing Unit (IPU), where some computationally-intensive parts of video processing chain are offloaded from the ARM CPU and accelerated in hardware.

The Image Processing Unit is designed to support video and graphics processing functions and to interface to video/still image sensors and displays. The IPU is equipped with a DMA engine to perform its tasks with minimal involvement of the ARM CPU control and synchronization. As a result, in most cases, the CPU involvement is focused on processing tasks such as video decoding. Moreover, the system-on-chip integration combined with internal synchronization, avoids unnecessary access to system memory, reducing the load on the memory bus and reducing further the power consumption.

The IPU performs also some processing-intensive image manipulations, such as:

- Downsizing with independent integer horizontal and vertical ratios;

- Resizing with independent fractional horizontal and vertical ratios;
- Color space conversion (YUV to RGB, RGB to YUV, YUV to different YUV);
- Combining a video plane with a graphics plane (blending of graphics on top of video plane);
- 90 degree rotation, up/down and left/right flipping of the image.

Implementing our framework, we paid great attention in optimizing all memory accesses, CPU and IPU utilizations. The communication between the CPU, main memory and IPU's sub-modules has been made through DMA transfers, while synchronization has been handled by interrupts.

4.4 DBS problem model

As described in previous chapter3 the emitted light from a pixel is function of both backlight and pixel transmittance values (16), (18), (4).

$$L(x) = f(BL) * g(x) \tag{4.1}$$

Intuitively, this equation shows that the luminance emitted by a pixel depends on functions *f()* of the backlight (BL) and *g()* of the pixel value itself, which sets its transmittance. These two functions are related to the LCD panel used, and generally are non-linear. We model these using the results of a real[1] TFT LCD display characterization showed in (16), that leads to the follow equation:

$$L_X(x) = (X)^{\gamma_X} * (Backlight)^{\theta}, X = \{R, G, B\} \tag{4.2}$$

If we want to save power, we need to dim the backlight and scale at the same time the pixel transmittance to re-equilibrate the target pixel luminance. If we dim the Backlight value of a scaling factor β, we need also to scale each RGB pixel component by a set of α factors to obtain the same target luminance:

$$L'_X(x) = (\alpha_X X)^{\gamma_X} * (Backlight/\beta)^{\theta}, X = \{R, G, B\} \tag{4.3}$$

From 4.2 and 4.3 we have:

$$\alpha_X{}^{\gamma_X} = \beta^{\theta}, X = \{R, G, B\} \tag{4.4}$$

[1]Sharp LQ035Q7DB02

The last equation shows that, for a given β backlight scaling factor, we need to compensate each component by a α_X[1] factor. Due to the bounded digital range [0..255] for each component, this operation is not able to compensate the luminance in the top end values, thus introducing a saturation effect. This can produce both a luminance dynamic range compression and a color distortion.

4.5 The HVS-DBS Approach

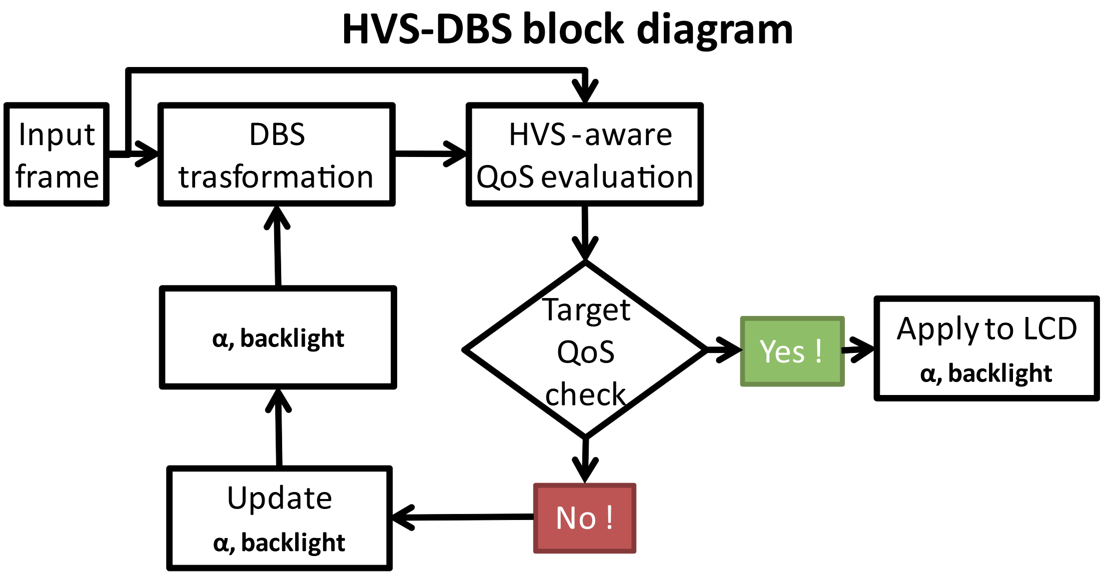


Figure 4.1: HVS-DBS Approach schematic.

As mentioned in (16) todays DBS approaches are incapable to keep the final image QoS under control: frame-dependent approaches do not use a HVS-aware on-line distortion metric, while frame-independent ones do not consider the image degradation function of the image itself. Our framework, showed in Figure 4.1, is frame-dependent and use a HVS-aware on-line distortion metric. It is based on a "try and test" approach to find the minimum backlight level that guaranties a goal QoS performance. It relies on the following steps:

1. The incoming frame pixels luminance is increased accordingly to Eq.4.3 ("DBS transformation" in figure), with a starting DBS parameter set equal to (α_{Xi}, $Backlight_i, i = 0$).

[1]For simplicity in the following sections we will use only α_B,and we will call it α.

2. It computes the distortion between the original frame and the DBS transformed one using the HVS-aware QoS metric(”‘HVS-aware QoS evaluation”’ in figure). In other word it predicts the human perceived distortion if the DBS parameter set $(\alpha_{Xi}, Backlight_i)$ will be applied to current frame.

3. The result of the previous step is checked (”Target QoS check” in figure) against the user defined QoS target threshold. If this value does not satisfy it (”No!” in figure), a new DBS parameter set $(\alpha_{X+1}, Backlight_{i+1})$ is selected and step 1 is triggered again. If instead the last measured distortion satisfies the goal QoS, the last DBS parameter set $(\alpha_{Xi}, Backlight_i)$ is applied respectively to the LCD input frame and the backlight driver.

The QoS control performance of the proposed framework strongly depends on the HVS-metric adopted, on the number of iteration that can be performed in a frame period and on the way we select the new $(\alpha_{X+1}, Backlight_{i+1})$ values. We choose to use an embedded version of the QoS framework presented in (16) as on-line metric. This will impact on the number of iterations, since it depends on the computational complexity of the single iteration step. As the QoS metric monotonously decreases with the aggressiveness of DBS transformation we obtain the new $(\alpha_{X+1}, Backlight_{i+1})$ by using a static, within frames, binary search tree. In this way each node of the tree contains a pair $(\alpha_X, Backlight)$. It must be noted that with this approach we bound the number of possible backlight values.

4.6 Framework Implementation

We provide a real implementation of the HVS-DBS framework. It has been embedded within a custom Video4Linux software subsystem running on a Freescale prototype development board host an i.MX31 multimedia application processor and a 3.3-inch QVGA Sharp display. The block diagram in Figure 4.2 describes how the entire framework operates.

We can divide the overall flow into two main sections:

1. frame-processing;

2. post-processing.

In the following sub-sections each phase will be described in depth.

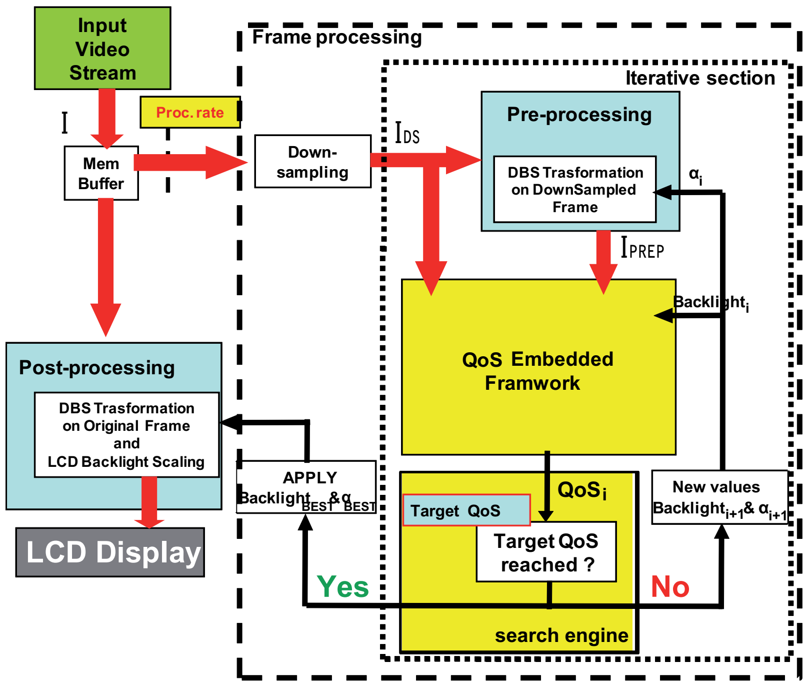


Figure 4.2: Framework implementation block diagram.

4.6.1 Frame-processing

The task of frame-processing is two fold: to evaluate in an HVS-aware fashion the final displayed frame distortion generated by a given set of DBS parameters $(\alpha_{Xi}, Backlight_i)$; to tune them keeping the final frame distortion below an user defined level. This must be done within a frame time budget. As it is reported on Figure 4.2, the frame-processing block relies on the following sub-steps:

1. Down-Sampling: it down-samples the input frame (I) reducing the computational complexity of the entire frame-processing stage.

2. Pre-Processing: it transforms the input down-sampled frame (I_{DS}) , to increase its pixels luminance.

3. QoS embedded framework: it takes as input the original frame (I_{DS}) and the one coming from the pre-processing (I_{PREP}). It generates as output a HVS-aware QoS score (QoS_i).

4. Search engine: this block receives as input the current QoS score (QoS_i) and checks it against the user specified QoS threshold. If the QoS target is not reached, it updates the

DBS parameters $(\alpha, Backlight)$ used in previous blocks and trigger block 2; The iteration path is called *iterative section* in Figure 4.2.

4.6.1.1 Pre-Processing

The pre-processing takes as input the down-sampled frame (I_{DS}) to increase the overall frame luminosity. This operation is done scaling up of a factor $\alpha_{R,G,B}$ all R,G,B pixels in the input image. It produce a frame with increased luminance (I_{PREP}). Implementing it on CPU should result in a very power expensive step. We implemented it in hardware exploiting in a smart way the capabilities of the available IPU. One of the main functions performed by the hardware IPU is colour space conversion which is done through conversion matrices. The conversion matrix coefficients are programmable and they are stored in the IPU main memory region.

The conversion equations are:

$$Z0 = 2^{scale-1} * (X0 * C00 + X1 * C01 + X2 * C02 + A0) \tag{4.5}$$

$$Z1 = 2^{scale-1} * (X0 * C10 + X1 * C11 + X2 * C12 + A1) \tag{4.6}$$

$$Z2 = 2^{scale-1} * (X0 * C20 + X1 * C21 + X2 * C22 + A2) \tag{4.7}$$

Where X0, X1 and X2 are the component of the input format; Z0, Z1, Z2 the component of the output format; C00, .., C22 and A0, .., A2 the conversion matrix coefficients.

For example, if input image format is RGB, the default conversion matrix is the identity matrix:

$$CSC_{default} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

In order to compensate an image after backlight scaling down, we need to scale up the value of $CSC_{default}$ by an $\alpha_{R,G,B}$ factor:

$$CSC_{DBS} = \begin{vmatrix} \alpha_R & 0 & 0 \\ 0 & \alpha_G & 0 \\ 0 & 0 & \alpha_B \end{vmatrix}$$

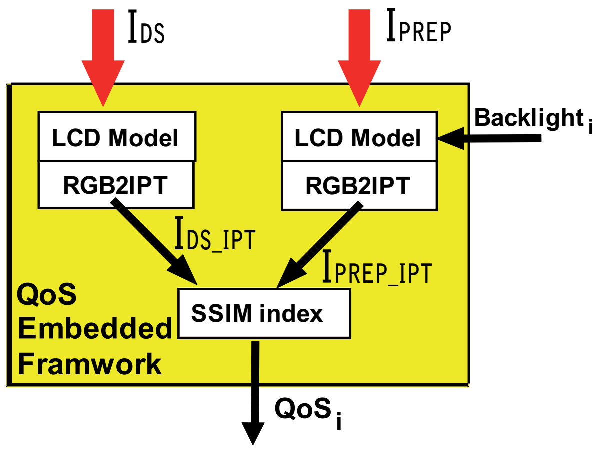


Figure 4.3: QoS embedded framework block diagram

4.6.1.2 QoS embedded framework

This block implements an embedded version of the QoS framework presented in (16). Figure 4.3 shows the block diagram of it. We can identify three main blocks:

1. The LCD model: this block models the behaviour of a real embedded LCD panel and its non-idealities. From the RGB input image and the backlight level, it generates an image representing how the image itself will look like on the LCD panel. This is done for both I_{DS} and I_{PREP} images.

2. RGB2IPT: this block transforms the LCD Model output images from RGB to IPT colour space to account human eyes different sensibilities for different colors. In IPT each quantity is correlated to the perceived lightness, chroma and hue (14).

3. SSIM index routine: this block models the HVS peculiarities and evaluates the differences between the two images from the LCD model block in IPT color space I_{DS_IPT} and I_{PREP_IPT}. Indeed SSIM index metric has been demonstrated to be able to quantify the HVS perceived differences between two images. It is based on the assumption that the human visual system is efficient in extracting structural information from the viewing field. The structural information of an image is an attribute which describes the structure of objects in the scene independently from average luminance and contrast (3).

Since all these steps are very computational expensive and need to be iterated, they represent the bottleneck and the challenge of our solution. We implemented it in a optimized and

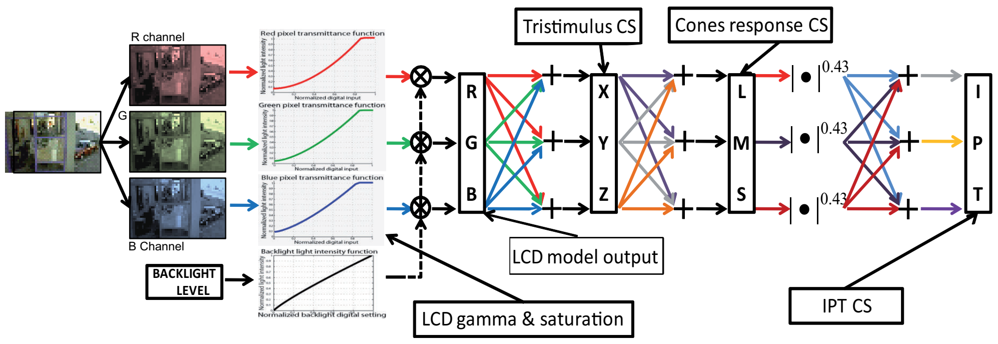

R channel
G
B Channel
BACKLIGHT LEVEL
Red pixel transmittance function
Green pixel transmittance function
Blue pixel transmittance function
Backlight light intensity function
Normalized backlight digital setting
Tristimulus CS
Cones response CS
R
G
B
X
Y
Z
L
M
S
I
P
T
0.43
LCD model output
LCD gamma & saturation
IPT CS

Figure 4.4: LCD model and IPT color space conversion schematics

efficient way as described in the following sections.

4.6.1.3 LCD model and RGB2IPT

Figure 4.4 shows the internal block diagram of the LCD model and RGB2IPT blocks. It is made up by the following steps:

- The input R,G,B image pixels and input backlight values are first non-linearly transformed to model the LCD non-idealities (LCD gamma & saturation) (16) and then combined together to produce an RGB image, where each pixel is proportional to the light emitted by the LCD modeled.
- The LCD model output is converted before to XYZ color space and than to LMS color space with two linear transformation. The last color space is related to the human eyes cone sensitivity and is suitable for detecting color adaptation artifacts.(15)
- The absolute value of the LMS image is then extracted and powered by a constant factor. Afterwards it is linearly transformed to the IPT color space.

The described image manipulation consists in a series of both linear and nonlinear transformations of the input pixels color components and the backlight level. It cannot be implemented in an energy efficient way by using only CPU ALU operations.

Since the input frame is represented in RGB565 format each pixel is coded with 16 bits. We implemented all the transformation steps with a set of look up tables (LuT), one for each backlight level. We need to store few LuTs, only the number of nodes composing the searching tree(no other backlight values will be used). We select the LuT corresponding to the backlight level that we want to simulate. The LuT is addressed by the RGB565 pixel value, a 16bit number, and provides as output the IPT representation, coded with 8 bits for each component.

4.6.1.4 SSIM index routine

Figure 4.5 shows the internal block diagram of the SSIM index routine. Its behavioral can be summarized by the following steps:

1. The two input images Im_O and Im_{DBS} are powered by 2 (${Im_O}^2$ and ${Im_{DBS}}^2$) and cross-multiplied.
2. Then Im_O, Im_{DBS},${Im_O}^2$,${Im_{DBS}}^2$) and the cross-multiplied are 2D filtered with a gaussian window. This operation locally averages the input signals.
3. The results of this operation are combined to produce statistical quantities on the input images. In fact it produces the averages (μ_O, μ_{DBS}), the variances (${\sigma_O}^2$, ${\sigma_{DBS}}^2$) and the cross-variance σ_{O-DBS}
4. Finally all these components are combined to produce a structural similarity error map.
5. The error map contains the spatial distribution of the error, and needs to be averaged producing a unique distortion index.

The computational cost of each point in the error map is dominated by the cost of the 2D filter routines. To let these routines to be computed in a real embedded processor, within a frame time budget, we needed to reduce the size of the input images. This is performed by the down sizing block(already see Section 4.6.1). By scaling down the input image we needed to carefully reduce the size of the filter window. Indeed the gaussian window must be scaled down according with the image itself to preserve the same locality information and do not lose the sensibility to the structural changes.

We decided to implement this routine with fixed point arithmetics to improve performance.

Since the DBS transformation mainly involves the luminance component (I in IPT domain) we can assume that the distortion introduced will affect mainly this component.

The Figure 4.6 shows that for a given image distortion the SSIM index computed considering the IPT components can be approximated by the SSIM computed only on the I component. This approximation decreases the size of the LCD model LuTs, improving its performance. It even reduces the SSIM routine computational cost, since neglects the computation of the SSIM

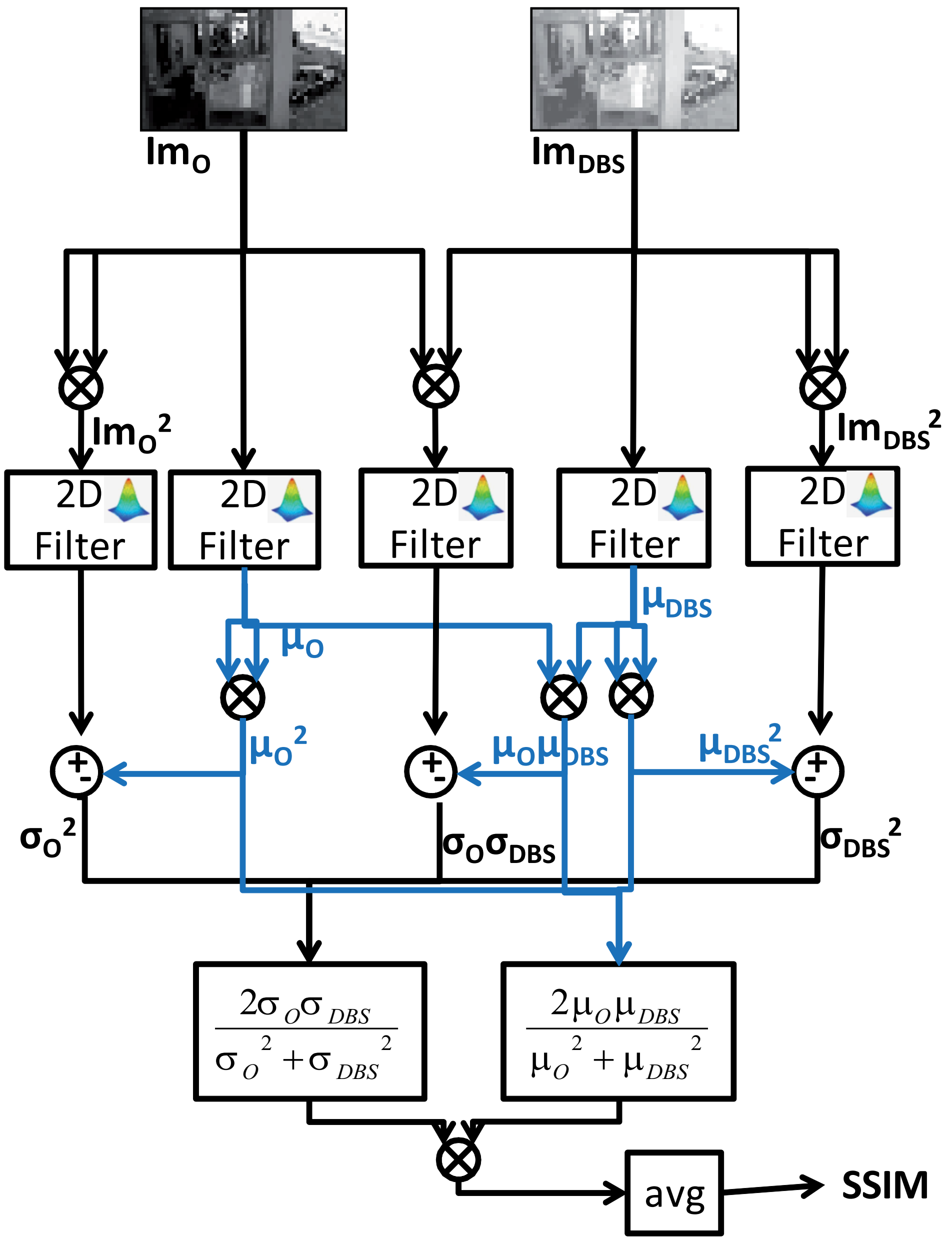


Figure 4.5: SSIM index routine schematics

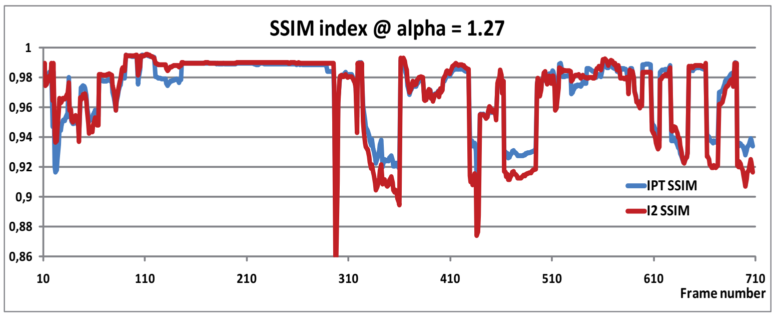


Figure 4.6: SSIM IPT vs SSIM I2 comparison.

for the P, and T colour components. Finally the SSIM error map has to be averaged to produce an unique index, we approximate it by computing the average to a down-sampled version of the error map. In this way we can reduce significantly the number of points in the error map that have to be computed.

4.6.1.5 Search Engine

This block receives as input the current SSIM index scores and checks the condition $SSIM_i > SSIM_{TARGET}$. According to that it explores the binary search tree to compute the future (α_{Xi+1}, $Backlight_{i+1}$) to be applied in next iteration. We limited the tree search dept due to real time constraints and performance trade-offs performing a characterization study. Based on previous results we choose to perform 3 iterations that allow us to explore up to 9 levels of backlight.

4.6.2 Post-processing

The final step of our framework is the *post-processing* phase. The main task of this section is to apply the RGB pixel scaling factor $\alpha_{R,G,B}$ found in the frame-processing phase to both backlight LCD and input video frame. The input frame is the starting image of the frame-processing phase with its original resolution.As previously described (see Section.4.6.1.1) this highly computational expensive pixel transformation, can be done in hardware by the colour space conversion module.

4.7 Experimental Results

We measured the QoS and power consumption of the LCD, CPU, and main memory during the playing of a video sequence from *Terminator 3* movie. Table4.1 shows the main characteristics of this video stream in terms of frame rate and resolution.

Video	Frame Rate	Frame Resolution
Terminator 3	24 fps	352x192

Table 4.1: Terminator 3 video stream features.

The selected video sequence is composed by two main sections with different frame characteristics. The fist sequence contains indoor scenes, which are characterized by the predominance of dark pixels; the second one is composed by outdoor scenes, with frames showing mainly bright images.

We evaluated the performance of the proposed HVS-DBS implementation using the methodology for visual quality fidelity presented in (16). This performance metric output has been showed to be proportional to human perceived degradation. Table 4.2 shows the correlation between the score intervals and QoS macro-regions.

SSIM Range	Quality Level
1 - 0.98	High quality
0.98 - 0.96	Medium quality
0.96 - 0.94	Low quality
$\leq$ 0.94	Unacceptable

Table 4.2: QoS framework output quality regions.

Using such calibration, we are able to measure the final quality of the video stream and the power savings.

Several experimental evaluations have been carried out on our framework. Firstly, we assessed the correctness of all our assumptions and approximations, then we compared our HVS-aware framework with another histogram-based solution. We carried out a characterization of the final rendered quality against different QoS goals (named $SSIM_{TARGET}$ in our implementation). This test has been performed for the input test video stream.

Table 4.3 shows the framework power overhead for CPU, peripheral and main memory. It is referenced to the original video4linux driver, while running the test video. Instead, in the last row we report the time required by the framework to complete the Frame-processing stage. This time is smaller than the frame time budget. Thus our framework does not degrade video decode and playback fluidity. As we will see later in this section even if the overhead cannot be

CPU [%]	11%
Peripheral [%]	1%
Main memory	13%
Δ Frame-processing	17,3ms

Table 4.3: HVS-DBS implementation power overhead.

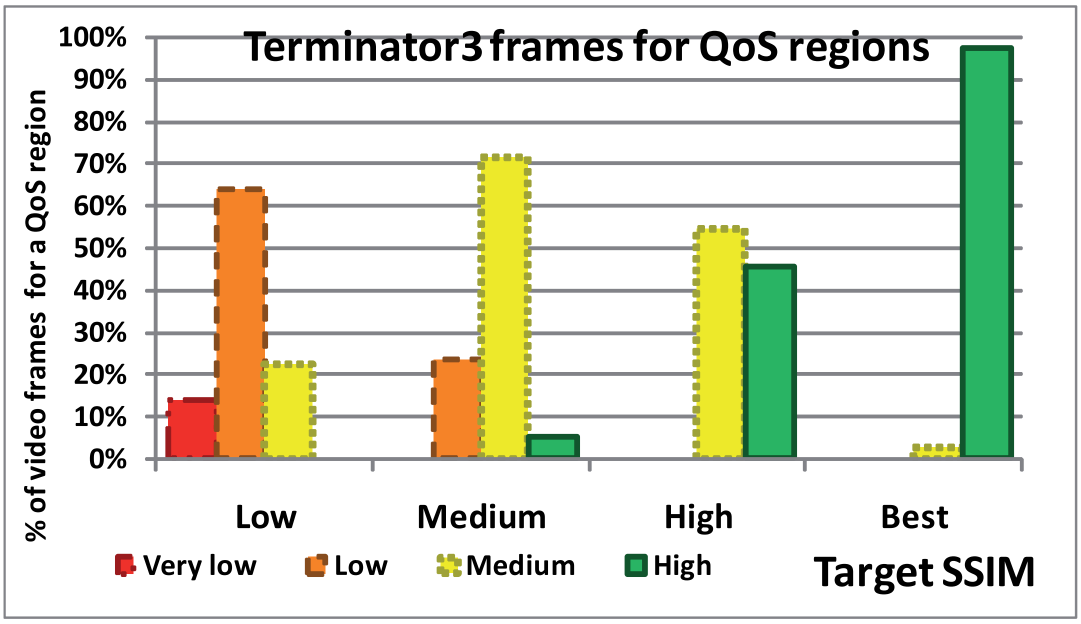


Figure 4.7: Terminator 3: QoS frame distribution for different target quality levels.

neglected, the savings on the LCD backlight allow our approach to achieve significant savings on the entire system.

Figure 4.7 shows on y-axes the QoS frame distribution for four different target quality levels (in x-axes). Moving from a high quality to low target qualities the highest number of video frames falls in the proper region. Only a small percentage of frames falls in the lower neighborhood QoS region. This demonstrates that our framework is able to keep visual QoS under control.

Figure 4.8 shows on y-axes the entire system (CPU, Peripherals, Memories and LCD) power consumption savings for four different target quality levels (in x-axes). The bars correspond to the several sequences of the video, namely total, indoor and outdoor. It can be noticed that power savings strongly depend on the target QoS level. If user requires a high quality output video stream, our framework is able to save almost 5% of the total energy budget. However

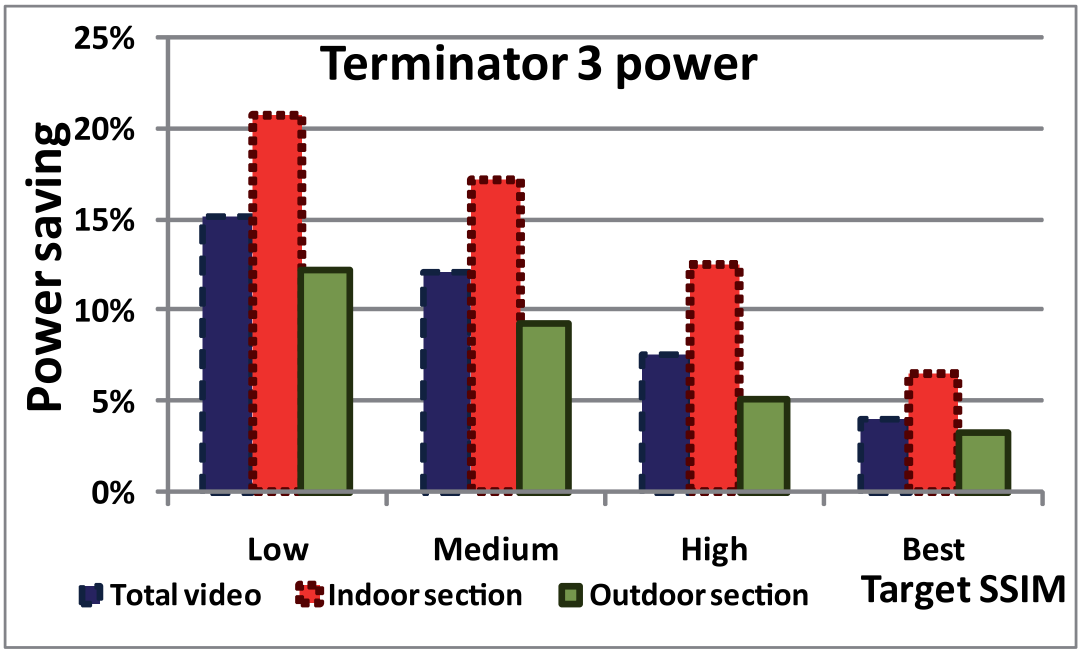


Figure 4.8: Terminator 3: Power saving breakdown for different video sequence under different quality levels.

power savings strongly depends on input video frames: lighter frames do not allow a more aggressive backlight scaling for a given target quality of service. This is indeed demonstrated by the gap between indoor and outdoor bars for the same target SSIM in the figure. If user allows a higher distortion in favour of lower system energy consumption, our HVS-DBS techniques increases power savings up to 21%.

We compared the performance and the power savings obtained using our novel HVS-DBS framework with a histogram-based approach. The histogram-based technique first builds a luminance histogram for every input frame, and then analyses it computing the optimal scaling factors. The histogram-based approach selects the proper scaling factors calculating the number of pixels which will saturate and introduce distortion. The distortion threshold is represented by the number of saturated pixels admitted in the image. A low distortion threshold corresponds to high quality video output and low power savings. A high distortion threshold will bring to high power savings at the cost of low quality video output. The Figure 4.9 shows the performance results of the histogram-based technique. The graph illustrates the percentage of frames which can be classified in a QoS region, varying the distortion threshold. Figure 4.10 shows the power savings for the different sequences of the target video stream considering

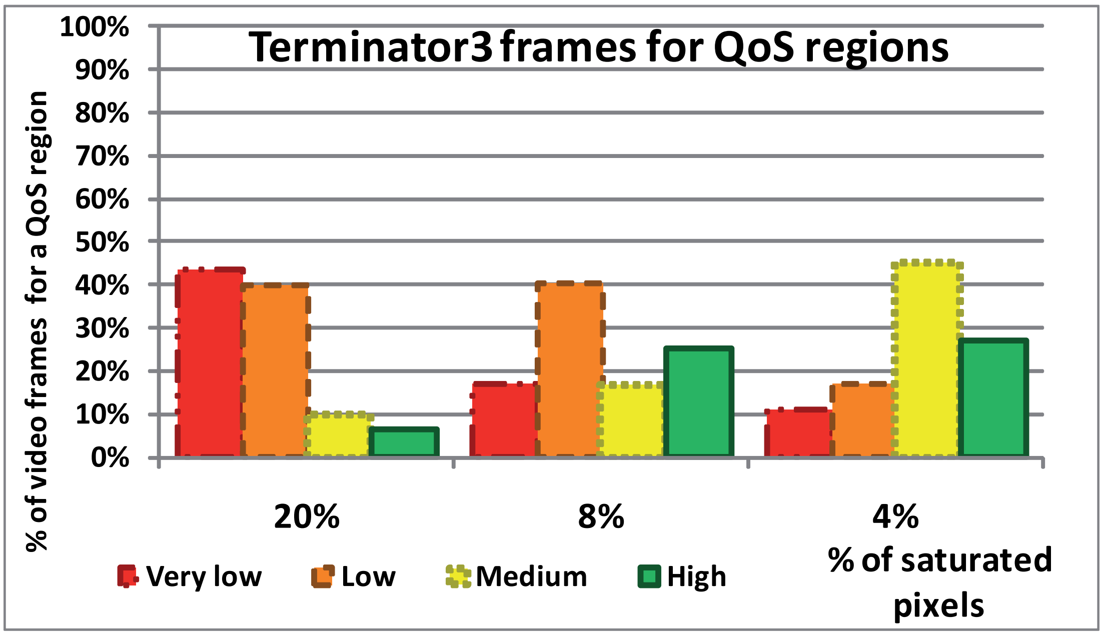


Figure 4.9: Terminator 3: QoS frame distribution for different target quality levels using the histogram-based approach.

several distortion thresholds.

The comparison between the two approaches brings to several considerations. HVS-DBS provides a better high-quality frame distribution for every QoS level. The histogram-based algorithm distorts a lot of frames (the ”Very Low” bars in the graph) also for 4% and 8% thresholds. The HVS-DBS instead produces some distorted frame only in the ”Low QoS” scenario, proving almost the same amount of power savings with regards to the histogram-based. Moving forward higher quality requirements, the HVS-DBS approach is always able to find the optimal trade-off between admitted frame distortion and power savings, thanks to its capability of being frame- and human-visual-system aware. The histogram-based instead does not scale its power savings accordingly the QoS requirements: it remains too aggressive thus distorting several frames. The HVS-DBS framework shows a higher capability of controlling the required QoS while optimizing the power consumption.

4.8 Conclusions

State-of-the-art DBS techniques cannot precisely produce the required QoS over all the displayed frames of a video. The main reasons are represented by the evidence that frame-

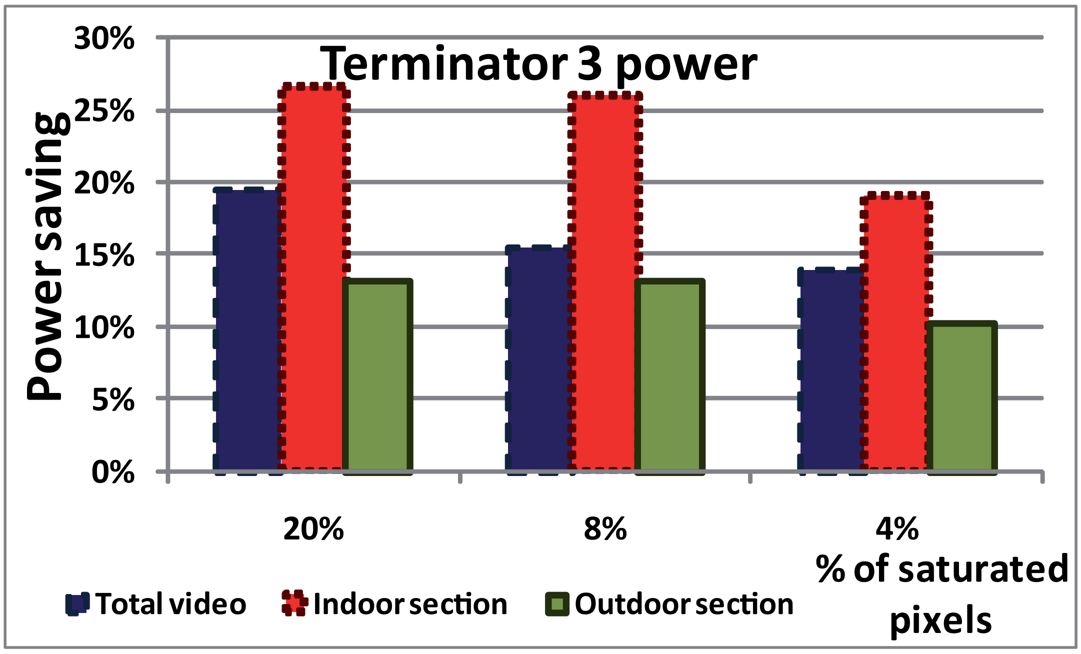


Figure 4.10: Power saving breakdown for different video sequence under different percentage of pixels saturated using the histogram-based approach.

dependent approaches do not use a HVS-aware on-line distortion metric, while frame-independent ones do not consider the image degradation dependent of the image itself. We proposed a novel dynamic backlight scaling approach overcoming these limitations. Our solution leverages on an on-line HVS-aware metric finding out the optimal backlight level for a specified QoS. We showed how to reduce the complexity of the adopted metric, providing a real implementation of the proposed framework on a multimedia embedded processor. Experimental results confirmed the high performance of our approach in terms of both power and QoS. Our HVS-DBS allows finding the optimal trade-off between QoS and power savings.

Bibliography

[1] Findlay Shearer. Power management in mobile devices. Elsevier LTD, Oxford. 41

[2] www.lcdtvassociation.org. www.lcdtvassociation.org/images/TV_Power_Consumption _White_Paper_LCD_TV_Association.pdf

[3] The SSIM Index for Image Quality Assessment http://www.ece.uwaterloo.ca/ z70wang/research/ssim/ 48

[4] Kerofsky L. abd Daly S. Brightness Preservation for LCD Backlight Dimming. In Sharp Technical Journal, ISSN:0285-0362, NO.95, PAGE.50–57 (2007) 43

[5] Shim H., Chang N. and Pedram M. A Backlight Power Management Framework for Battery-Operated Multimedia Systems. In IEEE Des. Test, ISSN:0740-7475, VOL.21, NO.5, PAGE.388–396 (2004) 41

[6] Chang N., Choi I. and Shim H. DLS: dynamic backlight luminance scaling of liquid crystal display. In Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, ISSN:1063-8210, VOL.12, NO.8, PAGE.837–846 (2004) 42

[7] Cheng W. and Chao C. 42
Perception-guided power minimization for color sequential displays. In Proceedings of the 16th ACM Great Lakes Symposium on VLSI, 290-295, 2006.

[8] Cheng W., Hsu C. and Chao C. Temporal vision-guided energy minimization for portable displays. In Proceedings of the 2006 international Symposium on Low Power Electronics and Design, ISLPED '06. 89-94, 2006. 42

[9] Cheng W., Hou Y. and Pedram M. Power Minimization in a Backlit TFT-LCD Display by Concurrent Brightness and Contrast Scaling. In Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1. 2004. 42

[10] Cheng L., Bossi S., Mohapatra S., El Zarki M., Venkatasubramanian N. and Dutt N. Quality Adapted Backlight Scaling (QABS) for Video Streaming to Mobile Handheld Devices. citeseer.ist.psu.edu/722345.html 41

[11] Cheng L., Mohapatra S., El Zarki M., Dutt N. and Venkatasubramanian N. Quality-based backlight optimization for video playback on handheld devices. Adv. MultiMedia 2007, 1 (Jan. 2007), 4-4. 41

[12] Iranli A., Fatemi H. and Pedram M.. HEBS: Histogram Equalization for Backlight Scaling. Design, Automation and Test in Europe, 2005. Proceedings, pp. 346-351 Vol. 1, 2005. 41

[13] i.MX31: Multimedia Applications Processor http://www.freescale.com 42

[14] F. Ebner and M. D. Fairchild Development and testing of a color space (ipt) with improved hue uniformity 6th IST/SID Color Imaging Conference, pp. 8-13, 1998. 48

[15] Fairchild M. D. Color appearance models Wiley ed. 49

[16] Bartolini A., Ruggiero M. and Benini L. Visual quality analysis for dynamic backlight scaling in LCD systems. Design, Automation and Test in Europe, 2009. Proceedings, 2009. 42, 43, 44, 45, 48, 49, 53

[17] Ruggiero M., Bartolini A. and Benini L. DBS4video: dynamic luminance backlight scaling based on multi-histogram frame characterization for video streaming application. In Proceedings of the 8th ACM EMSOFT, Atlanta, GA, USA, pp 109-118, 2008. 41

[18] Iranli A., Lee W. and Pedram M. HVS-Aware Dynamic Backlight Scaling in TFT-LCDs. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 14, N.10, 2006. 43

Chapter 5

Power management techniques for High Performance Computing

The second part of this thesis is focused on exploring and developing dynamic resource management policies for high performance computing and server platforms. The main objectives of of these techniques are to adapt at run-time the resource usage to achieve an optimal trade-off between performance, QoS, power consumption and cooling effort, while at the same time ensuring safe working temperature under all working conditions. Todays multicore processors include hardware support for dynamic power and thermal management, based on introspective monitors, sensors and performance knobs. This infrastructure provides the sensors and the actuators for feedback control management policies.

Fig. 5.1 shows an overview of the blocks needed for resource management. Based on output from a set of distributed sensors, such as performance monitors (PMU) and temperature sensors, the controller decides on the frequency and voltage levels (DVFS) for each core. If it is convenient it decides to power down the entire core (by power gating in modern cores) to both limit the power consumption and the temperature of each core .

The goal of this chapter is to explore and design a resource management infrastructure for the Linux O.S. In Section 5.1 the state-of-the art for this kind of techniques is analyzed and discussed. Section 5.2 describes the basics of our framework, named Per Task Sensor (XTS), developed to read the distributed performance counters and temperature sensors within the Linux kernel. It also represents a framework for the implementation of the dynamic resource management policy. Section 5.3 contains the results of a benchmarking stage carried out to highlight the potential savings and the corner cases in a real use case. Section 5.4 describes a

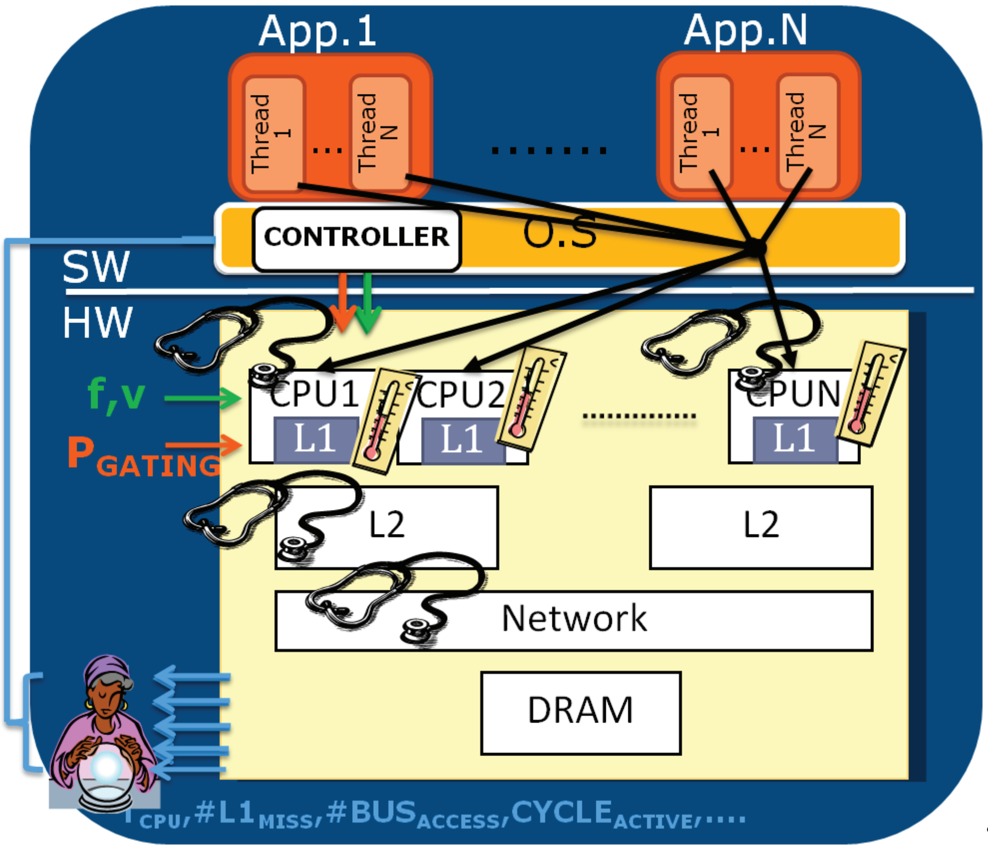


Figure 5.1: Building blocks for resource management

methodology to model the power consumption of a multi-core platform based on ”at the wall” real power measurements.

5.1 Power and Thermal Management Overview

Pushed by the market request of constant performance improvement across newer products, current and future processors have become more and more parallel, high-performing, extremely complex, and dense with high power consumption. Due to physical limitations this leads to complex and expensive thermal dissipation solutions. Since processors reveal a cost penalty due to power dissipation when they are not used optimally, dynamic power and thermal management techniques try to adapt system power and performance to required levels. Moreover, thermal techniques reduce performance to ensure a ”healthy” CPU working temperature. However, changing the system performance does not come for free, but requires both power and delay overhead. This leads to a complex cost-benefit problem where the global benefit depends on future performance requirements. Since the program flow is usually repetitive and locally steady it can be split into program phases. The challenge lies in accurately predicting the future behaviour in order to drive the dynamic power and thermal management decisions. The complexity of the prevision depends on the system considered. In fact the same task can have different run-time flow depending on other tasks running on the same CPU or in other CPUs that share resources. These approaches are classified as proactive and are opposite to reactive approaches, in which no assumption of the future behaviour is made. The following subsection reports different approaches to deal with phase prediction and DVFS (Dynamic Voltage and Frequency Scaling) power reductions. The first two papers show different solutions for a multitasking environment. The first ((1)) uses a hierarchical predictor structure, whereas the second ((2)) uses an on-line learning predictor. Differently from these approaches, the third ((3)) one shows how to consider the memory for energy minimization. The last two papers show how to use this information in a multicore environment. Section (4) shows a distributed approach in which each CPU selects its own operating point according to a global policy. The last one ((5)) uses a centralized controller instead to select the optimum frequency of each core to maximize the performance by optimizing the memory bus fairness. Section 5.1.2 shows how to use DVFS as knobs to constrain the working temperature of the CPU. The first paper ((6)) shows the advantage of using controller theory against heuristic approaches. The second one ((7)) shows a proactive thermal management approach based on future temperature prediction.

Finally the third one ((8)) shows how to deal with multiprocessor systems by showing the benefit of a distributed control system against a centralized one.

5.1.1 Phase prediction for DVFS power reduction

5.1.1.1 Vertigo : Automatic Performance-Setting for Linux (1)

Is this paper Vertigo is presented, a DVFS Linux framework targeted to single processor, embedded / laptop applications. It relies on a set of predictors organized in a hierarchical way to obtain the task idleness. This value is then used to select the optimum voltage and frequency. It is implemented as a set of kernel modules and patches that hook into the Linux kernel to monitor program execution and to control the speed and voltage levels of the processor. It is tested on a Transmeta Crusoe processor. As mentioned, the applied performance level is determined by the prediction of the future workload and its deadline. Two relative simple per-task hierarchic predictors are used to cover different application scenarios. The lower predictor covers the majority of the tasks and it is based on the following three steps:

- It computes the full-speed equivalent work done (Workfse) in the last expired scheduler interval. It is as the sum of the times spent at each performance level weighted by the performance level. It estimates the execution time of the task as if it was running at the maximum frequency allowed by the processor. It uses an exponential decay average with the previous predicted value (WorkEstold) to predict the future workload requirement (WorkEstnew) .

- It predicts the future deadline (Deadlinenew) as exponential decay average between the previous estimated value (Deadlineold) and the last expired interval probed value. Fig. 6 shows how to estimate the task duration. It can be done by counting the time delay between when the task starts its execution and when it gives up and releases the CPU. The deadline is the time elapsed between when the task starts and its following activation.

- It computes the future performance setting (Perf) as the ratio between the future full-speed equivalent work done and the future deadline.

On the top of this low-level prediction resides an interactive workload predictor. In this case the deadline is empirically defined as the human perception threshold equal to 50ms. It is obtained by accounting all the tasks that are awoken by a single GUI event. It computes a global Workfse and by dividing this value by the human perception threshold it obtains the future

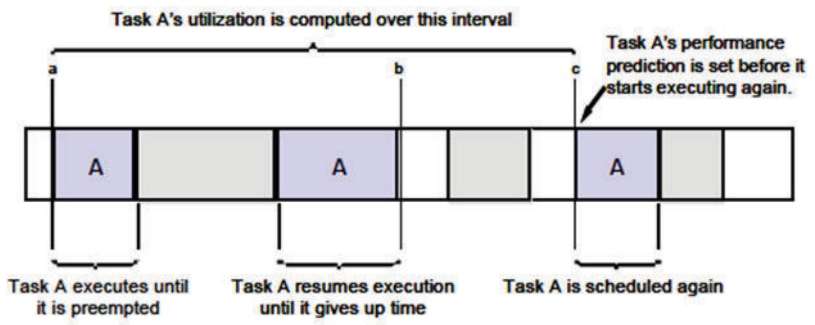

Task A's utilization is computed over this interval
a
b
c
Task A's performance prediction is set before it starts executing again.
A
A
A
Task A executes until it is preempted
Task A resumes execution until it gives up time
Task A is scheduled again

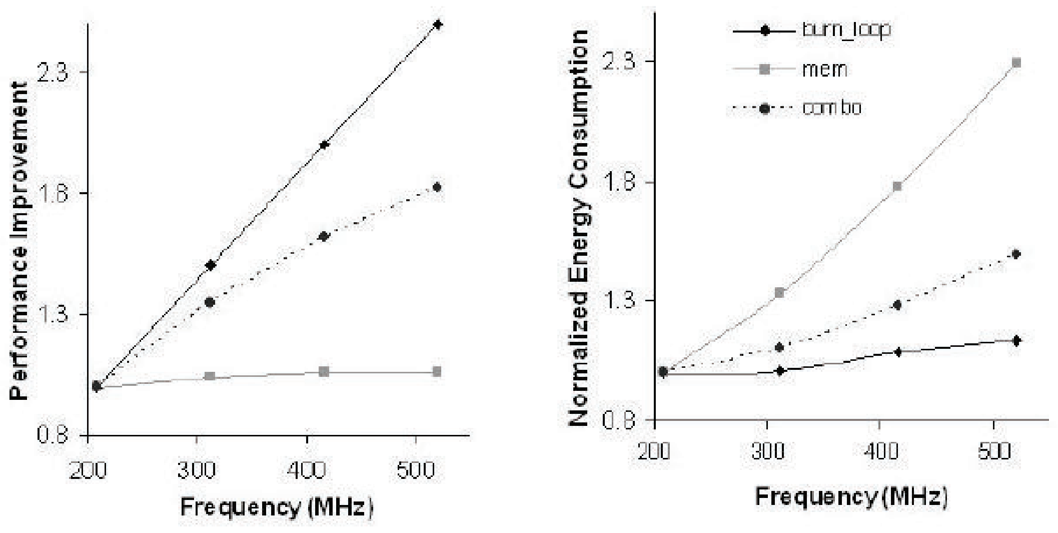
Figure 5.2: Measuring the utilization for task A

performance setting (Perf) . If the duration of the episode is bigger than a ”skip threshold” the predicted interactive performance level is applied. Otherwise it is not taken into consideration because it is too short. Instead, if the interactive episode is longer than a ”panic threshold”, the maximum performance is applied to minimize possibly recognizable performance loss. To avoid future performance loss if the panic threshold is reached the last execution time is weighted more in the exponential prevision. This biases the performance -energy saving trade-off toward performance.

In summary, this DVFS technique is able to predict future task idleness and expand the task duration in order to save energy. As we will see later, since it does not consider the portion of Workfse spent for memory accesses it does not exploit deeper savings. It is not optimal since it does not take possible energy consumption increases in the memory subsystem into account. It shows a hierarchical and modular approach that allows dealing with complex scenarios while keeping the single predictor relatively simple. However, this approach is not directly applicable to server systems since both GUI event and human perception thresholds are not easily definable in server systems.

5.1.1.2 DVFS for Multi-tasking Systems Using Online Learning (2)

This paper presents a Linux DVFS framework targeted to single processors and multitasking environments. It relies on an on-line learning approach to predict the future CPI (clock cycles per instruction) of each task. Since the CPI is a good metrics to quantify the memory bounded degree for a task, and consequently estimate the effect of frequency scaling on performance and energy, it is used to select the future processor performance level (voltage and frequency). Fig. 5.3 shows the frequency impact on performance (a) and energy (b) for three different benchmarks: burn_loop, mem, and combo. The first one is CPU bounded, the second one is memory bounded and the third one is a combination of the first two.

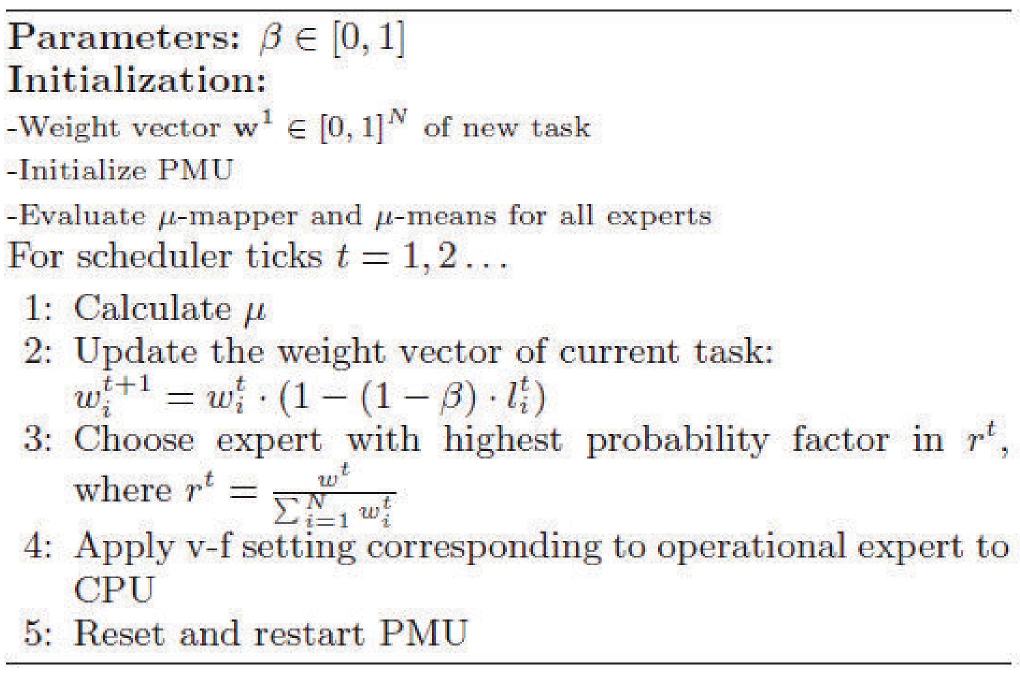


(a) Performance (b) Energy

Figure 5.3: Performance improvement and normalized energy consumption

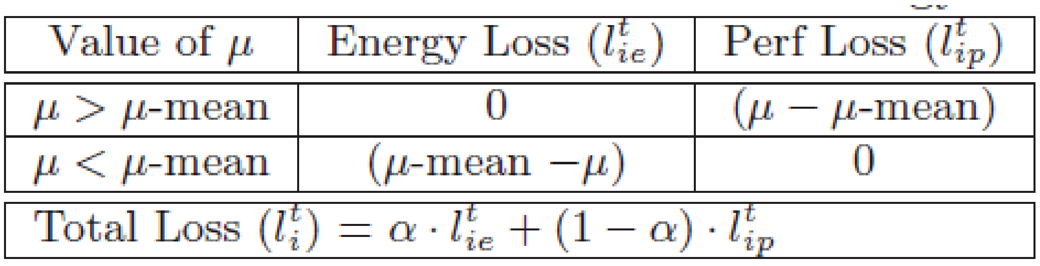
It can be noticed that CPU bounded tasks have benefits in terms of both performance and energy when running at higher frequency. Memory bounded tasks instead should run at the lowest frequency in order to maximize the energy saving without significantly impacting the performance. At run-time, first the current CPI is obtained by monitoring and combining the number of instructions executed (INST), data cache misses (DCACHE), number of cycles for which the instruction cache could not deliver instructions (ICACHE), number of cycles during which the processor is stalled due to data dependencies (STALL) and the total number of clock cycles (CCNT) elapsed since the PMU was started. Second the monitor data outputs are used together to derive the optimum performance setting ().

Based on the current performance setting () the framework updates the probability that a specific performance level will be right in the following interval. This is done by using an on-line learning approach made up of a set of experts. Each one is related to a particular performance level (voltage and frequency) and has an associated weight that represents its probability to be right. Every time the scheduler is called, the framework computes a new . According to that it updates the new weight for each expert, following the algorithm shown in Fig. 5.5. This update step requires estimating the total loss (lit) (composition of energy and performance loss) if this expert was used, see Fig. 5.4. The α parameter is used for tuning the performance vs. power saving trade-off.



```
Parameters: $\beta \in [0,1]$
Initialization:
-Weight vector $\mathbf{w}^1 \in [0,1]^N$ of new task
-Initialize PMU
-Evaluate $\mu$-mapper and $\mu$-means for all experts
For scheduler ticks $t = 1, 2 \ldots$
  1: Calculate $\mu$
  2: Update the weight vector of current task:
     $w_i^{t+1} = w_i^t \cdot (1 - (1 - \beta) \cdot l_i^t)$
  3: Choose expert with highest probability factor in $r^t$,
     where $r^t = \frac{w^t}{\sum_{i=1}^N w_i^t}$
  4: Apply v-f setting corresponding to operational expert to
     CPU
  5: Reset and restart PMU
```

Figure 5.4: On-line learning algorithm platform.



Value of μ	Energy Loss (l_{ie}^t)	Perf Loss (l_{ip}^t)
$\mu >$ μ-mean	0	($\mu -$ μ-mean)
$\mu <$ μ-mean	(μ-mean $-\mu$)	0
Total Loss (l_i^t) $= \alpha \cdot l_{ie}^t + (1-\alpha) \cdot l_{ip}^t$		

Figure 5.5: Loss evaluation methodology

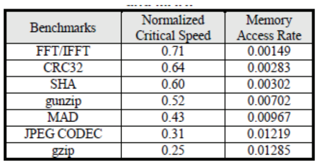
The Linux task data structure task struct is modified to include the weight vector to support accurate per task characterization in a multi-tasking environment. The process manager notifies the LKM of task creations (which is used to initialize the weight vector) and scheduler tick occurrences (at which point it runs the algorithm) and context switches.

Summarizing this framework: it shows a lightweight on-line learning approach to predict the optimum performance level for a task according to its percentage of memory accesses.

Both the energy loss and performance loss penalties are used to train a set of experts and are assumed linear with the frequency. This implies that no extra energy loss is accounted when a lower frequency increases execution time. Since there is no reset for the task weight discussed or implemented this predictor is not able to follow fast changes in program execution.

5.1.1.3 Memory-aware Dynamic Voltage and Frequency Prediction for Portable Devices (3)

This paper shows a DVFS approach to minimize the energy during different task phases addressing not only instant power savings but also minimizing the overall energy. Indeed it considers the power wasted due to possible extension of the execution time. This is done off-line by characterizing the critical frequency (that minimizes the system overall energy consumption) for different benchmarks and correlating it with the memory access rate (MAR), and on-line by using the correlation results to determine the critical frequency from on-line measured MAR. The MAR index quantifies the portion of cycles the CPU spends in memory accesses. It is defined as the ratio of the total number of cache misses (including both instruction and data cache misses) to the number of instructions executed. The relationship between MAR and critical frequency is obtained off-line as fit function, as shown in Fig. 5.6.



Benchmarks	Normalized Critical Speed	Memory Access Rate
FFT/IFFT	0.71	0.00149
CRC32	0.64	0.00283
SHA	0.60	0.00302
gunzip	0.52	0.00702
MAD	0.43	0.00967
JPEG CODEC	0.31	0.01219
gzip	0.25	0.01285

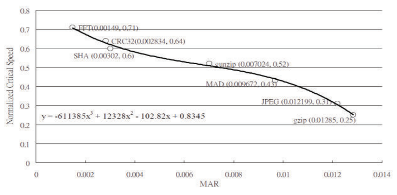


Figure 5.6: Correlation between critical speed and MAR.

This approach computes the MAR on-line and uses the off-line determined formula to obtain the related critical frequency. This frequency is then applied to the next scheduler interval. Thus, it implicitly adopts a last value prediction. In conclusion this paper shows the importance of accounting the overall energy consumption and proposes a technique to minimize it on-line. The off-line analysis shows that a non-linear relationship between MAR/CPI and critical frequency must be used to minimize the energy consumption. Unfortunately the one used on-line is statistically evaluated off-line and no further error analysis has been performed.

5.1.1.4 Feedback Control Algorithms for Power Manager of Servers (4)

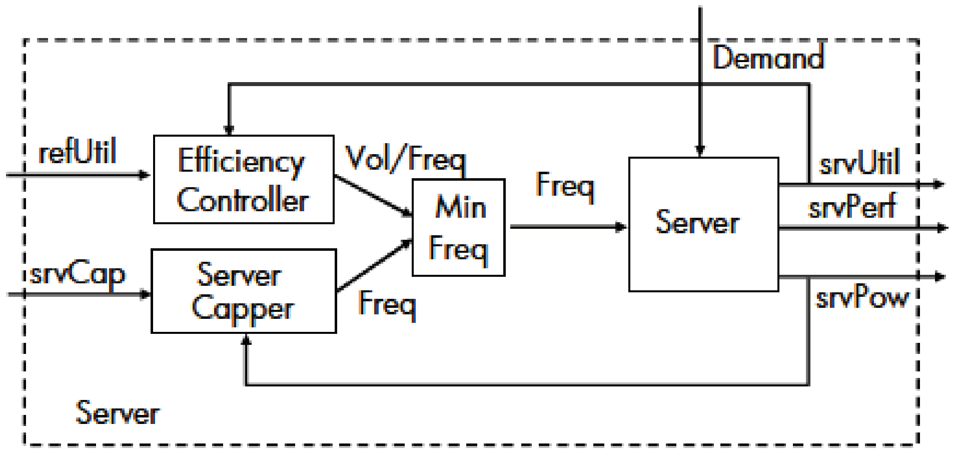
This paper presents a framework to reduce the power consumption of a multi-server system. At a processor level it achieves power savings by applying DVFS. The system is considered split in groups of servers, and a single server is the smallest unit. A global power reduction goal is achieved by imposing a power budget at each server group. Each server selects its own performance level to minimize the power without penalizing a target workload and to ensure to not overpass its power budget constraint. This is done relaying on a distributed and hierarchical set of controllers as shown in Fig. 5.7. It is made up of the following basic blocks:

- Efficiency Controller: there is one for each server. It tunes the performance level (voltage and frequency) to track a target workload, while minimizing its power. It is implemented using a PI controller;
- Server Power Capping: there is one for each server. It tunes the performance level (voltage and frequency) to track a given power budget. It is implemented as PI controller. The possible conflicts with the efficiency controller are solved by selecting the minimum performance level between the two.
- Group power capper: it distributes the group power budget to each server proportionally to their previous power dissipation.

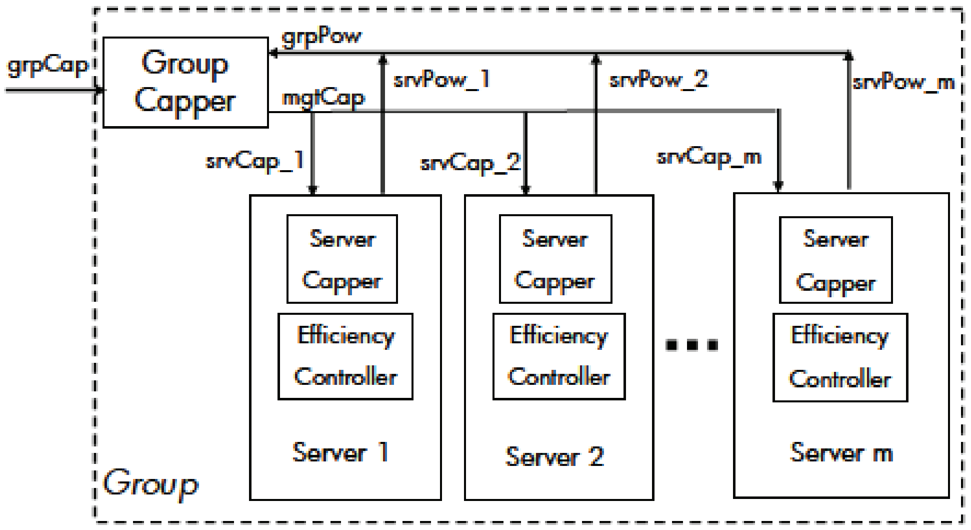
Fig. 5.8 shows the performance of this technique for different combinations of controllers. It can be noticed that the group power capper has the only effect of marginally reducing the performance loss, whereas the server capper in combination with the efficiency controller reduces the power consumption and ensures not to overpass the target power budget. Thus, the group power capper policy heuristic can be improved.

5.1.1.5 Improving Fairness, Throughput and Energy-Efficiency on a Chip Multiprocessor through DVFS (5)

This paper presents a framework to improve performance and energy efficiency of a single chip multiprocessor by increasing and balancing the fairness of the shared resources. This goal is obtained using a centralized controller to tune the frequency of each core, proportionally to the shared per core memory bus usage unfairness. Fig. 5.9 shows the concurrency effect on shared resources. It shows the execution time of a thread ”swim” when it runs alone or concurrently

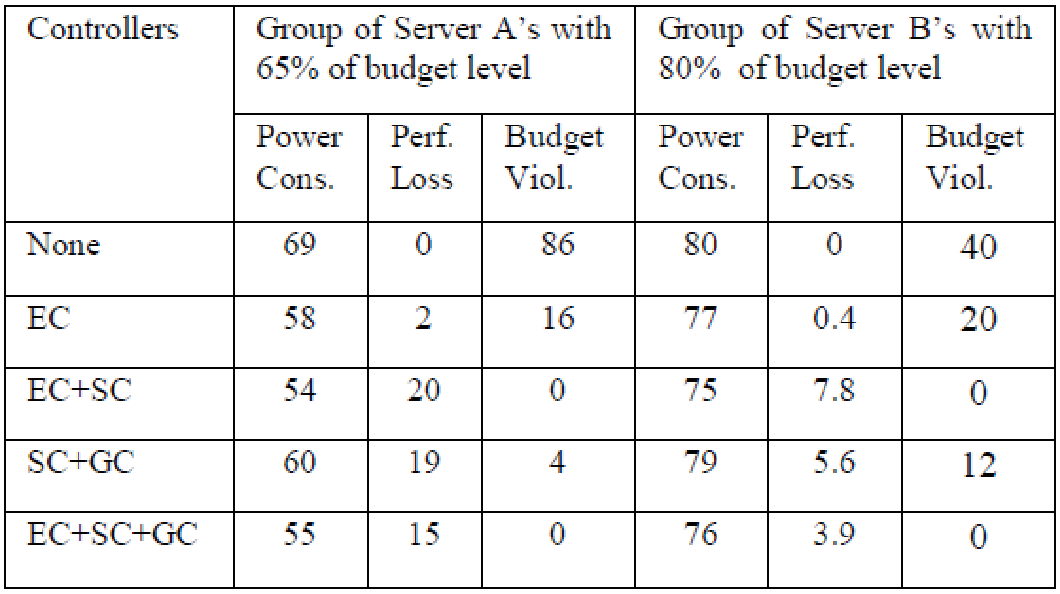


(a) A server under control of *Efficiency Controller* and *Server Capper*



(b) A group of servers with power consumption throttled by *Group Capper*

Figure 5.7: : Control system architecture



Controllers	Group of Server A's with 65% of budget level			Group of Server B's with 80% of budget level		
	Power Cons.	Perf. Loss	Budget Viol.	Power Cons.	Perf. Loss	Budget Viol.
None	69	0	86	80	0	40
EC	58	2	16	77	0.4	20
EC+SC	54	20	0	75	7.8	0
SC+GC	60	19	4	79	5.6	12
EC+SC+GC	55	15	0	76	3.9	0

Figure 5.8: Statistics on performance of two groups of servers (based on different processors) under different combinations of the controllers

with other treads in a two-core test architecture (private L2 cache and shared memory bus). It can be noticed that the performance degradation strongly depends on the co-scheduled thread and its number of last level cache misses. This work addresses this issue by dynamically changing the allocation of access slots of shared resources.

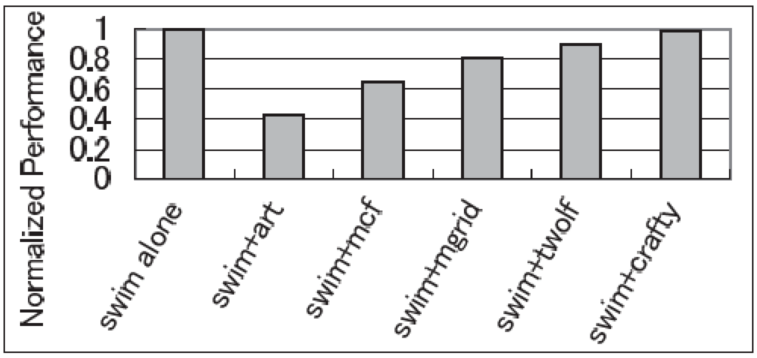


Figure 5.9: Performance degradation by bus contention.

The shared bus access equity for a given task can be obtained on-line using a last value prediction of the delay that CPUi imposes on the other CPUs while it is the bus owner. This quantity is computed on-line by counting the number of cycles a thread on CPUi keeps all the other CPUs waiting and is used for identifying the global bus-unfairness. Then, proportionally to this quantity, the controller selects the core frequencies to improve the global memory bus fairness. Thus, this method implicitly uses a last value predictor.

In conclusion this paper shows the importance of shared resources on global program execution behavior. However, the CPU frequency is not the best knob to constrain the memory access rate in memory bounded applications. Indeed the thread that overloads the memory bus must be memory bound . In addition the CPU frequency is shown, in previous results, to not significantly impact the execution time of a memory bounded task.

5.1.2 Thermal Management

The previous section showed the state-of-the-art techniques to predict the future task phases and select the frequency that minimizes the energy consumption while preserving the system performance. This dynamic energy minimization does not guarantee any thermal performance. In fact if a cpu-intensive task is scheduled repeatedly for a long period it will run at maximum frequency for a long period, pushing the cpu close to its thermal critical region. If this threshold is reached a protection hardware mechanism will stop the cpu to cool it down. This will impact performance. The next set of papers show how to use DVFS to prevent this problem, how to predict thermal problems and how to design the controller for multicore chips.

5.1.2.1 Control-Theoretic Techniques and Thermal-RC Modelling for Accurate and Localized Dynamic Thermal Management (6)

This paper mainly shows the benefits in limiting the maximum chip temperature using controller feedback approaches instead of threshold based ones. The results of this work are evaluated in a complete simulation framework made up by a superscalar cycle-accurate simulator connected to power (wattch) and temperature (hot-spots) simulators. A set of tests were performed to highlight the limitation of non-control based approaches and show the benefit of P, PI and PID feedback controllers. All variants are close-loop techniques and assume thermal sensors along the chip area.

Fig. 5.10 shows the performance of a non-CT (non formal feedback control) toggle controller, a manually-derived proportional controller (M), and a CT-toggle controller. All are able to completely eliminate thermal emergencies, so the metric of interest is performance. Recall that dynamic thermal management policies (DTM) only slow or disable parts of the processor, so the best accomplishment is to minimize the slowdown. Figure 10 therefore plots the

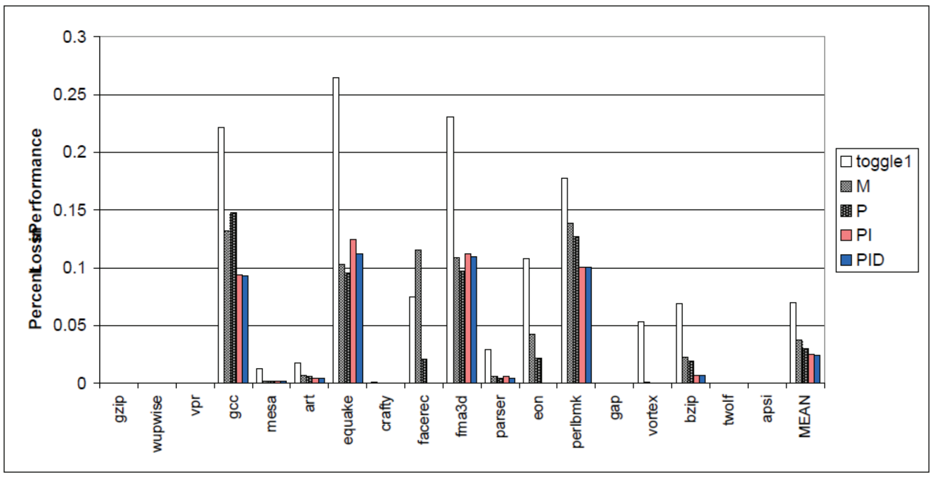


Figure 5.10: Performance loss for various DTM techniques relative to the IPC with no DTM. Smaller bars mean less loss in performance.

percentage loss in performance compared to the baseline case with no DTM. The adaptive techniques are substantially better than the fixed toggle policy, with the PI and PID techniques cutting the performance loss by about 65% compared to toggle, and by about 37% compared to the M controller. Even the P controller cuts the performance loss by 21% compared to the M controller.

The choice of a higher setpoint (temperature 107.9) for the PI and PID controllers is essential for their good performance. Fig. 5.11 shows the results when a setpoint of 107.5 is applied to these controllers (the setpoint used by the P controller). A PD controller is included in this graph too, for completeness. The PI and PID controllers still give impressive gains compared to the toggle controller, but are actually inferior to the simpler P controller. The reason for this is that these controllers are virtually doing a too good job in holding temperature near the setpoint. The P controller allows temperatures to rise as high as 107.9, while the other controllers hold the temperature extremely close to the setpoint of 107.5. Indeed, the temperature never exceeds 107.53 in the entire pipeline for the PD controller, and never exceeds 107.51 for the PI and PID controllers. This means that the setpoint for these controllers can be set be set to a higher temperature.

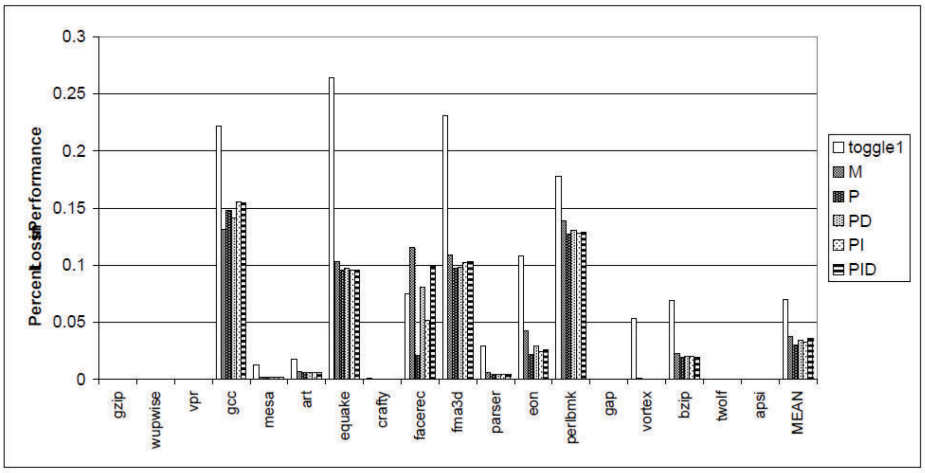


Figure 5.11: Performance loss for various DTM techniques relative to the IPC with no DTM, but with a setpoint of 107.5 for all the control-theoretic controllers (i.e., a trigger threshold of 107.0 for all the mechanisms is used).

5.1.2.2 Proactive Temperature Balancing for Low Cost Thermal Management in MP-SoC (7)

This paper shows how to proactively control the temperature of the chip to avoid hot-spots. This can be done by predicting the future temperature for each core and then allocating tasks and core performance to prevent temperature run-away and to mitigate future hot-spots across the chip area. The advantage of having a prediction ensures that no thermal violation will occur and allows lower design margins.

An autoregressive moving average (ARMA) technique is used for predicting the future thermal evolution for each core. If the observed stochastic process is stationary this technique produces a Gaussian white noise prediction residual. Since the workload characteristics are correlated across short time windows, and the temperature changes slowly due to thermal time constants, statistical stationary data can be assumed as input of the ARMA model. In addition by adapting the predictor parameters when the ARMA model no longer fits the workload, the stationary assumption does not introduce inaccuracy. Fig. 5.12 shows the performance of the ARMA predictor against the exponential one. It can be seen that the ARMA predictor provides accurate prediction even during thermal cycles.

As shown in Fig. 5.13 changes in program phases are detected by the usage of SPRT (Sequential probability ratio test) technique. It allows us to early detect changes in the statistical residual distribution (average, variance). In fact, if the ARMA prediction residual does

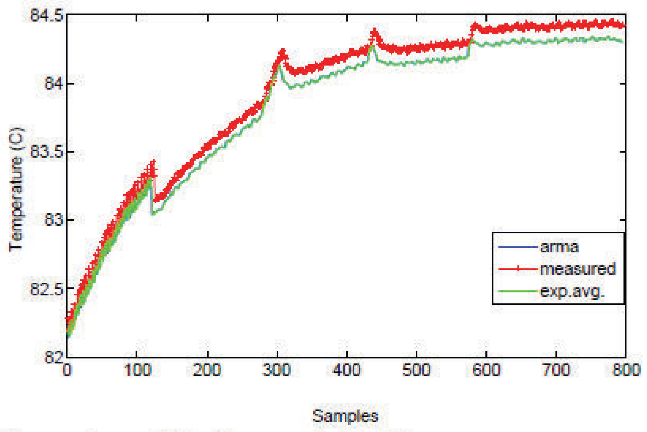


Fig. 4. Comparison of Predictors - Stable Temperature

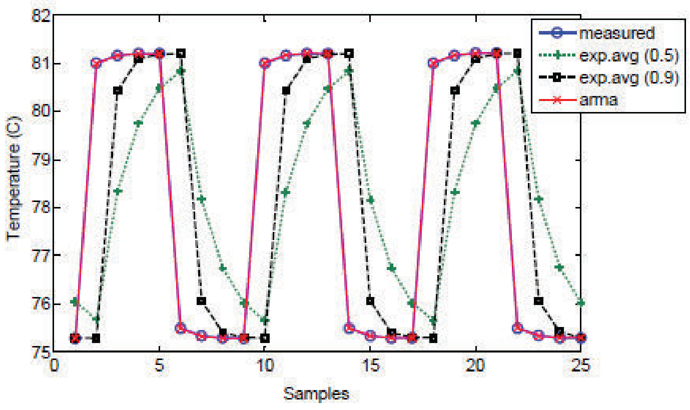


Fig. 5. Comparison of Predictors - Thermal Cycling

Figure 5.12: Comparison of predictors for stable temperature and thermal cycle

not show white Gaussian properties (null average) anymore it means that the prediction is no longer accurate and the predictor internal weight must be recomputed. SPRT does that; it constantly checks the residual average and resets the ARMA predictor if it recognizes a systematic derivation.

The results of this temperature predictor are used for minimizing the chip temperature and mitigating hot-spots. Different techniques are discussed and tested to achieve this goal:

- Reactive DVFS : reduces the voltage/frequency (V/f) level of a core if the threshold temperature is exceeded;

- Reactive thread migration: it migrates the workload from the hottest core to the coolest

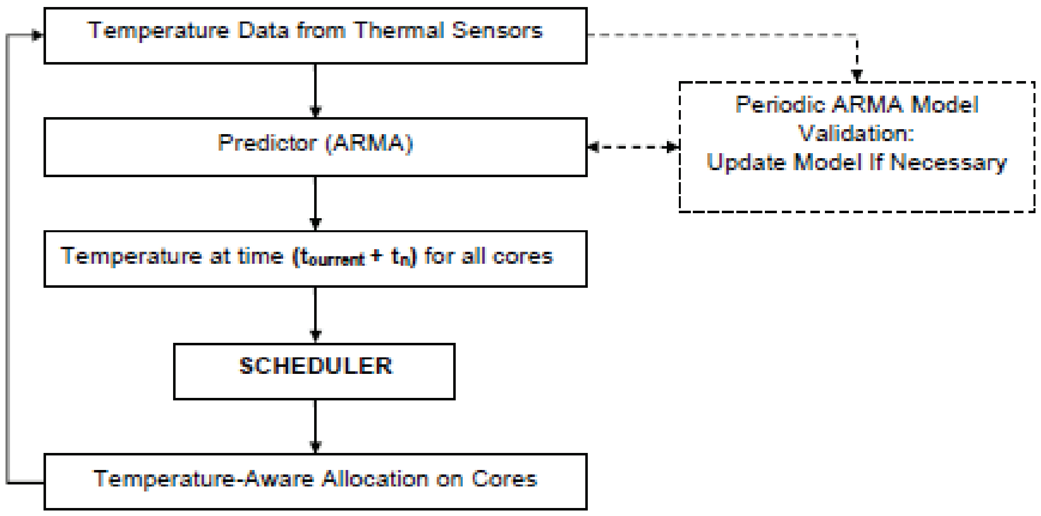
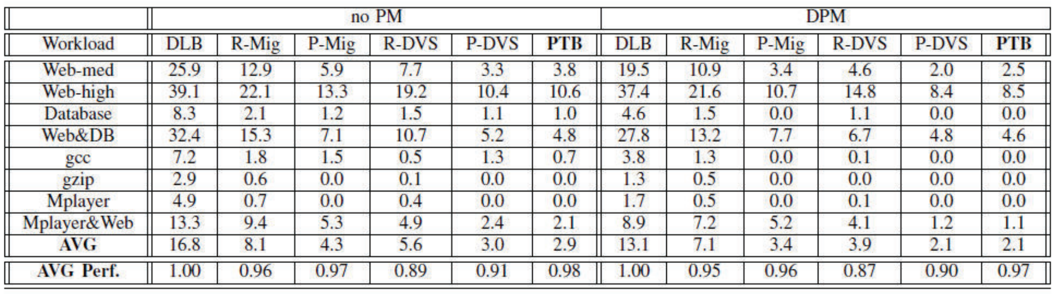

Temperature Data from Thermal Sensors
Predictor (ARMA)
Periodic ARMA Model Validation: Update Model If Necessary
Temperature at time ($t_{current} + t_n$) for all cores
SCHEDULER
Temperature-Aware Allocation on Cores

Figure 5.13: Proposed thermal predictor framework

core available if the threshold temperature is exceeded;

- Proactive thread migration: moves workload from cores that are projected to be hot in the near future to cool cores;
- Proactive DVFS: reduces the V/f level on a core if the temperature is expected to exceed the critical threshold;
- Proactive Temperature Balancing (PTB): at every scheduler tick, if the temperature of a set of cores is predicted to have imbalance in the next interval, threads waiting on the queues of potentially hotter cores are moved to cooler cores. Thus, the thermal hot spots can be avoided before they occur, and the gradients are prevented by thermal balancing.



	no PM						DPM					
Workload	DLB	R-Mig	P-Mig	R-DVS	P-DVS	**PTB**	DLB	R-Mig	P-Mig	R-DVS	P-DVS	**PTB**
Web-med	25.9	12.9	5.9	7.7	3.3	3.8	19.5	10.9	3.4	4.6	2.0	2.5
Web-high	39.1	22.1	13.3	19.2	10.4	10.6	37.4	21.6	10.7	14.8	8.4	8.5
Database	8.3	2.1	1.2	1.5	1.1	1.0	4.6	1.5	0.0	1.1	0.0	0.0
Web&DB	32.4	15.3	7.1	10.7	5.2	4.8	27.8	13.2	7.7	6.7	4.8	4.6
gcc	7.2	1.8	1.5	0.5	1.3	0.7	3.8	1.3	0.0	0.1	0.0	0.0
gzip	2.9	0.6	0.0	0.1	0.0	0.0	1.3	0.5	0.0	0.0	0.0	0.0
Mplayer	4.9	0.7	0.0	0.4	0.0	0.0	1.7	0.5	0.0	0.1	0.0	0.0
Mplayer&Web	13.3	9.4	5.3	4.9	2.4	2.1	8.9	7.2	5.2	4.1	1.2	1.1
AVG	16.8	8.1	4.3	5.6	3.0	2.9	13.1	7.1	3.4	3.9	2.1	2.1
AVG Perf.	1.00	0.96	0.97	0.89	0.91	0.98	1.00	0.95	0.96	0.87	0.90	0.97

Figure 5.14: Percentage of thermal Hot Spots and Performance Comparison

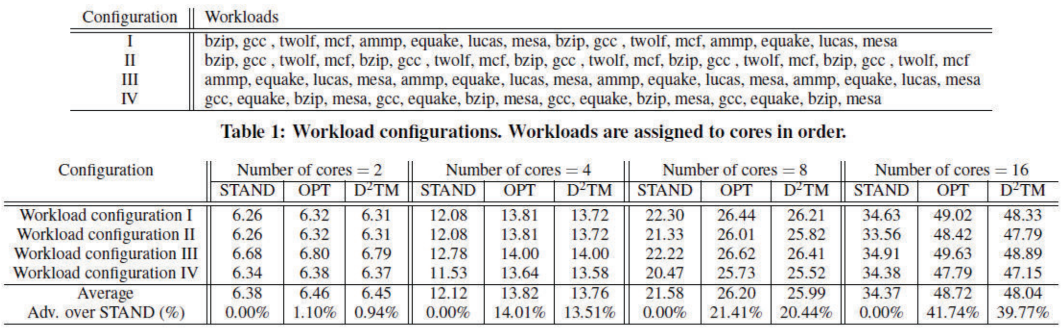
Fig. 5.14 reports the results in terms of percentage of hot-spots (ΔT ¿ 15 deg.) for different benchmarks and different thermal management techniques. The same tests are performed with and without an underlying energy management policy (DVFS and DPM). In both the cases the Proactive Temperature Balancing algorithm is performing best. It reduces the hot-spots significantly while keeping the performance overhead bounded.

In conclusion this paper shows an approach to predict the future temperature of each core in a multicore chip. These previsions are then used to improve the DTM policy performance.

5.1.2.3 Central vs. Distributed Dynamic Thermal Management for Multi-Core Processors: Which one is Better? (8)

This paper shows a methodology to maximize performance of a multicore system under thermal constraints adopting a distributed approach. It relies on having one feedback controller for each core to tune the system performance (voltage and frequency) to track a target temperature. The correct system behaviour is ensured even if spatial and temporal performance variabilities are present since the feedback controller output is modulated to ensure that no critical path delay error happens. This is obtained by using critical path replica sensors as input of the performance controller. Finally the results are compared with an oracle centralized approach.

A target temperature for each core is ensured by tuning the voltage and frequency levels of each processor with a dedicated PID controller: this prevents the usage of central controllers. The correct system behaviour under temporal and spatial performance variation is preserved by using a timing error avoidance scheme. It relies on adapting the system performance in a reactive fashion to reduce the error rate. Fig. 5.15 shows the per core controller architecture.



Configuration	Workloads
I	bzip, gcc , twolf, mcf, ammp, equake, lucas, mesa, bzip, gcc , twolf, mcf, ammp, equake, lucas, mesa
II	bzip, gcc , twolf, mcf, bzip, gcc , twolf, mcf, bzip, gcc , twolf, mcf, bzip, gcc , twolf, mcf, bzip, gcc , twolf, mcf
III	ammp, equake, lucas, mesa, ammp, equake, lucas, mesa, ammp, equake, lucas, mesa, ammp, equake, lucas, mesa
IV	gcc, equake, bzip, mesa, gcc, equake, bzip, mesa, gcc, equake, bzip, mesa, gcc, equake, bzip, mesa

Table 1: Workload configurations. Workloads are assigned to cores in order.

Configuration	Number of cores = 2			Number of cores = 4			Number of cores = 8			Number of cores = 16		
	STAND	OPT	D^2TM	STAND	OPT	D^2TM	STAND	OPT	D^2TM	STAND	OPT	D^2TM
Workload configuration I	6.26	6.32	6.31	12.08	13.81	13.72	22.30	26.44	26.21	34.63	49.02	48.33
Workload configuration II	6.26	6.32	6.31	12.08	13.81	13.72	21.33	26.01	25.82	33.56	48.42	47.79
Workload configuration III	6.68	6.80	6.79	12.78	14.00	14.00	22.22	26.62	26.41	34.91	49.63	48.89
Workload configuration IV	6.34	6.38	6.37	11.53	13.64	13.58	20.47	25.73	25.52	34.38	47.79	47.15
Average	6.38	6.46	6.45	12.12	13.82	13.76	21.58	26.20	25.99	34.37	48.72	48.04
Adv. over STAND (%)	0.00%	1.10%	0.94%	0.00%	14.01%	13.51%	0.00%	21.41%	20.44%	0.00%	41.74%	39.77%

Figure 5.15: Per-core timing-error avoidance thermal DVFS controller.Δs is the sampling period. Using a lookup table, the frequency is sent to 'sqrt(f)' to link the core voltage to the core's clock frequency.

The controller generally keeps the temperature of each core at about the maximum specification by varying the core's clock frequency, and optionally varying its core voltage as well. A core's temperature is compared to the set temperatures (typically 85C). The difference in temperature, i.e., the error e(t), is sampled every Δs seconds using an analog-to-digital converter (represented by KTN), and then the error is fed to a Proportional-Integral-Derivative (PID) process controller. The PID controller function is to produce a fast, stable system response. The output from the controller is converted to frequency ($_{KNf}$), and fed to both the core and the TEA avoidance circuitry. The TEA avoidance circuitry adjusts the frequency by incrementally lowering or raising the register 'N' value feeding the clock synthesizer. If the TEA circuit detects that a timing error is about to occur in the core, the down/up line is set to 1, lowering N and thereby reducing the clock frequency; otherwise, the down/up signal is cleared to 0, raising N and thereby increasing the frequency. Therefore the clock frequency of the core is approximately maximized for any value of the core's temperature.

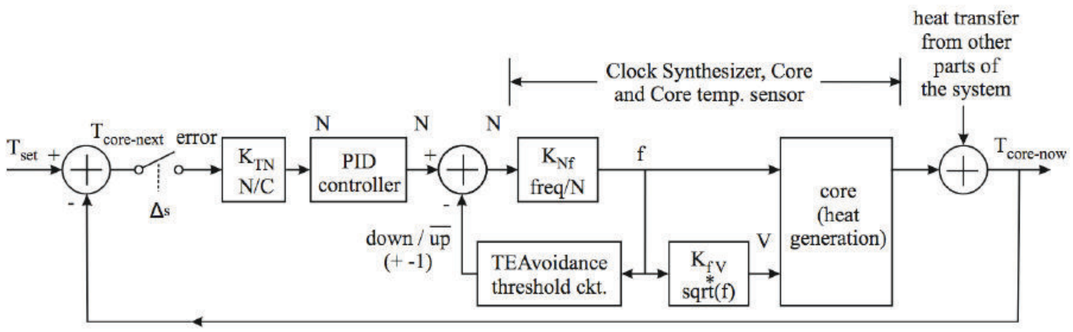


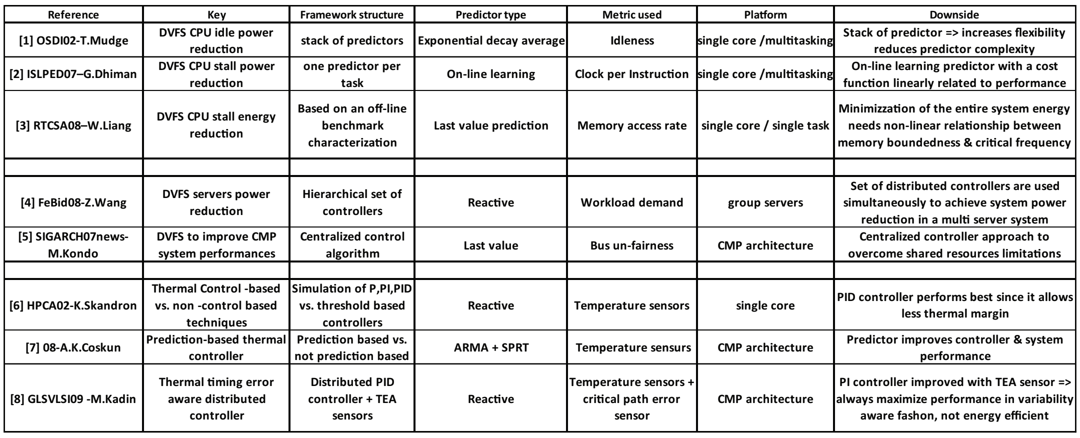
Figure 5.16: Total processor throughput (as measured by the sum of all core frequencies) for different management techniques.

Fig. 5.16 shows the performance of the proposed distributed approach for different numbers of cores. The distributed approach performs close to an oracle centralized approach.

5.1.3 Summary and Conclusion

In this section pros and cons are summarized for the analyzed approaches. Fig. 5.17 summarizes the key aspects of the discussed papers.

The first approach(1) shows the benefit of having a stack of predictors. It allows different performance decisions to coexist and takes advance from different task behaviors without increasing the predictor complexity. However, it targets deadline-based tasks and addresses



Reference	Key	Framework structure	Predictor type	Metric used	Platform	Downside
[1] OSDI02-T.Mudge	DVFS CPU idle power reduction	stack of predictors	Exponential decay average	Idleness	single core /multitasking	Stack of predictor => increases flexibility reduces predictor complexity
[2] ISLPED07–G.Dhiman	DVFS CPU stall power reduction	one predictor per task	On-line learning	Clock per Instruction	single core /multitasking	On-line learning predictor with a cost function linearly related to performance
[3] RTCSA08–W.Liang	DVFS CPU stall energy reduction	Based on an off-line benchmark characterization	Last value prediction	Memory access rate	single core / single task	Minimizzation of the entire system energy needs non-linear relationship between memory boundedness & critical frequency
[4] FeBid08-Z.Wang	DVFS servers power reduction	Hierarchical set of controllers	Reactive	Workload demand	group servers	Set of distributed controllers are used simultaneously to achieve system power reduction in a multi server system
[5] SIGARCH07news-M.Kondo	DVFS to improve CMP system performances	Centralized control algorithm	Last value	Bus un-fairness	CMP architecture	Centralized controller approach to overcome shared resources limitations
[6] HPCA02-K.Skandron	Thermal Control -based vs. non -control based techniques	Simulation of P,PI,PID vs. threshold based controllers	Reactive	Temperature sensors	single core	PID controller performs best since it allows less thermal margin
[7] 08-A.K.Coskun	Prediction-based thermal controller	Prediction based vs. not prediction based	ARMA + SPRT	Temperature sensurs	CMP architecture	Predictor improves controller & system performance
[8] GLSVLSI09 -M.Kadin	Thermal timing error aware distributed controller	Distributed PID controller + TEA sensors	Reactive	Temperature sensors + critical path error sensor	CMP architecture	PI controller improved with TEA sensor => always maximize performance in variability aware fashon, not energy efficient

Figure 5.17: Summary of the analyzed approaches

only the idle power by slowing down the execution time. Thus the approach is not suitable for server platforms. Instead, the second approach (2) learns the right performance level that has to be applied to the task on-line. The learning is driven by observing the last interval CPI and by computing the energy and performance costs for each possible performance level. The costs are a linear function of the performance. Thus no extra energy loss is accounted when the execution time is increased. The predictor chosen does not show any reset mechanism. It is not clear how it can cover different program phases. The third approach (3) shows how to correlate the memory access rate with the optimal frequency that minimizes energy. This relationship is non-linear. The non linearity is mainly due to execution time increases as draw-back of frequency scaling. The relationship is empirically obtained by an off-line benchmark characterization. All these approaches are related to single core architectures. (4) uses a set of distributed controllers to reduce the power in a group of servers. At lower level two controllers concurrently decide the local performance level to minimize energy and avoid power budget violations. (5) shows that in CMPs the CPI of a given task depends on the execution of concurrent threads. Thus program phase predictors must be aware memory usage of concurrent tasks. Finally, the last three papers address the problem of thermal management. The first one (6) shows that temperature feedback controllers (PID and PI) perform better than threshold based ones, since they allow lower safe thermal margins. The system can thus run at higher performance. (7) uses a thermal phase predictor to drive the task scheduler and the performance controller. This allows lowering the safety margin and increases performance. Finally the last paper (8) shows a distributed approach to bound CMP temperature by tuning the frequency according to the temperature sensor and a critical path's error rate. This allows the system to locally maximize the performance of all the cores in a robust fashion across temporal and spatial variability.

As result of this analysis in the following Section 5.2 we will investigate solution that will use a set of distributed controllers as demonstrated in (4). It will use an additional PID controller (chapter 6) and more advanced feedback control solutions(chapter 7) to regulate the core temperature to track a safe target temperature. It will work in conjunction to another controller that will select the frequency for minimizing the energy consumption(chapter 7). It will utilize the future CPI task prevision as input. By using a stack policy manager it will be possible to reserve different performance rules to special tasks: for example, apply the max performance for the first time a task runs. Thermal prediction as shown in (7) will be used to improve control policies(chapter 6).

5.2 Per Task Sensor (XTS) Linux Kernel Infrastructure

As already introduced early in this chapter the resource management architecture relies on the followings building blocks: Sensors, Actuators and control algorithm. In this section we describe a framework to explore the capability of the Intel Performance Monitor (Section 5.2.1) in characterizing the system core workload and the system performance state (frequency,CPI, idleness and temperature). The framework has been implemented completely in software as modification of the Linux O.S. with SMP extension enabled. It has been developed to be completely scalable over different systems and different degree of HW parallelism (Section 5.2.2). As we will see in Section 5.2.4 it does not only provide a distributed sensors driver, but also it provides the skeleton for a complete resource management implementation (Section 5.2.4.1). It provides also a powerful tool to profile a multi task application (Section 5.2.5).

5.2.1 Intel Performance Monitor

The Xeon® 3750 (9) is equipped with the Intel Performance Monitor Architecture 2 (10) that relies in three fixed counters and two programmable ones. We did support only the first set of counters, but our approach can be easily extended to take advantage from the programmable ones. The three fixed counters account for the following quantities, respectively:

- Instruction retired (FIX0);

- Number of un-halted clock cycle expired at the reference frequency (Maximum frequency) (FIX1);
- Number of un-halted clock cycle expired at the current frequency (FIX2);

All of these counters are completely manageable by accessing at a dedicated Model Specific Registers (MSR). Indeed as described in (10) the IA32_PERF_GLOBAL_CTRL enables /disables event counting for all or any combination of fixed-function PMCs (IA32_FIXED_CTRx) or any general-purpose PMCs via a single WRMSR. The IA32_PERF_GLOBAL_STATUS allows instead software to query counter overflow conditions whereas IA32_PERF_GLOBAL-_OVF_CTRL allows software to clear counter overflow conditions. The value counted by each counter can be read via a single RDMSR (MSR_CORE_PERF_FIXED_CTRX). Instead by writing in the same register via a single WRMSR it can be resetted. Other events can be monitored by the same procedure, but does not relies on performance counter unit and own a dedicated registers. These event are the temperature of the core and the time step counter(tsc). The first can be read by accessing the MSR_IA32_THERM_STATUS register with the mask (0x007F0000)[1] . All this infrastructure, counters and control registers, is repeated for each core in the platforms. In a multicore system this implies that the core that executes the msr instruction will access only to its MSR register. To force the reading of a specific core, within the platform, performance monitor it must be forced the execution of the WRMSR/RDMSR inside the core we want to access the register. Linux kernel provides a powerful abstractions to support it, named per_cpu_variable.

5.2.2 Per-CPU Variable

When it is created a per-CPU variable each processor on the system reads its own copy of the variable. This leads to the following advantages:

- Access to per-CPU variables requires (almost) no locking, because each processor works with its own copy.
- Per-CPU variables can also remain in their respective processors' caches, which leads to significantly better performance for frequently updated quantities.

[1]It contains the absolute value of the difference between the current temperature and the maximum allowed one (100C).

The declarations for per-CPU variables can be found in ¡linux/percpu.h¿. Basically a per-CPU variable is an array of data structures, one element per each CPU in the system. They must be used with the following precaution and advantage: a CPU should not access the elements of the array corresponding to the other CPUs; on the other hand, it can freely read and modify its own element without fear of race conditions, because it is the only CPU entitled to do so. The elements of the per-CPU array are aligned in main memory so that each data structure falls on a different line of the hardware cache [REFERENCE LDD3]. Therefore, concurrent accesses to the per-CPU array do not result in cache line snooping and invalidation, which are costly operations in terms of system performance. In our implementation we use it to save the reading of the performance counter inside a per-CPU variable. This allows us to have a single function that regarding on which CPU executes it saves the performance counters value in the proper per-CPU variable.

5.2.3 Metrics

As described in the related work section (Section 5.1) we want our systems be capable to tune the frequency of each core as reaction to workload phase changes (clock per instruction, idleness) and temperature trends. Indeed the XTS framework derives from the performance counters reading the following metrics:

- CPI (clock per instruction): this is obtained as ratio between the clock un-halted cycles at reference frequency and the instruction retired in the observed period (FIX1/FIX0).
- WL (percentage of cycle un-halted): this is obtained as ratio between the un-halted cycles at reference frequency and the time step counter (FIX1/TSC).
- Temperature: this is obtained directly by reading the dedicated sensor.
- Current Frequency indicator: this is obtained as the ratio between the clock un-halted cycles at current frequency and the clock un-halted cycles at reference frequency (FIX2/FIX1). It must be noticed that this is the only way to obtain a reliable frequency value. In fact hardware can select a frequency that is different from the one applied by the driver[1].

[1]CPU freq driver.

5.2.4 Architecture

The framework as been designed to not only access the PMU counters and the temperature sensor at each δ T[1] , but as described in previous section to compute a set of performance metrics linked with the task currently in execution on the CPU. This allows building a metrics history that follows the task when it migrates across different CPUs. Bringing to a task characterization rather than a CPU characterization. Indeed we believe that is a key feature to improve the accuracy of the workload estimator, key block in a resource management solution. In fact if task migration is allowed, The estimation of the future workload done by looking at each CPU in an isolated way can brings to wrong estimation when the task is scheduled out. We want to override this effect by let the workload estimation follow the task along different CPUs. The framework is made up by two basic section: a set of functions to access the PMU and the sensors for the current cpu and a mechanism to compute performance metrics and bind it to the running task.

Fig. 5.18 shows the framework functional diagram. During the kernel init phase an init routine create a per-CPU structure to hold the performance counter readings (xts_monitor_data). Than at each task creation a new structure is added to the task structure (xts_stat). At the beginning of a scheduler tick the scheduler routine[2] call the xts_start routine. This routine will clear the performance counters. When the scheduler quantum expires the scheduler routine calls the xts_update routine. This function will first read the performace counters and sensors(xts_read) and save the values in a per-CPU data structure(xts_monitor_data). Consequently it will access the same structure and combine the value to compute the performance metrics described in Section 2.6 . The newly generated metrics will be used to compute an estimation for the future quantum. This estimation will be then updated in a dedicated structure (xts_stat) inside the task_structure of the task it was running on the CPU. Before ending the routine calls a function to update the frequency accordingly to the future estimated workload[3].

5.2.4.1 Functions Description

In this section is described in more details the operation executes by the main functions:

[1] It is synchronized with the kernel scheduler tick.

[2] The scheduler routine executes on the core it relies to, and is not pre-emptive. Thus no synchronization mechanism are required.

[3] This capability is not currently implemented.

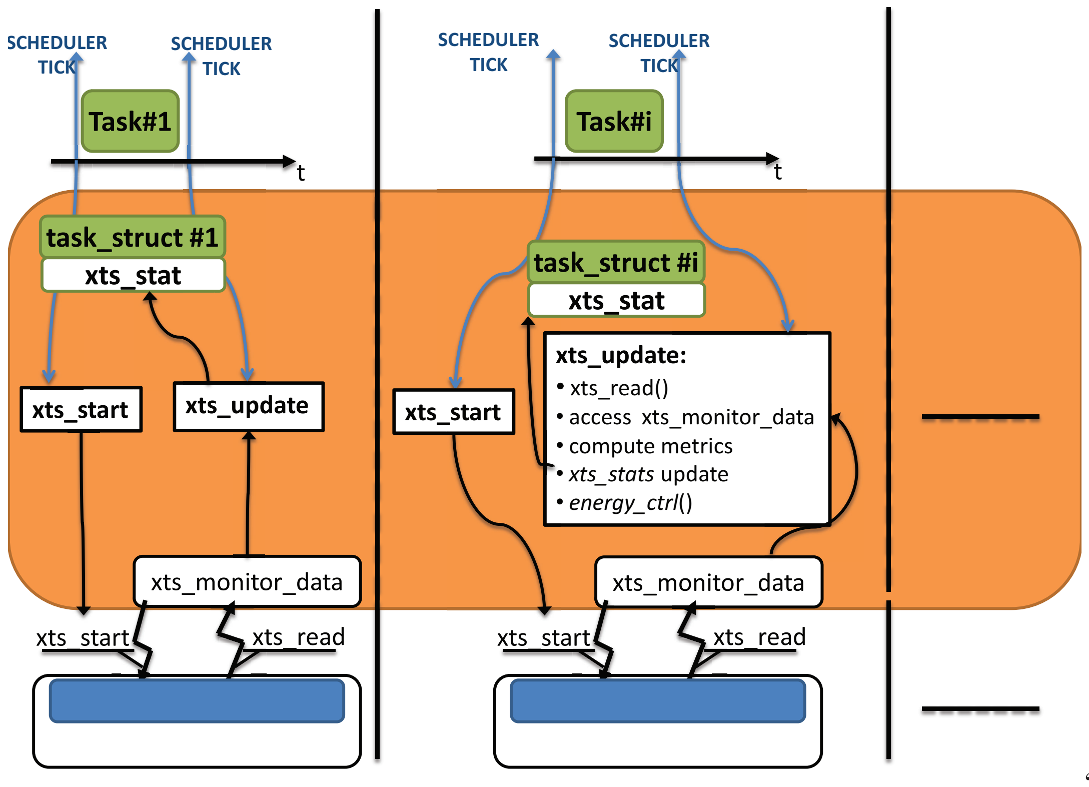


Figure 5.18: XTS Functional description

- void xts_start(void) : resets the performance counters and saves the initial TSC value in a per-CPU structure (xts_monitor_data).
- void xts_read(void) : It reads the performance counter and the temperature sensors on the current CPU and saves them in a per-CPU structure (xts_monitor_data).
- void xts_update(struct task_struct *xts_curr): It first calls the xts_read function. Then it access the xts_monitor_data for the current cpu and uses the data to compute the performance metrics related to the expired quantum. It then uses this values to predict the future evolution of this metrics. The prediction policy currently implemented is the last value prediction, but it can be easily substituted with more complex prediction algorithm. The predicted metrics are then saved on the per task structure xts_stats of the current task.

5.2.5 Export Statistics Functionality

As already introduced in previous section the framework as been thought to be not only the skeleton for a performance management controller implementation but also a powerful tool for profiling application in a multi-thread and multi-core scenario. Indeed the framework can be configured (XTS_EXPORT_STATS) to let each core memorize in an dedicated internal buffer the metrics and the taskID they belong to. In order to flush in the user space the buffer with the sampled statistics, when it is full , a new device driver has been created (/dev/xts) and an application has been developed to save the statistics into a file. Fig. 5.19 shows the functional diagram of the export statistics framework. We can easily notice that it requires three different parts: Linux Kernel : The xts_update functions it has been enhanced to save the metrics computed in a buffer (BuffX) in an incremental way. This is done if the per-CPU variable xts_exp_start is bigger than zero. There are two buffer for each CPU and each entry of the buffer is a xts_stat_data structure. Each one of this contains the following field:

- cpi : it contains the CPI measured in the last expired quantum.
- freq : it contains the frequency measured in the last expired quantum.
- wl : it contains the percentage of not halted cycle measured in the last expired quantum.
- temp : it contains the temperature of the core.
- pid : it contains the PID it was in execution in the specific CPU during the last expired quantum.

- CPU : it contains the current CPU identifier the stats refers to.
- tsc : it contains the time stamp counter value at the beginning of the last expired quantum.

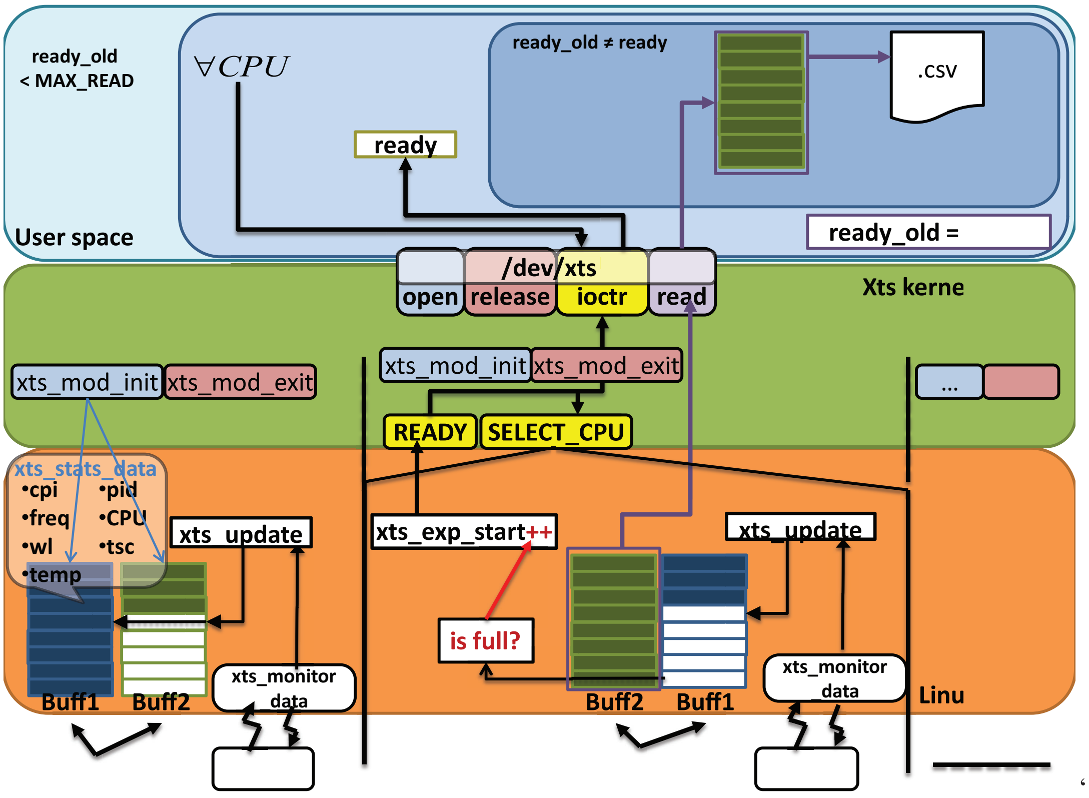


Figure 5.19: Export statistic functional description

As soon the buffer is full a counter variable is incremented (xts_exp_start) and the second buffer is selected to be the next to be filled in. If the other buffer still be full the sampling process stops until it is empty. This double buffer approach allows a continuous sampling of the statistics, indeed while the one buffer is saving the new data, the other one can be flushed to the user space. The dimension of the buffer is selected by the macro XTS_EXPORT_LENGTH.

Xts Kernel module : This kernel module has two main tasks: first allocate the buffers used by each and communicate to xts_update function the pointer to them[1]; secondly creating a

[1]Each cpu will have a private set of two buffer, and the correctness of the access is guaranteed by saving the pointer of each buffer in a per-cpu variable.

dedicated device driver and a mechanism to flush the collected data in the user space. The device driver exposes to the user space the following available method:

- open : this method calls for each cpu[1] the xts_mod_init function. This functions allocate two buffer of XTS_EXPORT_LENGTH* sizeof(xts_stats_data) dimension. The pointer of both is then saved in two externally declared (Linux kernel) per-CPU array ans the per-CPU variable xts_exp_start is cleared.
- release: this method calls for each cpu[2] the xts_mod_exit function. This functions clear the memory allocated by the xts_mod_init function and the xts_exp_start variable.
- ioctrl: this method allow a generic user space application to select the CPU for which we want to flush the buffer and to read the xts_exp_start variable for the selected CPU. If this variable as been incremented since last reading means there is full buffer to be flushed for that CPU. The absolute number instead contains the number of buffer flush that has been performed since the driver initialization.
- read: this method flushes the buffer full for the selected CPU into a same size user-space pre-allocated buffer.

User space application : In order to correctly read the sampled values for each CPU the application must execute the following operations:

1. Opening the device driver ”/dev/xts” ;
2. Allocating a buffer of XTS_EXPORT_LENGTH* sizeof(xts_stats_data) dimension;
3. Defining the number of consecutive read to perform (MAX_READ);
4. Clearing the ready variables. (ready_old=ready=0);
5. While the ready_old variable is less than MAX_READ do:
    a) For each CPU :
        -i- Selecting the specific CPU with the SELECT_CPU ioctrl method;
        -ii- Updating the ready variable with the READY ioctrl method output;

[1]This is done by using the work_on_cpu linux kernel functionality.
[2]This is done by using the work_on_cpu linux kernel functionality.

-iii- While differs from ready_old variable sleep for one second and retrigger point (ii);

-iv- Flush the kernel buffer for the selected CPU into the user-space buffer;

-v- Write the content of the user-space buffer in a trace file.

b) Ready_old = ready

6. Freeing the user space buffer;

7. Closing the device driver ”/dev/xts”;

5.2.6 XTS Performance

The described framework has been tested on a linux SMP distribution based on 2.6.29.6 linux kernel. We perform several tests to understand the performance and the stability. First of all we evaluate the overhead of a single performance counter access (done with a MSR write and read). It turns out to take for both read and write 200-300 Clock, that are roughly 80-100ns. That is a negligible overhead. Regarding instead the full framework we did measure the execution time of a series of synthetic benchmarks on the same platform running first the original linux kernel, then the xts patched one, then the xts patched one with tracing enabled. Between the first two configuration no overhead has been noticed, whereas the xts with tracing enabled and working introduced an overhead of the less then 0.02%.

5.3 Benchmarks and program phases

As introduced in Section 5.1 programs flows are not constant along their computation, but expose different program phases. This exacerbate the performance of static power and thermal design and also represent a challenge to dynamic solutions. Indeed resource management solution should be enough fast and smart to recognize them and react to abrupt phase changes. In this section we carried out a characterization of few corner-case benchmarks of a public available benchmark suite, named PARSEC(11) that represent the state-of-the-art of future application. We tested them in a real 16 core server platform, running our Per Task Sensor (XTS) Linux Kernel Infrastructure.

5.3.1 Hardware set-up

The following profile experiments are based on measurements on an Intel® server system S7000FC4UR. It runs four quad-core Xeon® X7350 processors at 2.93GHz and has a total memory capacity of 16GB based on FBDIMMs. The Xeon® X7350 consists of two dual-core Core™2 architecture dies in a single package. Each of the two dual cores share a common 4MB sized L2 cache.

The platform runs a CentOS 5 Linux operating system, based on linux 2.6.29.6 kernel. We patched the kernel source code with our Per Task Sensor (XTS) Linux Kernel Infrastructure modification presented in Section 5.2.

5.3.2 Parsec Benchmarks

The Princeton Application Repository for Shared-Memory Computers (PARSEC) (11) is a benchmark suite composed of multithreaded programs. The suite focuses on emerging workloads and was designed to contain a diverse selection of applications that is representative of next-generation shared-memory programs for chip-multiprocessors. The benchmark suite with all its applications and input sets is available as open source free of charge and combine different type of program kernels with the aim of describing and representing future uses cases of not only high performance computing, but of general server and consumers parallel application. Within the different benchmarks we selected a set of them, namely Fluidanimate, Bodytrack and Raytrace that show different memory access and parallelization patterns.

5.3.3 Characterization results

For each of the selected benchmarks, and for each core, we generated traces of the CPI, PID and core temperature under three different parallelization and scheduling patterns: i) Four thread running on four different cores in different chip; ii) Four thread running on four different cores in same chip; iii) sixteen thread running in all the sixteen available cores.

5.3.3.1 Fluidanimate

This Intel RMS application uses an extension of the Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid for interactive animation purposes. It was included in the PARSEC benchmark suite because of the increasing significance of physics simulations for animations.

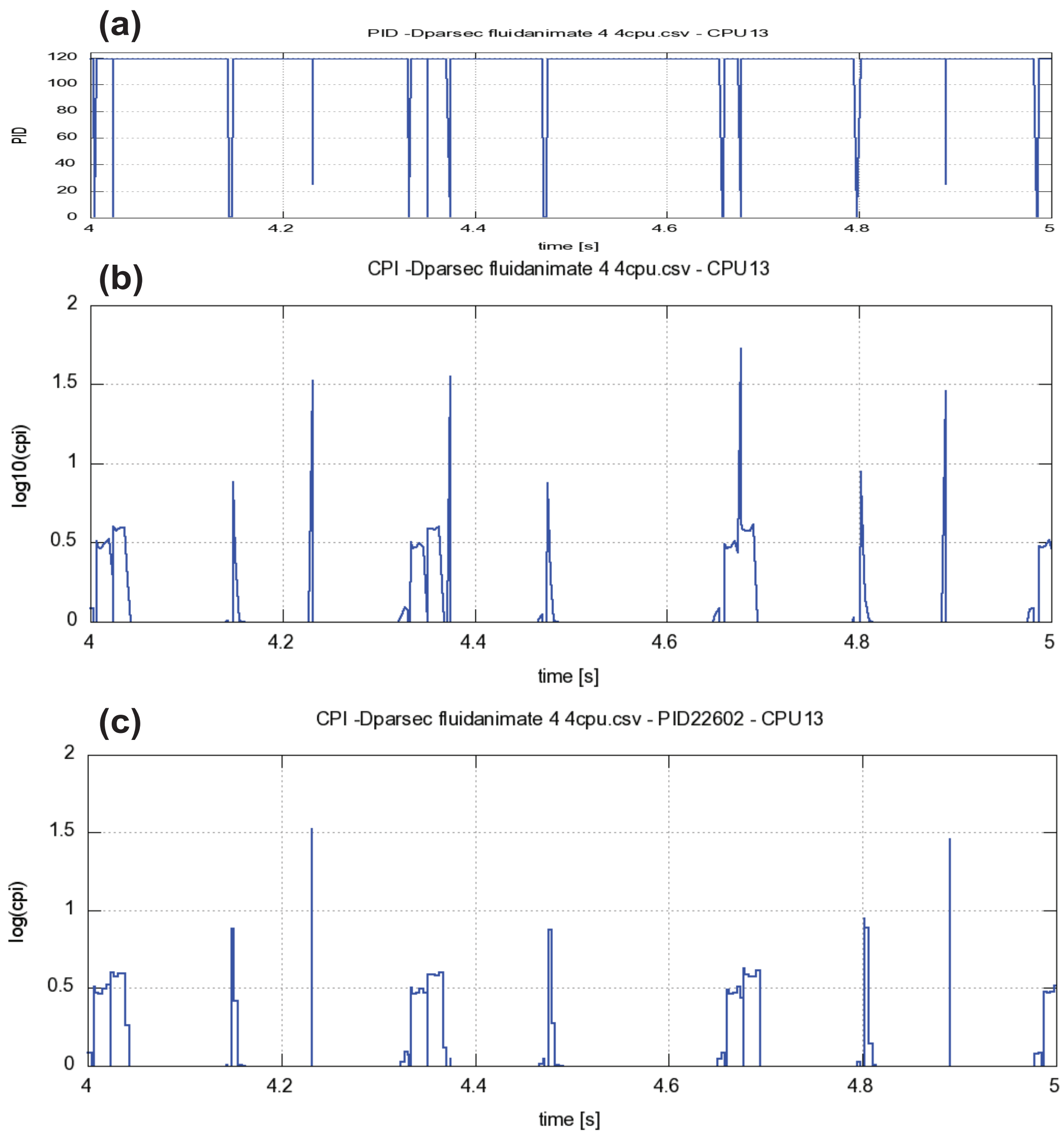


Figure 5.20: XTS PID and CPI traces for parsec fluidanimate benchmark when considering all thread or just one thread

Fig 5.20.a shows the CPI trace for portion of the region of interest (ROI) for one core in the case i). Fig 5.20.b shows the shows in the same time periode the sequence of different task(different PID) in execution on the same core. Our XTS generate trace allows us to filter only the data of a specific task, allowing us to discriminate where a given CPI value is due to the benchmark or to interrupt and kernel service routines. Fig 5.20.c shows the CPI trace for the only fluidanimate thread. It can be noticed that the thread shows regular phases with period of roughly 300ms, with portion of 30-40ms of constant CPI.

Fig 5.21.a shows instead the same plot for ii) case, as can be imagined by moving all the thread in the same chip, the memory subsystem becomes more tight and the phase period slightly increase, the CPI constant portion became longer, roughly double and CPI absolute values increase of a factor of 10x. Fig 5.21.b shows the iii) case for it, it can be noticed that the absolute CPI values do not change from previous case ii) but the constant CPI phase are now on the order of tens ms, on the same order of the kernel scheduler period. Fig 5.21.c shows the temperature variation due to the same workload ones, it can be noticed that the program phases are strongly correlated with the temperature cycles. Thus there is a tight correlation between CPI, power and core temperature. Indeed during high CPI phases the temperature decreases and vice versa during low CPI phases (cpu-bound) the temperature increases.

5.3.3.2 Bodytrack

This computer vision application is an Intel RMS workload which tracks a human body with multiple cameras through an image sequence. This benchmark was included due to the increasing significance of computer vision algorithms in areas such as video surveillance, character animation and computer interfaces.

Fig 5.22.a shows the CPI trace for portion of the region of interest (ROI) for one core in the case i). Fig 5.22.b shows the shows in the same time periode the sequence of different task(different PID) in execution on the same core. As before XTS generate trace allows us to filter only the data of a specific task, allowing us to discriminate where a given CPI value is due to the benchmark or to interrupt and kernel service routines. Fig 5.22.c shows the CPI trace for the only bodytrack thread. It can be noticed that the thread shows regular phases with period of roughly 70-80ms, with now really small portion of constant CPI.

Fig 5.23.a shows instead the same plot for ii) case, differently from previous case, by moving the four thread in the same cache region decreases significantly the average CPI of the thread. This suggest that bodytrack as less cache requirement and more data exchange that

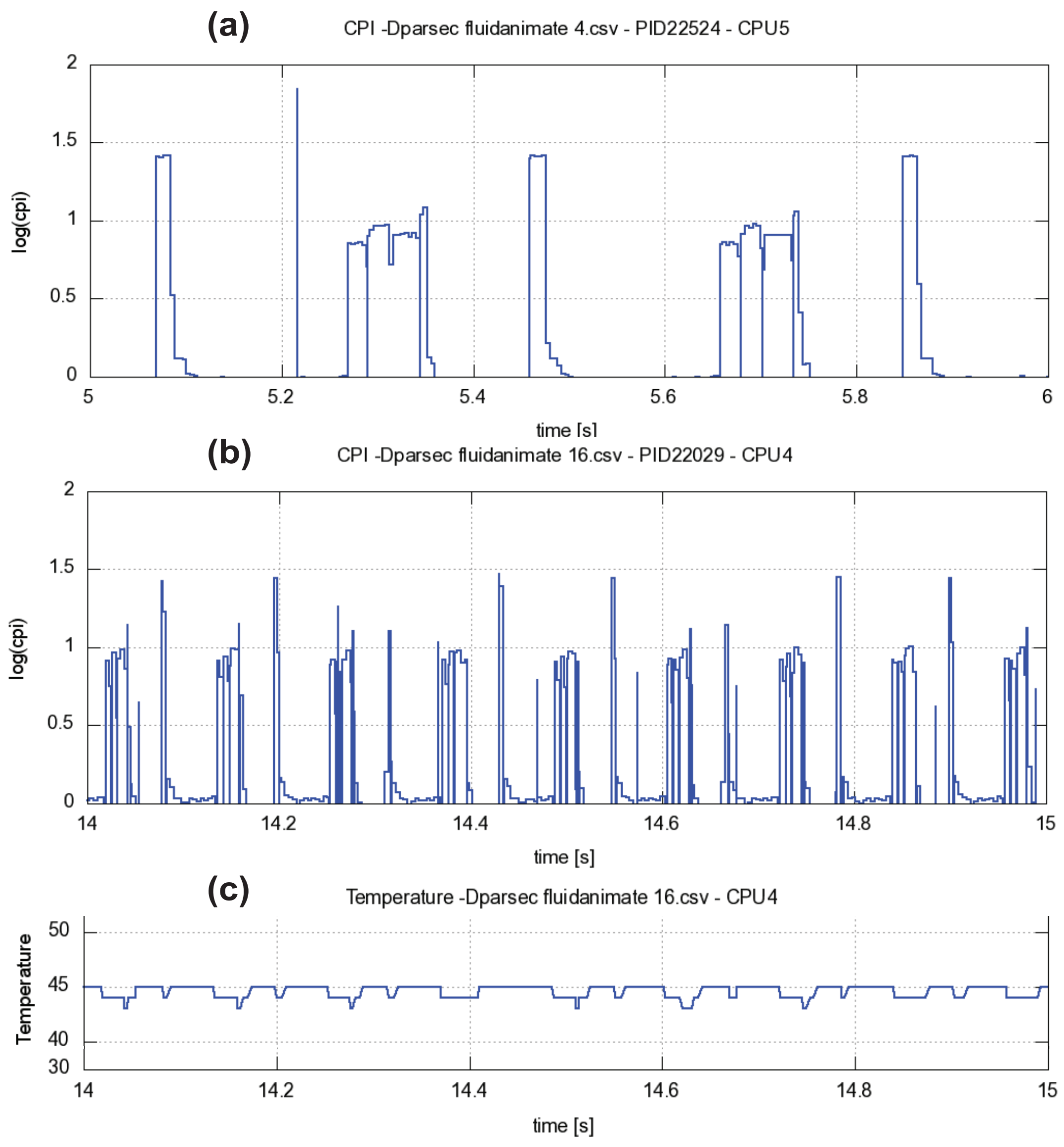


Figure 5.21: XTS CPI and Temperature traces for parsec fluidanimate benchmark under different thread allocation and parallelizzation

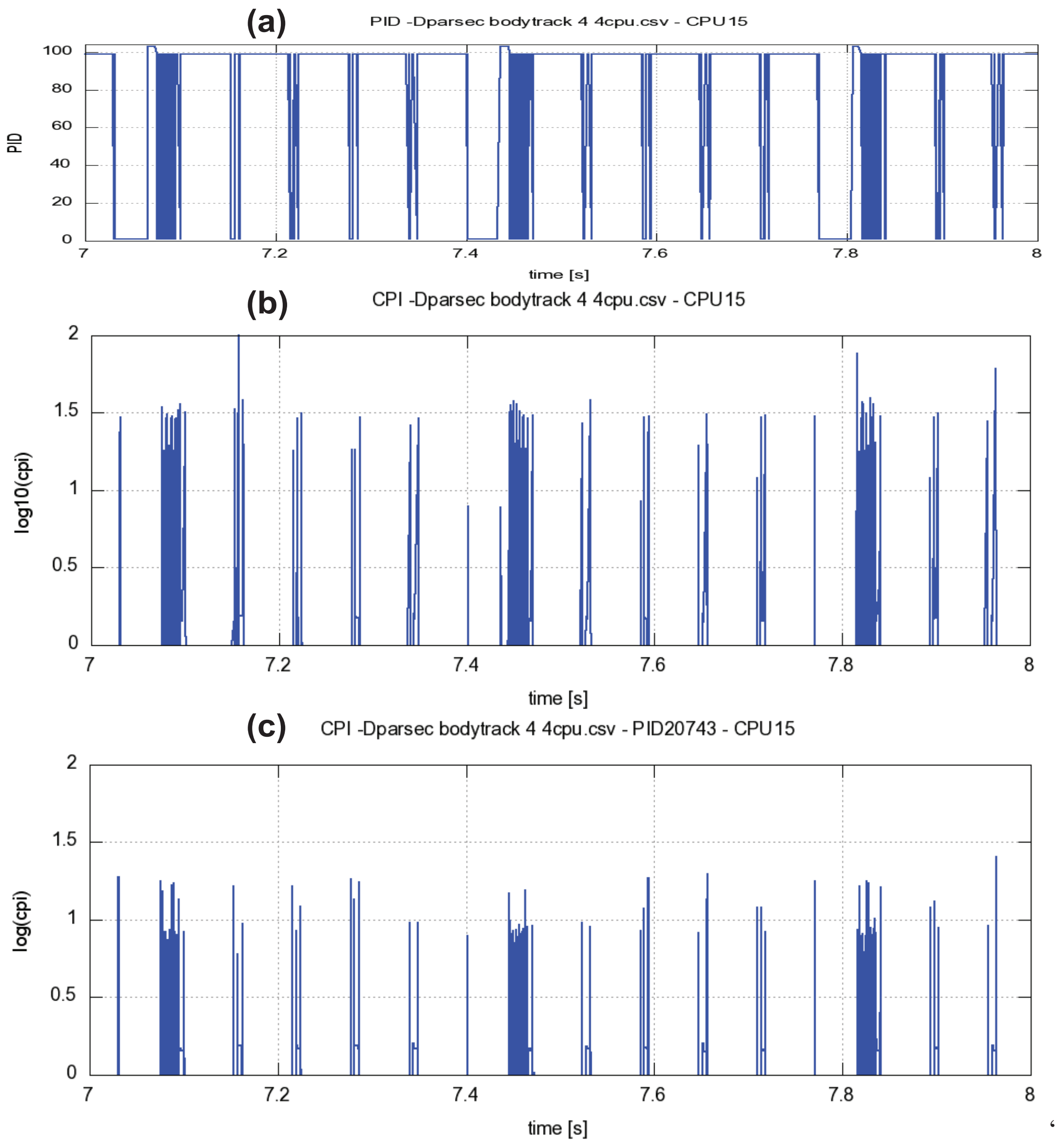


Figure 5.22: XTS PID and CPI traces for parsec bodytrack benchmark when considering all thread or just one thread

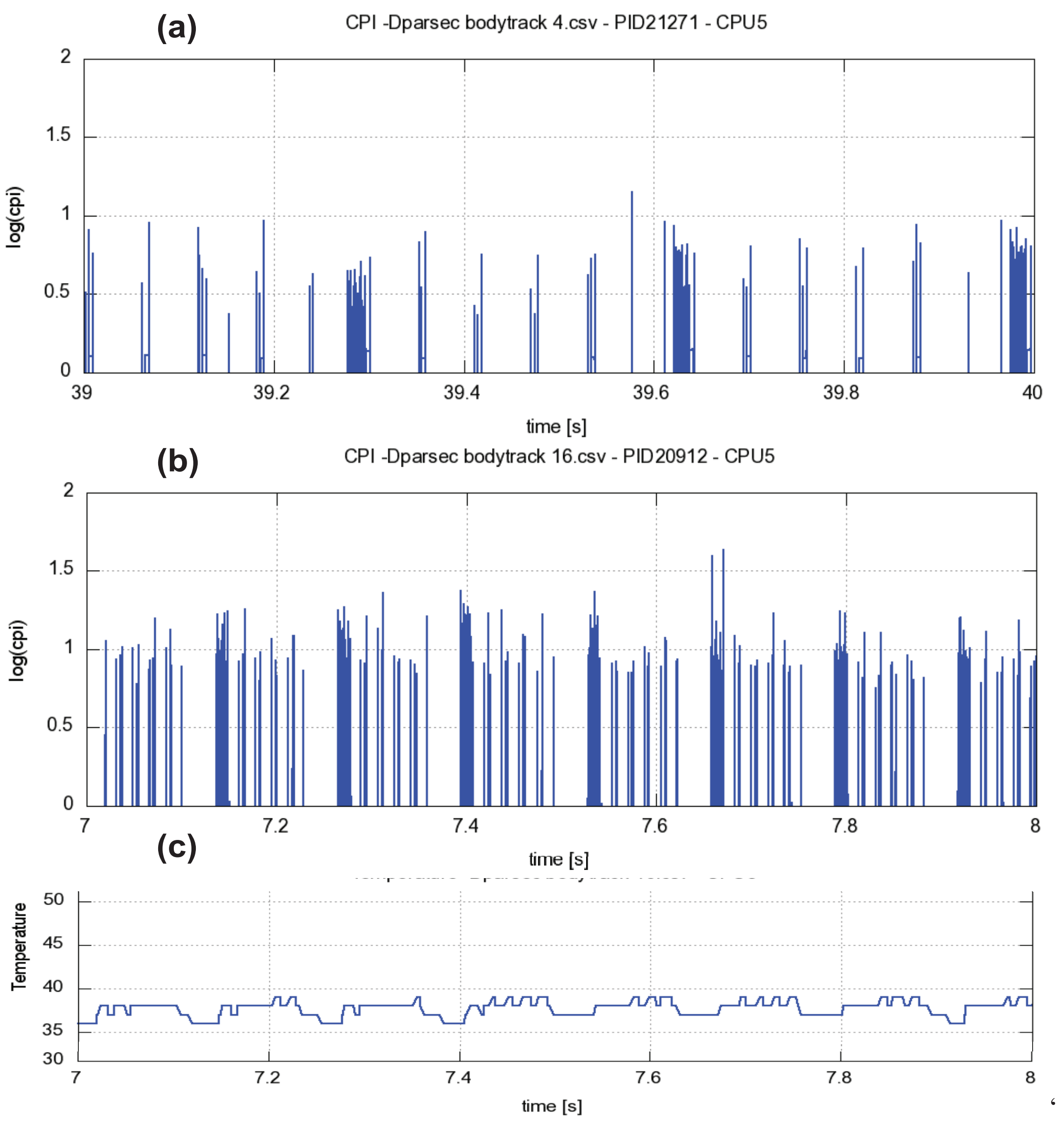


Figure 5.23: XTS CPI and Temperature traces for parsec bodytrack benchmark under different thread allocation and parallelization

benefit from data proximity. Fig 5.23.b shows the iii) case for it, it can be noticed two fold the portion of low CPI decreases the duration and the absolute CPI value, whereas the high CPI portion increases their CPI value. The distance and the duration of the phases decrease. As before Fig 5.23.c shows the temperature variation due to the same workload ones. As before high CPI phases cools down the core whereas low CPI phases (cpu-bound) produce a core temperature increase.

5.3.3.3 Raytrace

The Intel RMS application uses a version of the raytracing method that would typically be employed for real-time animations such as computer games. It is optimized for speed rather than realism. The computational complexity of the algorithm depends on the resolution of the output image and the scene.

Fig 5.24.a shows the CPI trace for portion of the region of interest (ROI) for one core in the case i). It can be noticed that is more regular than previous one. And constant CPI phases are longer. Fig 5.24.b shows instead the same plot for ii) case. As fluidanimate moving the four tread in the same cache region, worsen the memory performance increasing the CPI absolute values. Fig 5.24.c shows the iii) case for it, it can be noticed two fold first the phase period decreases significantly. Secondly it expose region with extremely high CPI. Thus the combination of both memory and cpu bounded phases.

5.3.4 Remarks

This characterization first highlights the benefit of our proposed XTS linux framework in analyze benchmark performance. Thanks to that it has been possible demonstrate that program phase exists in real applications. The time of them is close to the operating system one, thus operating system have the capability to react to follow them. This results also shows the importance of thread scheduling in program performance and phase behavior. The core temperature measurements also shows the dependency of program phases on core temperature and thus on core power. Also they shows thermal small thermal cycle due to program phase changes.

5.4 Power and Thermal Characterization and Modeling

In this section we describe a set of tests performed the power model used for extracting the power consumption of a general purpose processor, considering the effects of different power

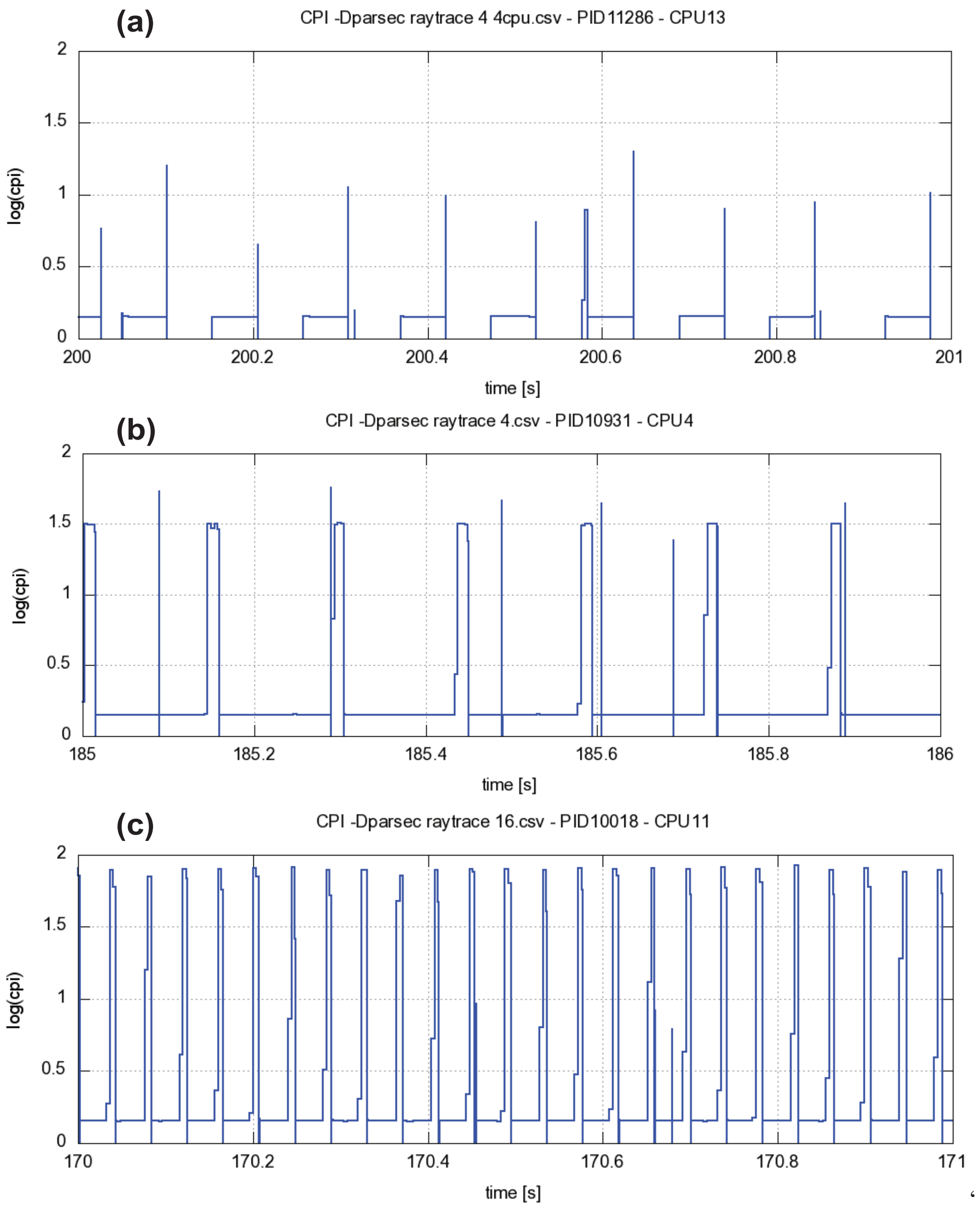


Figure 5.24: XTS CPI traces for parsec raytrace benchmark under different thread allocation and parallelization

states and the dependency on the workload. The goal of this model is to derive the overall power consumption of the CPU from high level information. The model must be suitable to be used inside an instruction set simulator. We generate it from a set of tests and power measurements made on real hardware (server platform) to derive an analytical model that fits real hardware behavior.

5.4.1 Target Platform and Measurement Set-up

We calibrate the power consumption model and the thermal profile based on measurements on an Intel® server system S7000FC4UR. It runs four quad-core Xeon® X7350 processors at 2.93GHz and has a total memory capacity of 16GB based on FBDIMMs. The Xeon® X7350 consists of two dual-core Core™2 architecture dies in a single package. Each of the two dual cores share a common 4MB sized L2 cache.

We connect the platform wall power supply to a digital multimeter[1]. We use it to measure the full platform power consumption. We then correlate it to both the temperature of each core and the clock per instruction of the task in execution. We take advantage of the per-core internal temperature sensors and performance counters available on the platform to extract this information.

5.4.2 Power Characterization and Modelling

To characterize the power profile of the target platform we perform the following three sets of tests. The first set is focused on highlighting the power consumption for corner cases, whereas the second set is to understand how voltage and frequency differently account to the whole power. The third one is instead to characterize the relation between the core power consumption and the workload at different performance levels[2].

5.4.2.1 Power corner cases

The first test runs a power virus[3] on each core of the platform while changing the performance state. To extract only the dynamic power contribution we subtract the idle power from the

[1]Agilent 34401A with additional shunt resistor.

[2]P-states: different CPU dynamic voltage and frequency states.

[3]*cpuburn* power virus by Robert Redelmeier: it takes advantage of the superscalar architecture to maximize the CPU power consumption.

maximum power as shown in eq. 5.1.

$$P_{dynamic} = P_{MAX} - P_{IDLE} \tag{5.1}$$

From the following equation (eq. 5.2) we can correlate the thermal design power(12) (TDP) specification with the dynamic power to extract the static power value.

$$P_{TDP} = P_{dynamic} + P_{static} \tag{5.2}$$

Table5.1 shows the results of these tests, i.e., the single core dynamic and static power consumption.

Table 5.1: Corner cases: full system vs per core power

Power virus	850W	Idle	451W
$P_{dynamic}$	22.5W (1 core)	P_{static}	9.7 (1 core)

5.4.2.2 DVFS vs DFS (No Voltage Scaling)

To better understand where the highlighted savings are mainly due to both voltage and frequency scaling or to frequency scaling only, we carried out a set of tests that try to force the CPUs to switch to different P-states without scaling the voltage. We run the following two tests sequentially while changing the frequency:

1. All CPUs are forced to run a power virus that retires two instructions per cycle.

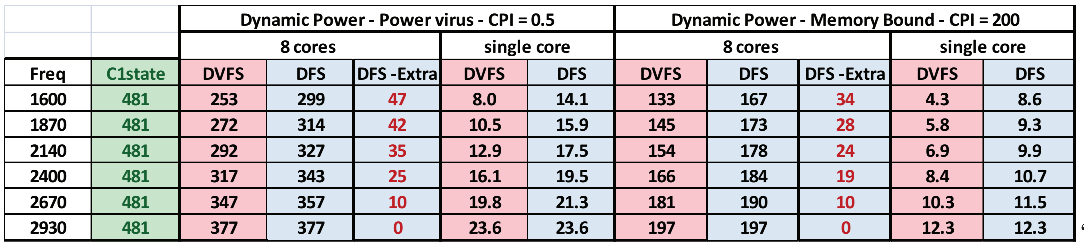
2. All CPUs are forced to run a memory bound synthetic benchmark that retires one instruction every 200 cycles.

In order to force some CPUs to scale only the frequency while keeping the voltage at the maximum level we take advantage of a limitation of the frequency and voltage control HW mechanism. In fact we discovered that the Xeon® 7350 allows only setting cores in the same die at the same frequency and all the cores in one package at the same voltage. This means, if the CPUs on two dies inside the same package run at different frequencies, the voltage level required to sustain the maximum frequency is applied to all the cores in the package. We then changed the frequency for test 1 and 2 as follows:

1. We scale the frequency only in one die per package, while keeping the other die at maximum frequency. This forces 8 CPUs to be at maximum frequency and maximum voltage while the others scale the frequency and keep the voltage at the maximum level (DFS) per package.

2. We scale the frequency for both the dies in the package for two packages only, while keeping the remaining two packages at the maximum frequency and voltage. This forces 8 CPUs to be at maximum frequency and maximum voltage while the others scale both frequency and voltage (DVFS).

Column ”8 cores” in Table 5.25 shows the real measured power values in Watt. The columns ”singe core” contain the dynamic power for a single core obtained as interpolation of the measured values. We can notice that the voltage reduction is an important portion of the power savings achievable with DVFS techniques. The column DFS refers at the case where the frequency changes without imply a voltage reduction. Indeed Table 5.26 quantifies the impact of DVFS versus DFS only and shows that up to 10% of system savings is due to the voltage reductions. From the ”Dynamic power reduction ratio” columns we can see that the dynamic power reduction ratio is super-linear on the frequency scaling factor for DVFS (for a decrease in frequency by 1.83, the dynamic power reduces by 2.95 and 2.86 for power virus and a memory bound benchmark respectively), whereas for DFS the decrease is sub-linear only (for a decrease in frequency by 1.83, the dynamic power reduces by 1.68 and 1.43 respectively).

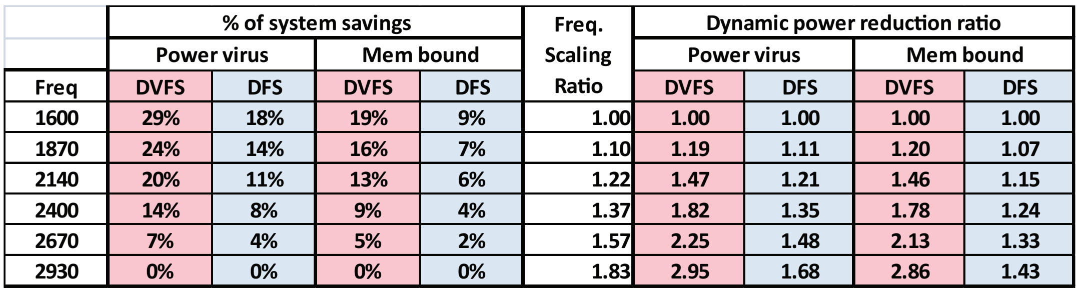


		Dynamic Power - Power virus - CPI = 0.5					Dynamic Power - Memory Bound - CPI = 200				
		8 cores			single core		8 cores			single core	
Freq	C1state	DVFS	DFS	DFS -Extra	DVFS	DFS	DVFS	DFS	DFS -Extra	DVFS	DFS
1600	481	253	299	47	8.0	14.1	133	167	34	4.3	8.6
1870	481	272	314	42	10.5	15.9	145	173	28	5.8	9.3
2140	481	292	327	35	12.9	17.5	154	178	24	6.9	9.9
2400	481	317	343	25	16.1	19.5	166	184	19	8.4	10.7
2670	481	347	357	10	19.8	21.3	181	190	10	10.3	11.5
2930	481	377	377	0	23.6	23.6	197	197	0	12.3	12.3

Figure 5.25: Dynamic power [Watt] - DVFS vs. DFS

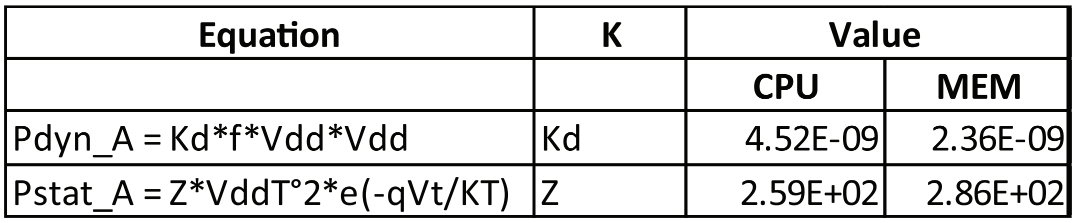
5.4.2.3 Calibrated power model

We use the results of the previous tests within the dynamic power fitting constants of a simple analitical model as shown by the following table. We can now build the following Table 5.27 to interpolate the dynamic power values for different voltages and frequency levels.



	% of system savings				Freq. Scaling Ratio	Dynamic power reduction ratio			
	Power virus		Mem bound			Power virus		Mem bound	
Freq	DVFS	DFS	DVFS	DFS		DVFS	DFS	DVFS	DFS
1600	29%	18%	19%	9%	1.00	1.00	1.00	1.00	1.00
1870	24%	14%	16%	7%	1.10	1.19	1.11	1.20	1.07
2140	20%	11%	13%	6%	1.22	1.47	1.21	1.46	1.15
2400	14%	8%	9%	4%	1.37	1.82	1.35	1.78	1.24
2670	7%	4%	5%	2%	1.57	2.25	1.48	2.13	1.33
2930	0%	0%	0%	0%	1.83	2.95	1.68	2.86	1.43

Figure 5.26: DVFS vs. DFS: percentage of system savings and reduction ratio



Equation	K	Value	
		CPU	MEM
Pdyn_A = Kd*f*Vdd*Vdd	Kd	4.52E-09	2.36E-09
Pstat_A = Z*VddT°2*e(-qVt/KT)	Z	2.59E+02	2.86E+02

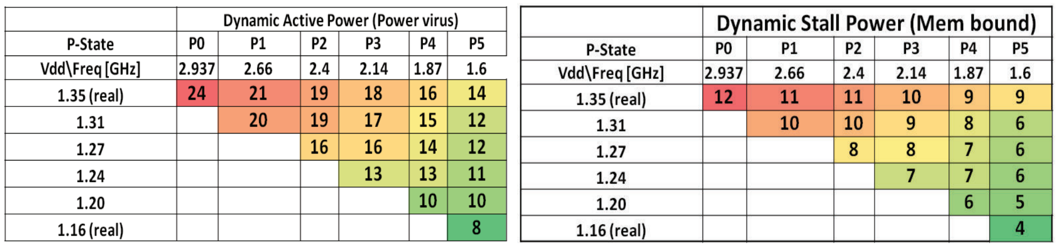
We can now combine it with the TDP information that infers the Static Power value.

$$P_{sta} = P_{TDP} - P_{dynmax} = 8.5W \tag{5.3}$$

This calibration stages highlight a bigger variability of the dynamic power due to a voltage and frequency changes (DVFS). This leads to a higher impact of DVFS policy on changing the dynamic power consumption.

5.4.2.4 Power consumption vs. CPI

Modeling the power consumption only at the corner cases is not enough for accurate describing the relationship between program phases and core power dissipation. Indeed in the third test set we execute a series of synthetic benchmarks with different memory pattern accesses on all



	Dynamic Active Power (Power virus)					
P-State	P0	P1	P2	P3	P4	P5
Vdd\Freq [GHz]	2.937	2.66	2.4	2.14	1.87	1.6
1.35 (real)	24	21	19	18	16	14
1.31		20	19	17	15	12
1.27			16	16	14	12
1.24				13	13	11
1.20					10	10
1.16 (real)						8

	Dynamic Stall Power (Mem bound)					
P-State	P0	P1	P2	P3	P4	P5
Vdd\Freq [GHz]	2.937	2.66	2.4	2.14	1.87	1.6
1.35 (real)	12	11	11	10	9	9
1.31		10	10	9	8	6
1.27			8	8	7	6
1.24				7	7	6
1.20					6	5
1.16 (real)						4

Figure 5.27: Dynamic - active / stall - power at different P-states

the processors. We repeat the tests while forcing all the cores to run at different performance levels. For each benchmark we extract the clocks per instruction metrics (CPI) and correlate it with the power consumption.

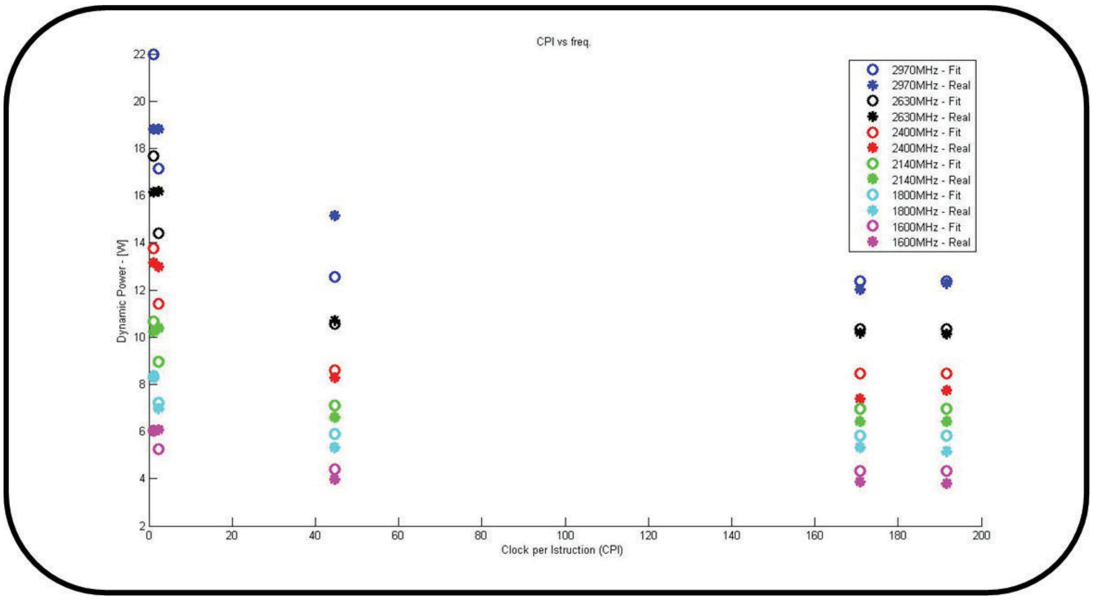


Figure 5.28: Per-core Power Based on Activity

Fig. 5.28 shows are previous model performs with different task cpi values. The model used is the follow:

$$P_{dyn} = P_{dyn-ACTIVE} \cdot \frac{1}{CPI} \cdot f_{MAX} + P_{dyn-STALL} \cdot \frac{CPI-1}{CPI} \cdot f_{MAX} \tag{5.4}$$

We can notice that it is accurate at for high CPI values but lost percision for low CPI values. To improve it we change approach and we looked for an analytical fitting model. Fig. 5.29 shows the CPI on the x-axis, and the model output power (with solid lines) on the y-axis, as well as the real measurement values with dots. Different plots refer to different performance levels. The curve highlights a power relation between CPI and power, which we exploit to fit them with eq. 5.5. The fit results are displayed in the figure with solid lines. The constants are reported in Table 5.2.

$$P_{dynamic} = k_A freq * V_{DD}^2 + k_B + (k_C + k_D freq) * CPI^{k_E} \tag{5.5}$$

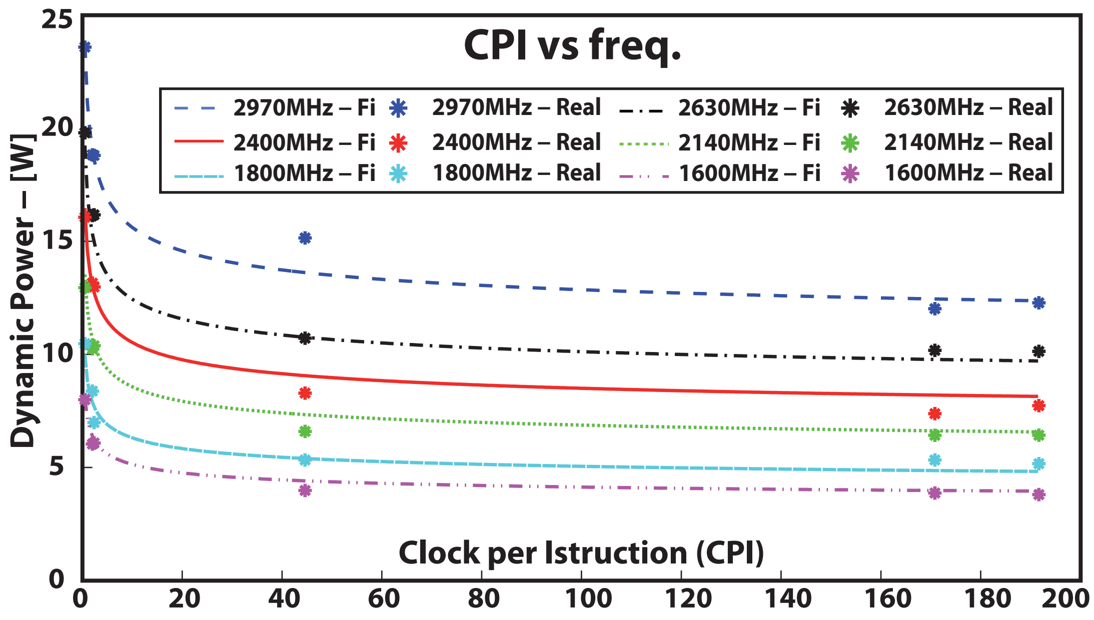


Figure 5.29: Per-core Power Based on Activity

We then relate the static power with the operating point by using a simple analytical model, described by eq. 5.6.

$$P_{static} = Z * V_{DD} * T^2 e^{\frac{-qVt}{K*T}} \tag{5.6}$$

These two models allow us to relate the power consumption with the workload (CPI), the clock frequency, the voltage supply and the temperature. Indeed, from fig. 5.29 we can notice the accuracy of our model (solid line) in estimating the core power consumption, given the CPI of the task in execution and the clock frequency of the core. We use these models to simulate the power of the target cores in our virtual platform.

Table 5.2: Power Model Fit Constants

k_A	2.13e-3	k_B	-1.45
k_C	-4.1376	k_D	0.0051
k_E	-0.3016	Z	2.59E+02
K	1.38E-23	q	1.60E-19

5.4.3 Thermal Characterization and Modelling

To characterize the power and thermal profile of the target platform we perform the following test. We use a CPU-bound workload, which runs on each core of the package one by one. As workload, we again use the power virus. When the power virus starts the execution the scheduler must switch out the idle task, forcing the core to exit from a low-power idle state. Consequently, the power consumption increases accordingly with the new workload. We combine that with monitoring the thermal response of each core in one package by reading out the internal temperature sensors. Fig. 5.30 shows the results of these tests for one core. The dashed line shows the temperature measurements, whereas the dash&dot curve shows the input power step.

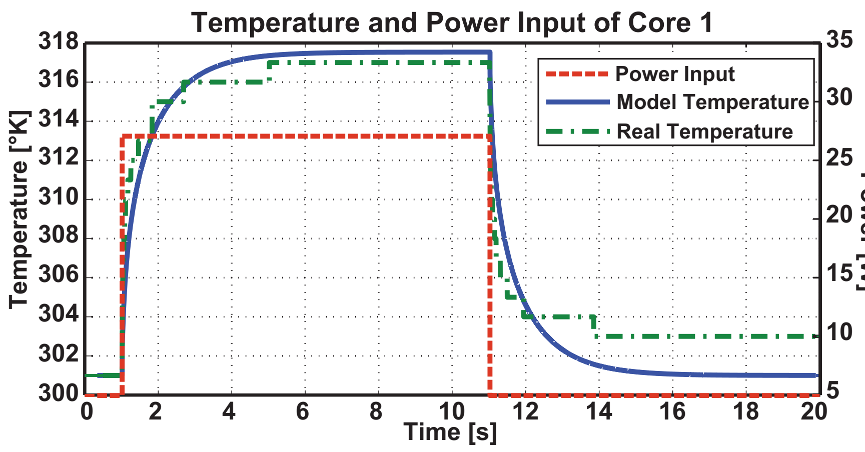
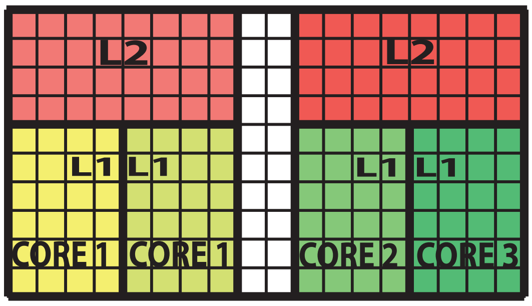
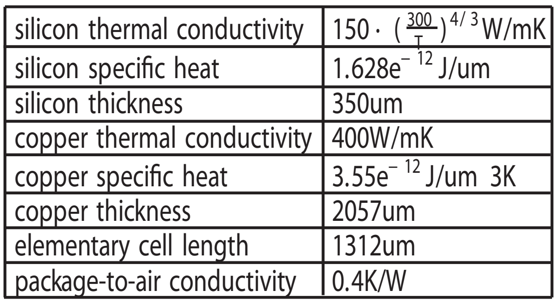


Figure 5.30: Thermal Response Comparison

We use these results in combination with the layout of the Xeon® X7350(9) to empirically fit the parameters of the temperature model used in our virtual platform presented in chapter6. In fig. 5.31 the floorplan and parameters used are shown. The output of the thermal model for the same input (power step) is shown in fig. 5.30 with a solid line.

We can notice that our model perfectly tracks the real thermal transient: it responds with the same dynamics and finds the asymptotic temperature value with less than 1% error.





silicon thermal conductivity	$150 \cdot (\frac{300}{T})^{4/3}$ W/mK
silicon specific heat	$1.628e^{-12}$ J/um
silicon thickness	350um
copper thermal conductivity	400W/mK
copper specific heat	$3.55e^{-12}$ J/um 3K
copper thickness	2057um
elementary cell length	1312um
package-to-air conductivity	0.4K/W

Figure 5.31: Approx. Intel® Xeon® X7350 Floorplan

Bibliography

[1] K. Flautner , T. Mudge, Vertigo: automatic performance-setting for Linux, Symposium on Operating systems design and implementation, 2002. ii, 63, 64, 78

[2] G. Dhiman, T. Simunic Rosing Dynamic voltage frequency scaling for multi-tasking systems using online learning, ACM International symposium on Low power electronics and design, 2007. ii, 63, 65, 79

[3] W.-Y. Liang, S.-C. Chen, Y.-L. Chang, and J.-P. Fang, Memory-aware dynamic voltage and frequency prediction for portable devices, in Proc. RTCSA, 2008, pp. 229236. ii, 63, 68, 79

[4] Z. Wang, X. Zhu, C. McCarthy, P. Ranganathan, and V. Talwar, Feedback Control Algorithms for Power Management of Servers, *FeBid*, Annapolis, MD, June 2008. iii, 63, 69, 79, 80

[5] M. Kondo , H. Sasaki , H. Nakamura, Improving fairness, throughput and energy-efficiency on a chip multiprocessor through DVFS, ACM SIGARCH Computer Architecture News, v.35 n.1, March 2007. iii, 63, 69, 79

[6] K. Skadron, T. Abdelzaher, M. R. Stan, Control-theoretic techniques and thermal-RC modelling for accurate and localized dynamic thermal management. IEEE High-Performance Computing Architecture, 2002. iii, 63, 72, 79

[7] A. Kivilcim Coskun , T. Simunic Rosing , K. C. Gross, Proactive temperature balancing for low cost thermal management in MPSoCs, IEEE/ACM International Conference on Computer-Aided Design, November 2008. iii, 63, 74, 79, 80

[8] M. Kadin, S.Reda, A. Uht. Central vs. distributed dynamic thermal management for multi-core processors: which one is better?. ACM Great Lakes symposium on VLSI, 2009. iii, 64, 77, 79

[9] N. Sakran et al. The implementation of the 65nm dual-core 64b merom processor. In IEEE International Solid-State Circuits Conference, 2007. 80, 103

[10] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer's Manual - Volume 3B, June 2009. 80, 81

[11] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", *PACT*, 2008. 88, 89

[12] Intel Corporation. Intel® Xeon® Processor 7200 Series and 7300 Series Datasheet-Datasheet, September 2008. 98

Chapter 6

A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-performance Multicores

6.1 Overview

As introduced in previous chapter the use of high-end multicore processors today can incur high power density with significant variability in spatial and temporal usage of resources by workloads. This situation leads to power and temperature hotspots, which in turn may lead to non-uniform ageing and accelerated chip failure. These drawbacks can be mitigated by online tuning of system performance and adopting closed-loop thermal and reliability management policies. The development and evaluation of these policies cannot be performed solely on real hardware – due to observability and flexibility limitations – or just by relying on trace-driven simulation, due to dependencies present among power, thermal effects, reliability and performance. We present a complete and virtual platform to develop, simulate and evaluate power, temperature and reliability management control strategies for high-performance multicores. The accuracy and effectiveness of our solution are ensured by integrating a established system simulator (Simics) with models for power consumption, temperature distribution and aging.

The models are based on characterization on real hardware. Control strategies exploration and design are carried out in the MATLAB/Simulink framework allowing the use of control theory tools. Fast prototyping is achieved by developing a suitable interface between Simics and MATLAB/Simulink, enabling co-simulation of hardware platforms and controllers.

6.2 Introduction

Pushed by Moore's law and the demand for better performance, processors have become high-performance multi-cores, characterized by high power density and total power consumption. Due to physical limitations this leads to complex and expensive thermal dissipation solutions(18). In addition, significant spatial and temporal variability of resource usage by workloads is the source for non uniform performance, power consumption and temperature distributions(6). On-die hot spots suffer from larger static power, due to the exponential dependency of leakage current on temperature. Moreover, hot spot areas age faster since degradation effects, such as NTBI, and HCI (14), are exponentially accelerated by high temperature. All these drawbacks can be mitigated on-line by tuning system performance and temperature through closed-loop control management policies.

Today processors include generic support for these techniques, based on introspective monitors (15), sensors and performance knobs. The former allow monitoring the system status (clock frequency, temperature, supply voltage, body bias voltage) and the workload requirements (clocks per instruction, memory accesses, L1/L2 miss rates), whereas the latter allow modifying system performance (by setting clock frequency, voltage supply, power gating). This infrastructure provides the sensors and the actuators for feedback control management policies. For instance, (11) shows the effectiveness of dynamic and voltage scaling, as well as task migration versus simple stop-and-go policies to mitigate hot spots and decrease thermal run-out occurrences. (14) shows the efficacy of FBB and voltage scaling to reduce aging effects. These policies can be implemented at different levels: within operating system modules (17) or with dedicated hardware blocks(19). The former approach is more flexible, but the latter is more efficient and can react to faster dynamics.

System simulators, also called virtual platforms, are required to support strategic decision making on the allocation and complexity of hardware monitors and knobs, as well as on the development of software (or hardware-software) policies for thermal and reliability management. Hardware prototypes and test-chips are expensive, require high effort and usually provide limited introspection. For instance, the complete spatial temperature map cannot be obtained easily without invasive and expensive infrastructure(5) and, even worse, an incorrect control policy can lead to permanent HW damage.

Simulators and virtual platforms must be capable to first accurately simulate the entire system evolution (program flow, data coherency conflicts, complex memory latency)(21), (8),(10) and emulate the same performance knobs and introspective sensors mechanism of the real platform(15). Secondly they must co-simulate the physical effects we want the controller to be developed for with high degree of fidelity. Finally, they have to allow the co-simulation of control strategies acting on the performance knobs and depending on the physical behavior (usually, the thermal behavior) measured by means of suitable sensors of the virtual physical model. The co-simulation is a key feature in the above considerations, since evolution of digital and physical effects of the platform are fed back and interconnected through the controllers. Therefore, trace-driven cascade simulation loses information on cross-dependencies, resulting in degraded simulation accuracy(14),(11).

6.2.1 Related Work

A large variety of simulators can be found(21),(10),(3),(2), (1). Cycle accurate simulators on one hand show higher modelling accuracy(3),(2) but on the other hand are too costly in terms of simulation time when simulating a complete platform. Differently, functional and instruction set simulators(21),(10),(1), at the cost of less internal precision, allow full system simulation in a reasonable time. This aspect is a key feature to avoid unrealistic simulation artefacts and to evaluate a control solution that fits in the system. While cycle accurate simulation output has been used in combination with well-established power and temperature simulators based on analytical models that require microarchitectural knowledge of the system(4),(12),(7), adopting high-level simulators introduces the challenge of having accurate power and temperature physical models. Indeed, moving from micro-architectural level to whole system simulation prevents the use of micro-architectural analytical models, due to less detailed input data.

6.2.2 Contributions

Bearing in mind the above considerations, in this chapter we present a complete virtual platform with the following key features:

- High-level functional modelling, where corresponding power and thermal models are included, allowing fast, but "physically accurate" simulations of the complete multicore platform.

- Integration with a suitable framework to perform co-simulation of multicore SoCs with control algorithms for rapid prototyping and to achieve flexibility and effectiveness in the design, testing and review of control strategies, exploiting control-theory tools.

The proposed solution is based on Simics[0](21) and Matlab[0]/Simulink[0](20). Simics is a commercial instruction set simulator that models a complete multicore platform based on in-order x86 cores with memory, I/O interfaces and OS. MATLAB/Simulink is a tool for matrix-oriented numerical computation and simulation, where many specific toolboxes supporting control system design are available. Three main issues have been tackled in developing the proposed virtual platform.

- Identification, from experiments on a real general-purpose multicore platform, of accurate, but component-oriented, power and thermal models. Run-time program flow statistics are the inputs of models developed to derive power consumption and the temperature evolution of each system functional unit.
- Interfacing of the above-mentioned models with Simics simulation engine.
- Development of a co-simulation bridge between Simics and MATLAB/Simulink.

The chapter is organized as follows. In section 6.3 the proposed virtual platform is presented. Particular attention is devoted to multicore platform characterization to derive suitable power and thermal models and to the Simics-Matlab/Simulink interfacing at simulation time. In section 6.4 a development cycle for thermal control strategies is proposed taking advantage of the virtual platform, while in section 6.5 a case study is presented. Final conclusions are drawn in section 6.6.

6.3 Virtual Platform

Fig. 6.1 depicts the block diagram of the platform that combines a full-system multicore simulator (Simics, see section 6.3.1) with a control system development environment (MATLAB/Simulink, see section 6.3.2) through an ad-hoc interface (see section 6.3.3).

[0]Other brands and names are the property of their respective owners.

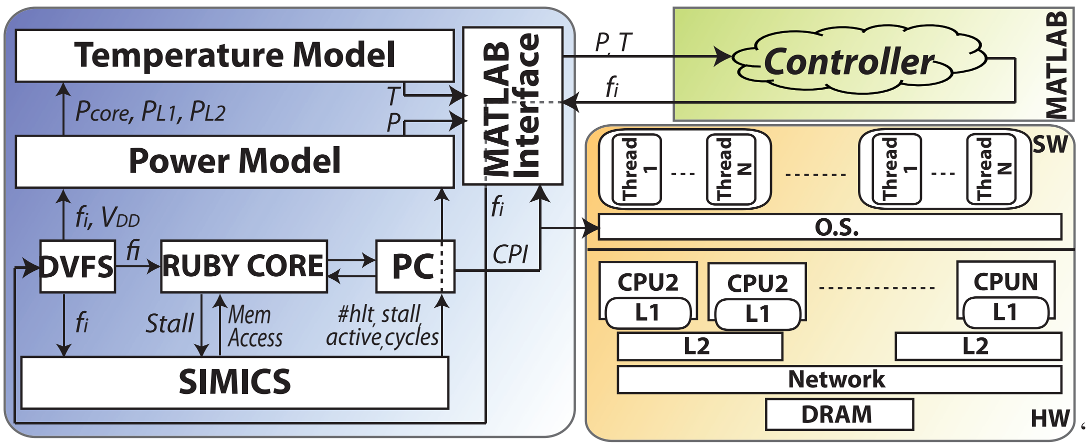


Figure 6.1: Virtual Platform Overview

6.3.1 Simics

Simics^0, by Virtutech(21), is a full system functional simulator. It models an entire system, with peripherals, BIOS, network interfaces, cores, memories, etc. Moreover it allows the target system to boot a full OS, such as Linux^0 and Windows^0. Simics can be configured with different predefined target CPU architecture modules (arm, sparc, x86, ...) and it can simulate a multicore target environment. Linux SMP is supported, which enables to boot up to 256 cores.

In our tests we configure Simics to emulate the X7350 Intel® core, with four Pentium®4 cores as proxy. Simics simulates each x86 instruction in one CPU clock time period, without accounting for memory latency, and different execution times for different instructions.

6.3.1.1 GEMS

To overcome this limitation, we use a public cycle-accurate memory timing-model, named GEMS(8). It consists of several modules, but in our platform we configured it to use only the memory model, named RUBY. This module provides a detailed memory system simulator. It can simulate a wide variety of memory hierarchies and systems ranging from broadcast-based SMP to a hierarchical directory-based multiple-CMP system.

RUBY is loadable by Simics at system configuration time. During simulation RUBY is called from each core before executing each memory access. As a result, RUBY stalls the target processor, decodes the address, determines the memory location inside the memory hierarchy, determines the latency and simulates the timing and race behavior by posting the event inside a global timing event queue (EQ). The EQ is ordered and indexed by the internal RUBY cycle.

As it is implemented, one RUBY cycle is one Simics CPU0[1] cycle. RUBY scans the EQ cycle-by-cycle. When it finds the conclusion of a memory transfer it un-stalls the specific processor models.

Since RUBY is written in C++ and fully integrated with Simics, we use it as skeleton in which we apply our add-ons. These are added by expanding RUBY with new modules (C++ object).

6.3.1.2 Dynamic Voltage Frequency Scaling Module

Simics allows each core in the system to run at its own frequency and to change the value at run-time. However, RUBY does not offer this flexibility: it forces CPU0 to be constantly bound at the smallest clock period among all the cores and does not have internal knowledge of the core frequencies. To support dynamic frequency scaling while preserving correct memory timing, we first configure Simics with N+1 cores to simulate a target platform with N cores. Secondly we logically turn-off[2] the N+1 core from the linux kernel. Thus the N+1 core is a dummy and will not affect the simulated program flow. Finally we modify RUBY to be synchronized with the N+1 CPU instead of CPU0 and we bind its frequency to be constantly at the maximum value allowed by the system. This ensures the L2 cache and DRAM to have a constant clock frequency, with value proportional to CPU N+1 frequency. Then we modify, internally in RUBY, the L1 latency value computation to be dependent on the Simics processor frequency clock[3]. These modifications are combined with a frequency controller that stores the frequency settings and dispatches them to other blocks inside a new RUBY module, named DVFS in fig. 6.1.

6.3.1.3 Performance Counters Module

Since a performance control policy relies on software access to system performance counters and sensors, we introduce a new module in RUBY to model this interaction. This module is time-triggered to execute each N RUBY cycles, where N is a user selected simulation parameter. Each time it is woken up, it accesses Simics-internal registers and extracts the following information: number of instruction retired, clock cycles and stall cycles expired, halt instructions and other core events. These values are then stored in an internal, per-core data structure

[1]First CPU available in the simulated multicore.

[2]By using the CPU hotplug Support.

[3]Assuming L1 and core in the same clock domain.

and made visible directly in the target address space. This is done during simulation by explicitly copying the data structure after every update of the performance counters module to a target system-reserved memory area. Simics magic-instructions(21) are used to perform this.

6.3.1.4 Power Module

To simulate the power consumption of the target architecture at run-time we integrate the virtual platform with an experimentally calibrated analytical power model. This power model takes the core, cache and memory usage from the performance counters module as input and combines it with the operating conditions (core frequencies, Vdd and temperature) to provide the average power and energy dissipated as output. This model is implemented as a new RUBY module. It is self-timed by using the RUBY event queue functionality. Each time it is woken up, it computes the power and energy dissipated since the last invocation (for cores, L1, L2, DRAM).

Cores When the power module is woken up it checks the CPU usage by reading the performance counters of the target core. We use an empirically derived power model, described in section5.4.2 that fits the dynamic power consumption as function of the frequency and average CPI (clocks per instruction). As a consequence of that the power model internally extracts the CPI and the idleness for each core (as measured by percentage of HLT cycles[1]). The power module combines these values with the operating point (Vdd, frequency), input from the DVFS module, and accordingly with the power model 5.4.2 that computes the total power with eq. 6.1

$$\begin{aligned} P_{Total} &= [P_{dynamic}(f, CPI) + P_{static}(T, V_{DD})] * \\ &\quad *(1 - idleness) + idleness * (P_{idle}) \end{aligned} \tag{6.1}$$

Idleness is used to model the effects of deep sleep power states[2].

Memory We first use CACTI(13) to extract the power dissipated by each memory component (L1 and L2 caches, as well as DRAM) in different states: when it is accessed (Read & Write), when it is in stand-by mode, and (for the DRAM only) when it is refreshed. Then, we collect the memory usage from inside the RUBY memory model and multiply it with the power dissipated by each node to obtain the total power.

[1] HLT cycles happens as consequence of HLT instructions that are executed when the OS schedules the idle task and consequently goes into deep sleep power states.

[2]C-states in ACPI formalism.

6.3.1.5 Thermal Module

To simulate the effects of power consumption on the temperature distribution of the target multicore, we integrate our virtual platform with the thermal simulator presented in (12). As shown in fig. 6.1 the temperature model receives the power dissipated by the main functional units composing the target platform (cores, L1 and L2) as input and provides the temperature distribution along the simulated multicore die area as output. Internally, starting from a representation of the multicore floorplan, the thermal simulator spatially allocates the input functional units energy, dissipated in the last time quantum (ΔT), over the floorplan. Then the entire surface of the die is split in small square blocks. Each block models a heat source ($E(i,j)$) and is characterized by an intrinsic temperature ($T(i,j)_t$). This models the bottom surface and the injection of heat in the target multicore package. In addition, the package volume is partitioned in cubic cells. Each cell is modelled with the equivalent thermal capacitance and resistance ($R(i,j), C(i,j)$) and connected with the adjacent ones. The top surface models the heat spreader with an equivalent thermal resistance. At each simulation step the R,C thermal-equivalent differential problem is solved providing the new temperature value for each cell ($T(i,j)_{t+\Delta T}$) as output.

6.3.1.6 Simulation Performance

We test the presented virtual platform to extract the simulation time overhead introduced by our modifications. Table 6.1 shows host simulation time required by the virtual platform to execute 1 billion instructions under different simulator configurations. The power model introduces an overhead of less than 7% on Simics and GEMS, whereas with the temperature model the overhead is 19.2%.

Table 6.1: Virtual platform overhead (power model updated every 13us, temperature model - 68 cells, integration step=100ns)

Target	Host	Module	Sim Time
4 core Pentium® 4	Intel® Core™ 2 Duo	Simics + Gems	1040s
2GB DRAM	2.4 Ghz	..+ DVFS	1045s
32 KB private L1	2GB RAM	..+ Power	1110s
4 MB shared L2		..+ Thermal Interface	1160s
$1*10^{12}$ instr.-0.5 s		..+ Thermal Simulator	1240s

6.3.2 MATLAB

MATLAB^0(20) is a numerical computing environment developed to design, implement and test numerical algorithms. Simulink is a graphical environment that extends MATLAB for simulation of dynamic systems. The main core of MATLAB/Simulink environment is complemented by toolboxes, among them the Control System Toolbox. This toolbox simplifies and speedups the development cycle of control systems. It allows the use of tools from control theory and simulating tests at early stage. This avoids most of the time-consuming trial-and-error steps often performed directly in physical systems following basic approaches to control system design. The MATLAB/Simulink platform is open for integration and data exchange with other domain-specific tools for modeling and simulation. MATLAB does not only execute as stand-alone program, but can be called as a computational engine by writing C and Fortran programs that use MATLAB's engine library. The engine library is the link between a C program and a separate MATLAB engine process.

6.3.3 Simics - MATLAB Interface

Performance controller design can be described by two intrinsic difficulties: developing the control algorithm that optimizes the system performance and implementing it in the system. In order to let the designer focus only on the first problem without loosing the capability of testing the solution in a real system, we allow a MATLAB/Simulink description of the controller (SC) to directly drive the performance knobs of the emulated system. This is done by combining the flexibility of RUBY in adding new C++ modules supporting the MATLAB engine library. For this, we introduce a new module named Controller (RC) in RUBY. This module is self-timed in the simulation domain by using the RUBY event queue timing capabilities. At initialization time it first starts the MATLAB engine process that executes concurrently to the simulator. Then, the MATLAB environment is initialized. During this stage the Simulink controller model (SC) is loaded and initialized. Then, two communication channels are established between the RUBY controller module (RC) and the Simulink controller model (SC). The forward communication ($RC \rightarrow SC$) is used for providing input to the controller algorithm (SC). The output of SC is converted for the target simulated system by using the backward communication channel ($SC \rightarrow RC$).

At every refresh, with period ΔT, the RUBY controller module (RC) executes the following steps:

1. It triggers the Simulink controller model (*SC*) to load the initial status vector from a private variable.
2. It loads the target system state from the performance counters, DVFS, power and temperature modules and copy it to the MATLAB environment.
3. It triggers one step of Simulink simulation for the controller (*SC*) of length ΔT.
4. It waits and receives the last output vector (new cores frequencies) from the Simulink controller model (*SC*) and sends it to the DVFS module.
5. It triggers the Simulink controller (*SC*) to save the last internal status vector in the initial status private variable.

6.4 Control-strategies Development Cycle

Taking advantage of the features of MATLAB/Simulink and Simics, we can set up an integrated platform and define a model-based development cycle for control systems embedded in System-on-Chip as multicores. The basic idea of the proposed development cycle is reported in the following and briefly sketched in fig. 6.2.

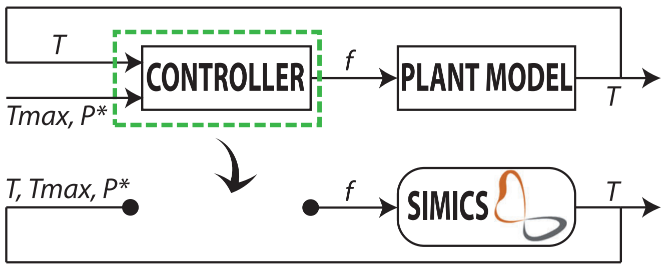


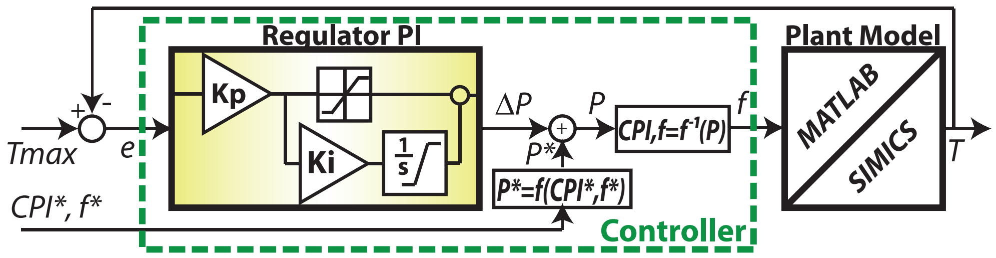
Figure 6.2: Thermal Control Design Process

As first step, the controller design is carried out in the MATLAB/Simulink framework. The system to be controlled (”the plant” in control terms) is represented by a simplified model in this framework (the so-called ”control model”) obtained by physical considerations and identification techniques. A first set of simulation tests and design adjustments can be done in Simulink. As a second step, the tuned controller is directly interfaced with the accurate model of the plant (the so-called ”evaluation model”) represented in Simics exploiting the MATLAB/Simulink interfacing features. Hence, accurate simulation of the overall system is performed for performance testing. Depending on the results, a design loop back to the MATLAB/Simulink framework can be chosen to exploit its flexibility and simplicity in controller re-parameterization and re-design. The key point of this method is the integration between Simics and Simulink in simulating the overall system.

6.5 Case Study

To illustrate the use of the virtual platform for designing, analyzing and testing control strategies a case study of a complete system with running OS is presented. We consider the multicore system shown in fig. 5.31. The temperature threshold is set to 330°K. Each core consumes a maximum power of 30W; the chip has an area of 300mm^2. The other elements of the multicore consume 30% of the power consumption of the processing cores. Our target is to realize a thermal controller that avoids violation of temperature bounds. The system is composed by four cores and two memory slots. There are four temperature sensors, one for each core. This fact is very important because it limits the information that the controller can use (e.g., the controller does not know what the memory contributes to the temperature of the cores).
The first step is to model the system in Simulink. It could be done by dividing the system in small cells and associating an equivalent electric circuit with them. The thermal energy exchange occurs through resistances and capacities. In this phase it is not necessary to build up an accurate model for controller implementation. For this reason it is feasible to consider only one cell for every core and two cells for every memory.
The model receives the core frequencies (f) and the CPI from the regulator and returns the core temperature (T). Regarding the control strategy a distributed approach is used: every core has an assigned PI regulator. A PI regulator generally consists of two independent terms: the proportional term (P) and the integral term (I). The regulator output is $Kp \cdot e(t) + Ki \cdot \int e(\tau)d\tau$ where e(t) is the error value (the temperature error in this case). Every PI regulator has the structure shown in fig. 6.3.

Tuning the regulators is simple because we can use the functions and tools provided by MATLAB/Simulink. For example, it is possible to find the K_p and K_i parameters imposing a phase margin and a crossing frequency in the Bode diagram.
In fig. 6.4 the regulated temperatures of the cores are shown when a constant power of 9W is assumed at every core. At a given instant, a step of 30W is applied to core 0. Normally, This



	Cntr 0	Cntr 1	Cntr 2	Cntr 3
k_p	13.9439	13.7875	13.7875	13.9439
k_i	0.9840	0.9840	0.9840	0.9840

Figure 6.3: PI Controller

would lead the core to overshoot the temperature threshold. However, the controller reacts to bring the core's temperature back under the limit.

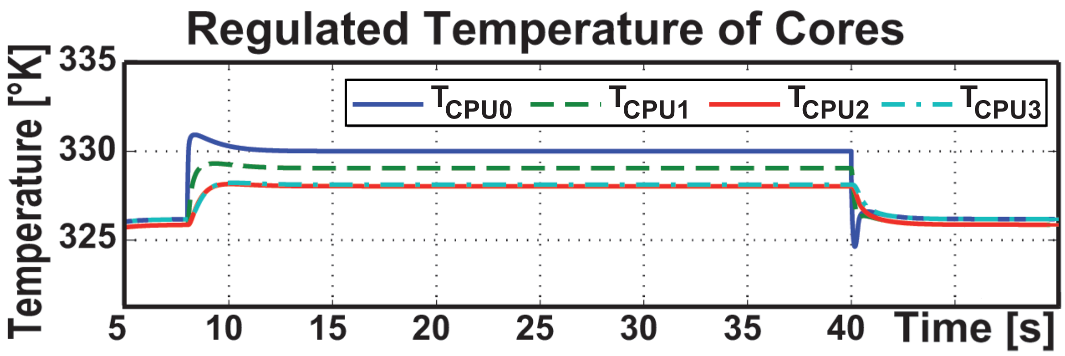


Figure 6.4: MATLAB: Temperature Step Responce of the Cores over Time

After designing the controller it is possible to test the controller on the virtual platform. In fig. 6.5 we can see the thermal performance of the four cores under the supervision of the distributed controller. Cores 0, 2 and 3 execute three tasks in parallel that contain three different sections each: the first phase has low CPI (CPU bounded), the second one has high CPI (memory bounded) and the third one executes an idle loop (sleep). It can be noticed that these tasks cause the temperature limit to exceed within the first two phases forcing the controller to reduce the frequency, whereas within the third phase the power consumption decreases. CPUs are able to cool down allowing the controller to increase the frequencies in that phase.

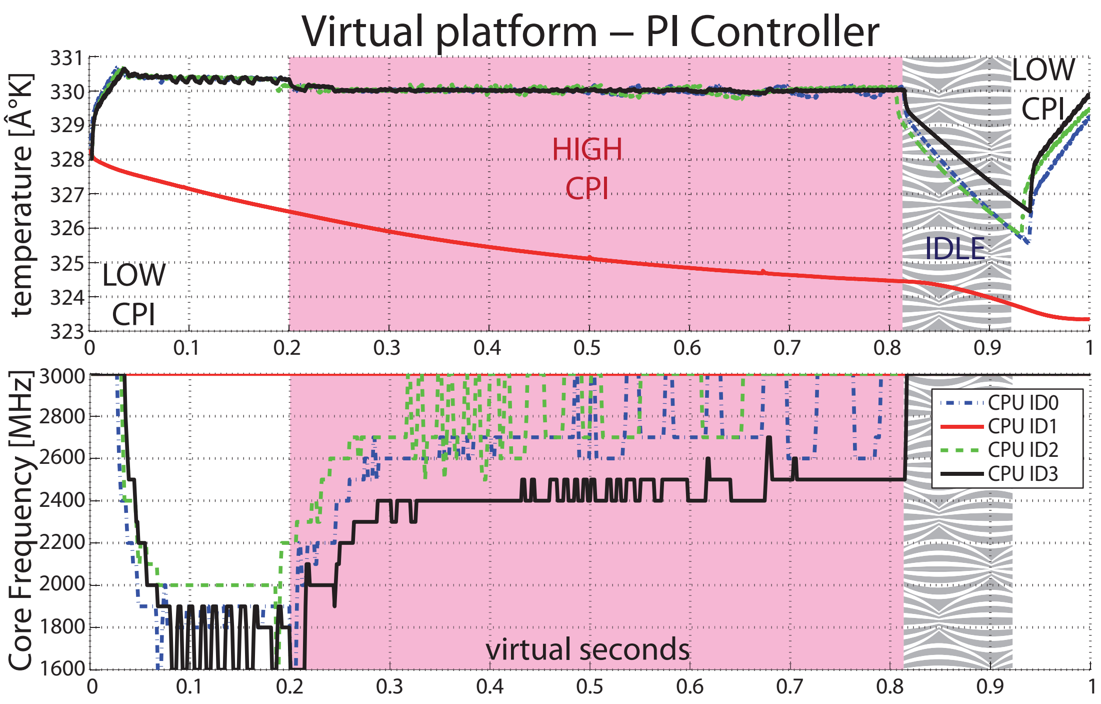


Figure 6.5: Simulation: Frequency/Temperature Responce of the Cores over Time

6.6 Conclusions

We have presented a novel virtual platform for efficiently designing, evaluating and testing power, thermal and reliability closed-loop resource management solutions based on control theory. Firstly, a modular high-level simulation platform for full multicore systems has been created. Instruction set emulation, spatial temperature distribution and power dissipation models, performance knobs (DVFS, deep sleep states) and monitors (defined according to real platforms) have been defined and composed accordingly. Models for power dissipation and thermal dynamics, compatible with high-level instruction set emulation, have been derived from real hardware characterization. Secondly, the design procedure of the performance controller and the simulation capability have been enhanced by allowing a MATLAB/Simulink description of the controller to execute natively as a new component of the virtual platform. The controller directly drives the performance knobs of the emulated system. This feature opens the door to an innovative controller development cycle that helps the developer in converging to an optimum "in the field" performance control solution quicky by exploiting powerful control-theory methodologies supported, as well as easy-to-implement, in the MATLAB/Simulink environment.

Bibliography

[1] Argollo E. et al. COTSon: Infrastructure for full system simulation. In ACM SIGOPS Operating System Reviews, Jan 2009 109

[2] Atienza D. et al. A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. Design Automation Conference (DAC), pages 618-623, 2006. 109

[3] Benini L. et al. MPARM: Exploring the multi-processor SoC design space with SystemC. The Journal of VLSI Signal Processing, 41:169-182, Sep. 2005. 109

[4] Brooks David et al. Wattch: a framework for architectural-level power analysis and optimizations. SIGARCH Comput. Archit. News, 28(2):83-94, 2000. 109

[5] Hamann H. F. et al. Hotspot-limited microprocessors: Direct temperature and power distribution measurements. IEEE Journal of Solid-State Circuits, 42:56-65, Jan. 2007. 109

[6] Hanson H. et al. Thermal response to DVFS: analysis with an Intel Pentium m. In ISLPED '07, pages 219-224, 2007. 108

[7] Huang Wei et al. Accurate, pre-RTL temperature-aware design using a parameterized, geometric thermal model. IEEE Trans. Comput., 57(9):1277-1288, 2008. 109

[8] Martin Milo M. K. et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Comput. Archit. News, 33(4):92-99, 2005. 109, 111

[9] N. Sakran et al. The implementation of the 65nm dual-core 64b merom processor. In IEEE International Solid-State Circuits Conference, 2007.

[10] Nathan L. Binkert et al. The m5 simulator: Modeling networked systems. IEEE Micro, 26:52-60, 2006. 109

[11] P Chaparro et al. Understanding the thermal implications of multi-core architectures. IEEE Transactions on Parallel and Distributed Systems,18(8):1055-1065, Aug. 2007. 108, 109

[12] Paci G. et al. Exploring ”temperature-aware” design in low-power MPSoCs. In DATE ’06, pages 838-843, 2006. 109, 114

[13] Thoziyoor Shyamkumar et al. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. ISCA ’08, pages 51-62, 2008. 113

[14] Tiwari A. et al. Facelift: Hiding and slowing down aging in multicores. MICRO ’08, pages 129-140, 2008. 108, 109

[15] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer’s Manual - Volume 3B, June 2009. 108, 109

[16] Intel Corporation. Intel® Xeon® Processor 7200 Series and 7300 Series Datasheet-Datasheet, September 2008.

[17] ACPI Advanced Configuration and Power Interface Specification http://www.Intel.com/products/processor/manuals/ 108

[18] IDC. Worldwide server power and cooling expense 2006, 2010 forecast. http://www.sun.com/service/eco/IDCWorldwideServerPowerConsumption.pdf. 108

[19] Intel Corporation. Intel Corporation. Intel® turbo boost technology in Intel® core™ microarchitecture (Nehalem) based processors. Technical report, 2008. 108

[20] The MathWorks. MATLAB & Simulink. http://www.mathworks.com/. 110, 115

[21] Virtutech. Virtutech Simics. http://www.virtutech.com/. 109, 110, 111, 113

Chapter 7

A Distributed and Self-Calibrating Model-Predictive Controller for Energy and Thermal management of High-Performance Multicores

7.1 Overview

High-end multicore processors are characterized by high power density with significant spatial and temporal variability. This leads to power and temperature hot-spots, which may cause non-uniform ageing and accelerated chip failure. These critical issues can be tackled on-line by closed-loop thermal and reliability management policies. Model predictive controllers (MPC) outperform classic feedback controllers since they are capable of minimizing a cost function while enforcing safe working temperature. Unfortunately basic MPC controllers rely on a-priori knowledge of multicore thermal model and their complexity exponentially grows with the number of controlled cores.

In this chapter we present a scalable, fully-distributed, energy-aware thermal management solution. The model-predictive controller complexity is drastically reduced by splitting it in a set of simpler interacting controllers, each allocated to a core in the system. Locally, each node selects the optimal frequency to meet temperature constraints while minimizing the performance penalty and system energy. Global optimality is achieved by letting controllers exchange a limited amount of information at run-time on a neighbourhood basis. We address model uncertainty by supporting learning of the thermal model with a novel distributed self-calibration approach that matches well the controller architecture.

7.1.1 Related Work

Power budgeting and power capping (6) techniques use built-in power meters as inputs to close-loop feedback controllers for constraining the power consumption to a given budget by reducing the cores clock frequencies. This approach has two main drawbacks: first, it relies on the availability of accurate and reliable power measurement infrastructures; second, in many cases it does not leads to a global energy reduction due to execution time overhead. Indeed, high power dissipation phases usually are correlated with CPU-bound computational ones. Thus power budgeting techniques happen to reduce the frequency mainly in this situation, leading to energy losses due to static power and to execution time linear dependency with frequency (7). Different solutions have been presented to achieve a system energy reduction instead of power capping, but unfortunately the energy minimization alone cannot enforce a safe working temperature (7).

Closed-loop thermal control policies aim to address this problem. In (8), (9), (10) the authors show the benefit of feedback-control approaches vs. open loop policies based on temperature thresholding heuristics. Model predictive controllers (MPC) (11) (6) outperform classic feedback controller, which cannot take into account hard constraints in the state space. MPC controllers instead rely on an system model (12) to predict the future temperature while finding the optimal control action by solving a constrained optimization problem *for one or more control steps in the future*. Thus they generally lead to higher-quality control, providing that an effective thermal model is available.

Wang et al. (6) present a MPC that constraints both the power and the temperature of the cores while maximizing the performance. It uses power sensor as input to the optimization problem and to generate on-line a power to frequency linear model. This reduces the complexity of the controller (even though it can lead to sub-optimal controller corrective actions). Zanini et al. (11) assume a workload input requirement, so that the MPC minimizes the performance loss in tracking it while constraining the core temperatures. Internally it adopts a non-linear frequency to power model, statically precomputed off-line avoiding usage of power sensors. Unfortunately the adopted model is simplistic and it does not consider the power dissipation dependency on workload properties, assuming it to be only related to core frequency.

Performance of MPC solutions strongly depends on the thermal model accuracy. Unfortunately, often an accurate model is not a-priori available. Different approaches have been proposed to identify it from HW introspective monitors. In (6) a first order thermal model is estimated using off-line least square method. Differently, Cochran et al. (13) extract a set of linear models to relate temperatures to workload characteristics and core frequencies. Eguia et al. (14) combine identification techniques with an overfitting remedying algorithm to reduce complexity and improve accuracy; nevertheless, the results need to be improved for fast power changes and thermal interactions between adjacent cores. All the above solutions exploit centralized control and deal with model identification of the whole die. Their complexity and computational burden increase fast with the number of cores. Therefore applying these solutions in upcoming many-cores (15) is very expensive. This complexity problem has been addressed in the control-theory literature. Several authors (16) (17) have shown how to reduce significantly the complexity of MPC by using distributed solutions. This approach is well-suited for large-scale systems and consists in dividing the control problem into sub-problems with lower complexity.

7.1.2 Contributions

In this chapter we present a complete distributed solution that combines energy minimization, MPC based thermal capping and thermal model self-calibration. according to the incoming task workload characteristics, each local node first selects the minimum frequency (f_{EC}) that preserves the performance within a tolerable overhead. Second, if necessary each local MPC controller trims the frequency to ensure a safe working temperature. Local controllers jointly optimize global system operation by exchanging a limited amount of information at run-time on a basis. Third we address model uncertainty by self-calibration: each thermal controller node extracts automatically the local thermal model by applying a set of training stimuli and monitoring the thermal response of the neighbourhood area. The distributed controller strategy combined with the distributed thermal model calibration phase allow us to take advantage of the parallelism of the underlying multi-core platform by running different instances of the controller and self-calibration routine in parallel.

The chapter is organized as follows. Section 7.2 introduces the key aspects of power and thermal issues in a multicore scenario. Section 7.3 describes the building blocks of the proposed solution. In Section 7.4 the performance of the presented distributed energy-aware thermal controller are fully evaluated in a real uses case scenario by implementing it in a full-system virtual platform. Final conclusions are drawn in Section 7.5.

7.2 Background Concepts

As previously introduced a multicore scenario, the temperature distribution across the die area depends on the chip floorplan, its thermal environment, and the power consumption of the cores. The latter has been shown to be related to the operating point/performance level and workload characteristics, such as instruction type, density and data locality (18).

Recalling results in previous chapter 5.4 Fig. 5.29 shows the results of a set of tests performed to quantify the relationship existing between power, frequency and Clocks-Per-Instruction (CPI) of the running task, for each core in a general purpose multicore[1]. The dots represent the actual power consumption whereas the solid lines represent the fitting model curve extracted by these data, described by Eq. 5.5:

From the figure we can notice that core dynamic power depends non-linearly on frequency, sublinearly on the CPI of the application and the two dependencies are coupled, CPI dependency is influenced by frequency value.

From Fig. 7.1a we can notice that a significant power reduction can be achieved, not only in cpu-bound program phases (low CPI) but also in memory-bound phases (high CPI). This is a key effects to use dynamic voltage and frequency scaling to obtain energy-efficiency instead of only power reduction. Indeed whereas scaling the frequency of a core executing a cpu-bounded task brings to a performance loss and to a energy inefficiency due to static power, scaling down the frequency of a core executing a memory-bound task does not lead to execution time overhead, thus the dissipated total energy is reduced.

To derive a complete thermal model of multicore dies, the ”causal chain” $(Frequency, CPI)$ $\rightarrow DissipatedPower \rightarrow Temperature$ can be split in two parts. The first part can be addressed separately in each core according to Eq. 5.5. Differently, the temperature variation in each point of the die will be affected by: the distribution of the power dissipated by all the cores, the current die temperature map and the chip physical properties. Nevertheless, the whole powers-to-temperatures model can be split in simpler interacting models by dividing the die area in regions, aligned with cores for obvious convenience.

[1]Intel® Xeon® X7350

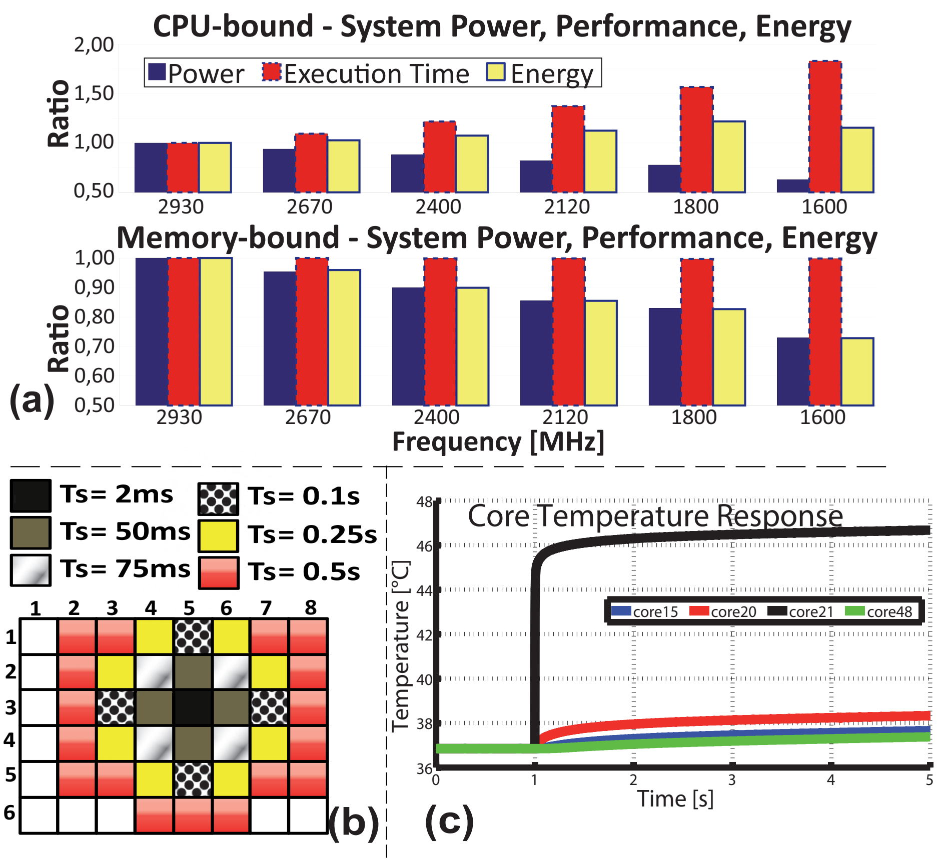


Figure 7.1: Multicore exploration results

According to the granularity usually required for thermal control, we can assume uniform power and temperature distributions in each core area. Then, recalling Fourier's law, the temperature of each core can be assumed dependent on its own dissipated power, ambient temperature and adjacent cores temperatures (boundary conditions). This assumption is actually straightforward for continuous time models only. When discrete-time models are considered, a larger coupling among cores has to be considered to account for the "chain of interactions" taking place during the blind intervals among samplings. Recalling again the Fourier's Law, the coupling among two cores will be inversely related to their distance and directly related to the sampling time period. Hence, the "equivalent neighbourhood" of a core depends on the floorplan combined with the adopted sampling.

To verify this assumption we took an example loosely correlated with the Intel® SCC experimental architecture (15). The floorplan is fully tiled with 48 core/regions, each with an area of $11.82mm^2$ and a maximum power consumption of 2.6W. We used this set-up with the

HotSpot thermal analysis tool (19), stimulating it with a power step in the central core (21) while keeping all the other cores at zero power consumption. Fig. 7.1b shows with different colours the cores that increases their temperature as result of the central core power step after different time interval. We can notice that the radius of thermal influence of the central core increases with the time interval: within 50ms it impacts only the closest core along the four cardinal directions, whereas at 1s all cores are effected.

Thus, if a small sampling time is adopted with respect to thermal time-constants (50ms or less), the effect of time-discretization can be neglected, assuming the equivalent neighbourhood equal to the physical one. These considerations are the basis for developing our distributed thermal control.

Finally, Fig. 7.1c and results in (20) highlight that the thermal dynamics of each core is characterized by two time constants: a faster one, at a few ms, is related to the silicon surface, whereas the slower one, at a few seconds, is related to the heat spreader. This behaviour, needs to be carefully accounted in model identification and control design.

7.3 Architecture

Fig. 7.2 depicts the block diagram of the proposed solution. Each controller node (i) is made-up by three main parts:

- The Energy Controller (EC_i): at the k-th sampling instant, it takes as input the predicted CPI value of the running task for the following time interval ($CPI_i([k, k+1]|k)$) and produces as output the core frequency settings for the following (k to k+1) time interval ($f_{EC_i}(k)$) that minimizes the core power consumption while allowing a tolerable performance overhead.

- The MPC-based Thermal Controller: at the k-th interval, it receives as inputs the Energy Controller output frequency ($f_{EC_i}(k)$), its own core temperature ($T_i(k)$), the temperature of the physical neighbours ($T_{neig_i}(k)$)[1] and the ambient temperature ($T_{AMB}(k)$). Then, according to the safe reference temperature (T_{MAX}) at which the core temperatures ($T_i(k)$) must be constrained, the MPC algorithm adjusts the actual frequency command ($f_{TC_i}(k)$), minimizing the displacement from the Energy Controller requirement[2].

[1]The sampling time is assumed small enough.

[2]The computation and actuation times for EC and TC are assumed negligible with respect to sampling time interval. Hence, for mathematical modelling, control outputs are considered generated at the same instant of sampled

- The Thermal Model Self-Calibration Routine: it automatically derives, off-line, the local, but interacting, thermal prediction model adopted in MPC-based TC blocks (again according to Section 7.2).

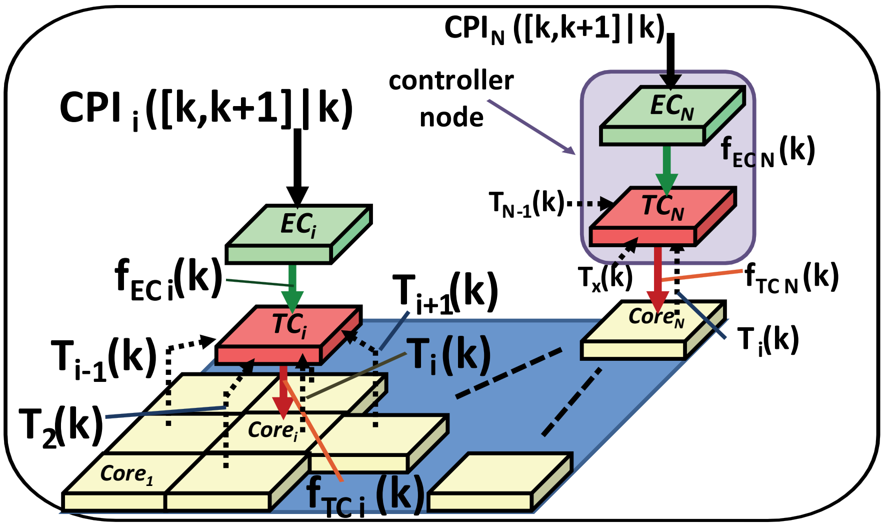


Figure 7.2: General Architecture

Section 7.3.1 describes the Energy Controller algorithm whereas Section 7.3.2 describes our thermal controller solution. Section 7.3.3 instead presents the thermal model self-calibration routine.

7.3.1 Energy Controller

The goal of the Energy Controller (EC_i) is to provide the optimal frequency trajectory ($f_{EC_i}(k)$) to the Thermal Controller (TC_i). This frequency is selected to minimize the power consumption while preserving the system performance. Thus we can split the energy minimization problem from the temperature constraining one. The Energy Controller takes advantage of the parallel architecture by letting each core (i) compute autonomously the future frequency in line with the incoming workload requirements.

Considering an in-order architecture[1] and the average time needed to retire an instruction, composed by two terms: (T_{ALU}) portion of time spent in active cycles and (T_{MEM}) portion of

inputs.

[1]Multicore trend is toward in integrating high number of simpler processor(15).

time spent in waiting for memory cycles. Whereas the first term is proportional to the input frequency, the second one is constant to it and depends to the memory access latency.

Assuming f_M the maximum frequency allowed by the system, $f_{CK_i}(k) = \frac{f_M}{\alpha}$ a generic one and given the task CPI requirement for the interval(k), for the core i ($CPI_i(k)$), we can write the task execution time ($Time_i(k)$) as:

$$Time_{M_i}(k) = \#_{INST} \cdot [1 + (CPI_i(k) - 1)] \cdot \frac{1}{f_M} \tag{7.1}$$

$$Time_{CK_i}(k) = \#_{INST} \cdot [\alpha + (CPI_i(k) - 1)] \cdot \frac{1}{f_M} \tag{7.2}$$

By combining them the execution time overhead % can be represented as function of the new frequency $f_{CLK_i(k)}$ and $CPI_i(k)$ as reported in Eq. 7.3.

$$\%_i(k) = \frac{Time_{CK_i}(k)}{Time_{M_i}(k)} - 1 = \frac{\alpha + (CPI_i(k) - 1)}{1 + (CPI_i(k) - 1)} \tag{7.3}$$

Inverting the last equation (Eq. 7.4) we can find the future frequency ($f_{EC_i}(k)$) output of the Energy Controller (EC_i) that minimizes the power consumption for the given processor i running a task characterized by the $CPI_i([k, k+1]|k)$, while preserving the performance within a tolerable performance penalty (%).

$$f_{EC_i}(k) = \frac{f_M}{(1 + \%) + (CPI_i([k, k+1]|k) - 1) \cdot \%} \tag{7.4}$$

7.3.2 Thermal Controller

Our solution goal is tracking the desired cores frequencies, without crossing the cores temperature bounds imposed by each local MPC controllers of our distributed approach. As discussed in Section 7.3, we use MPC controllers because it has been shown in (11) and (6), that well perform the previous aim. A MPC controller consists of two elements: the optimization problem solver and the model used for predictions.

First, each of our local controllers minimizes a quadratic cost function with constraint, formalized as:

$$min \sum_{k=0}^{N-1} \|Q \cdot (f_{TC}(t+k) - f_{EC}(t+k))\|_2 \tag{7.5}$$

s.t.

$$0 \leq T(t+k|t) \leq T_{MAX} \tag{7.6}$$

This formulation simply means that, to maximize the performance and respect the thermal constraint of one core, the differences (also called tracking error) between the desired core frequency f_{EC} and the frequency selected by the thermal controller f_{TC} must be minimized. At the same time the temperature T must be constrained below a thermal safety threshold, called T_{MAX}. Matrix Q is the non-negative weights matrix for the tracking error and N is the prediction horizon.
This optimization problem has been obtained by "locally projecting" the problem formulation proposed in (11), where a single MPC controller optimally constraints all the cores temperatures, maximizing the global performance. Later in this section we refer it as "centralized solution", where f_{EC}, f_{TC} and T are vectors rather than scalars.

Second, each local MPC regulator estimates future core temperature by using the following non-linear model for the single core:

$$\dot{x} = A \cdot x + B \cdot [P_{EC}, T_{AMB}, T_{NEIGH}]' \quad with \quad P_{EC} = g(f_{EC}, WL) \tag{7.7}$$

where x is the state vector (one is T), P_{EC} is the power consumption of the core, WL is the workload and T_{NEIGH} is the temperature contribution of the neighbours cores.
(21) shows that MPC model must be linear to have a convex problem and a simple and robust control solution. Thus we have confined the non-linearity outside the MPC, in the frequency to power relation (expressed by the $g(\cdot)$ function) and kept only the linear part of the prediction model inside the LQ MPC controller. As consequence the core power becomes the controlled variable, $g^{-1}(\cdot)$ transformation is needed to convert the output power (P_{TC}) in frequency command and the cost function can be expressed using core power error instead of core frequency error. This is shown in Fig. 7.3 that describes the distributed controller architecture.
The linear model accuracy is a key issue of MPC controllers. We preserve it by extracting the model parameters directly from the target multicore. This is achieved thanks to a self calibration routine explained in Section 7.3.3. As discussed in Section 7.2 the core thermal transient shows two time-scale responses, thus a second order linear model should be used for each core. This leads to a model order (number of states) larger than the number of temperature sensors (one per core). Therefore an observer block is required to estimate the state values, as in Fig. 7.3.

We tested our solution against the centralized one that, differently from (6), it uses a non linear power model and, differently from (11), it uses a more accurate power model ($g(\cdot)$) that account also the workload properties. We evaluate the benefits of these add-ons in trace-driven

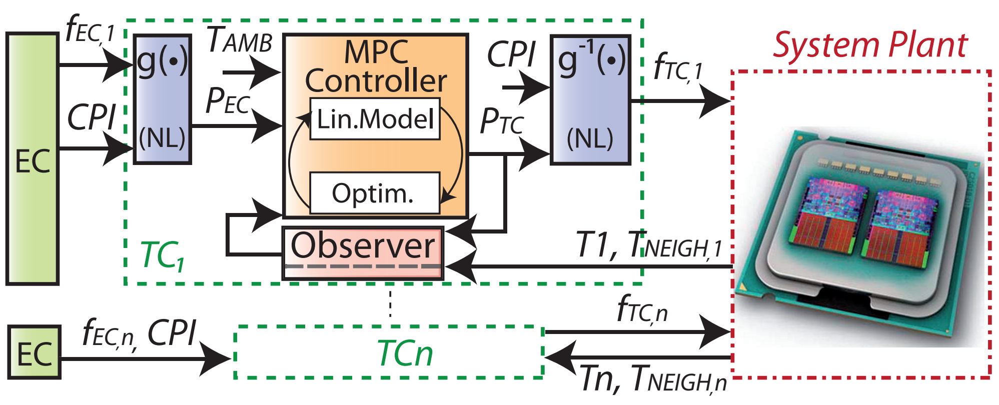


Figure 7.3: Thermal Controller structure

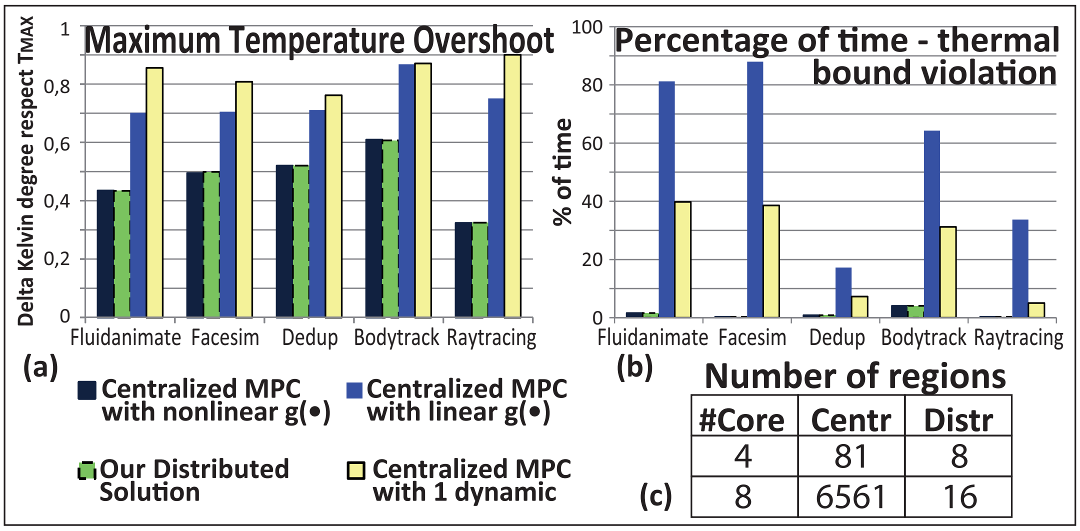
simulator[1] running Parsec2.1 (22) benchmark traces.[2] Fig. 7.4a and b, show the performance losses in the “centralized solution” by first, substituting the non-linear $g(\cdot)$ function with a linear[3] one (as in (6)) and secondly, by using a first order prediction model instead of a second order model (as in (6)). Both cases shown a significant performance worsening.

To reduce MPC computation complexity in (12) an explicit approach has been proposed to reduces the computational burden (one of the major drawbacks of the classical MPC strategy). It solves the optimization QP problem off-line for each system state $x(t)$, instead of solving it online for each time step. The obtained solution is a continuous piecewise affine function of the state ($u(x(t))$) similar to divide the state space in regions each one characterized by a different linear control law. Thus, depending on the current state value a different control law is applied on system. Unfortunately the explicit solution only moves the complexity from the computational cost to the memory usage (12)(confirmed by results in (11)). Thus we can use the regions number of the explicit formulations as a complexity metric for the on-line controllers, since it directly relates to computational cost. Fig.7.4c shows that our distributed solution improves scalability and effectively reduces complexity. Whereas the number of regions of the centralized solution grows exponentially, the number of regions of the distributed one globally grows linearly. Indeed the complexity of the single controller node remains constant regardless the number of cores in the system. Also note that the distributed solution runs in parallel on all the cores encouraging robustness, safety and a lower computational burden for the cores, while the centralized one runs only on one core. Even though the distributed solution complexity is

[1]Internally it considers the four core floorplan presented in (18).

[2]Traces are profiled on a general purpose quad core.

[3]Best least-squares fit of the power data using as input the mean frequency and the mean workload



#Core	Centr	Distr
4	81	8
8	6561	16

Figure 7.4: Thermal MPC performance analysis

lower than the centralized one, Fig.7.4a and b shows that both the solutions have comparable performance.

7.3.3 Self-calibration routine

The knowledge of the thermal model of the multi-core die is a MPC prerequisite. In many practical cases this model is not available or imprecise. We address model uncertainty by self-calibration: our thermal control framework directly learns the thermal model by monitoring the thermal response of the system after applying a set of training stimuli. This approach relies on system identification (SID) techniques. As shown in previous section and in (23), complex prediction models complicate the control action and cause overhead. Thus the implemented approach has two aims: capturing entirely the system thermal response and reducing the state space model of the plant that in reality it would have an infinite dimension.

Indeed our self-calibration routine is distributed: each core is associated with a self-identification agent that learns the local model. This approach perfectly fits our distributed control solution, since the regulator of each core directly exploits the identified local model for prediction. Secondly, it offers a low complexity solution to counteract the SID computational cost in large multicore systems. Indeed for MIMO model the SID complexity explodes with the number of inputs. Each agent implements an ARX (AutoRegressive eXogenous) model

(24) (25):

$$T(k) = \alpha_s \cdot T(k-1) + \dots + \alpha_1 \cdot T(k-s) + \beta_{1,s} \cdot u_1(k-1) + \dots$$

$$+\beta_{1,1} \cdot u_1(k-s) + \beta_{2,s} \cdot u_2(k-1) + \dots + e(k) \tag{7.8}$$

where T is the temperature of the core (the model output), s is the model order, $u_i(\cdot)$ are the model inputs (the dissipated power of the core P_{EC}, the ambient temperature T_{AMB} and the temperatures of the neighbours units), $e(k)$ is a stochastic white process with null expected value representing the model error and α_i, $\beta_{i,j}$ are the identified parameters. As shown in Section 7.2 and in Eq. 7.8, each model is a simple MISO model: we have a single output and multiple inputs $u_i(\cdot)$. The core power consumption can be estimated from the core operating point and from the current workload characteristic (see Eq. 5.5 in Section 7.2) or can be directly measured from power sensors present in recent MPSoC (15).

The self-calibration routine first forces a Pseudo-Random Binary Sequence (PRBS) power input to each core, while probing the cores temperature. Then it derives the parameters α and β by solving a least square problem that minimizes the error $e(k)$. To take into account both the slow and the fast dynamics highlighted in Section 7.2, we use a second order ARX model[1]. Fig. 7.5a shows the model and plant temperature responses to a different PRBS from the self-calibration one.

The performance of the identified model against the original one are evaluated by looking at the temperature response of each core running Parsec2.1 benchmarks. Fig. 7.5b shows the mean absolute errors of the identified model in a four cores floorplan(18) and in an eight cores floorplan obtained duplicating the previous one. The resulting errors are less than $0.3\,^{\circ}K$. The showed results are obtained running simulations on Matlab/Simulink environment. In a real system we expect to run the self-calibration routine during start up phase and each time the model behaviour differs from plant one.

7.4 Experimental Results

In this section we illustrate the performance of our solution.

We implemented it in the virtual platform environment presented in previous Chapter 6. For each core (i) in the emulated system we execute, with a time step of 1ms, both the Energy

[1]The identified models states have not a physical meaning. To match the core temperature with the first state of each model we apply a change of coordinate transformation to obtain a matrix $C = [I_n \mid 0_n]$

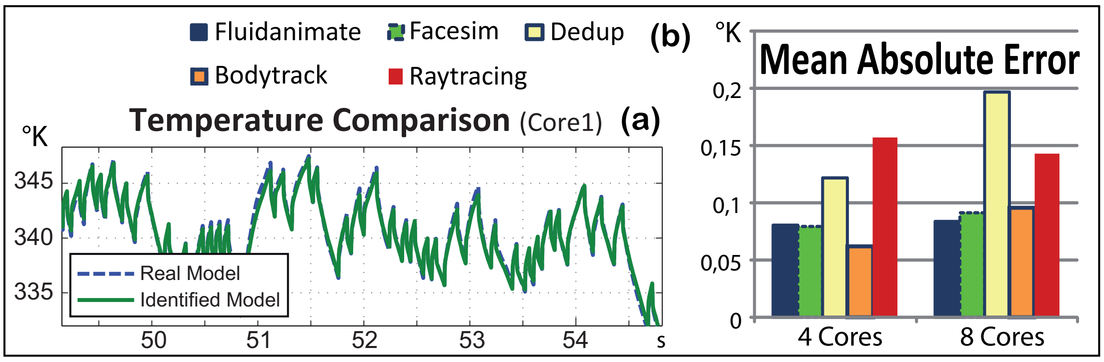


Figure 7.5: Self-calibration routine results

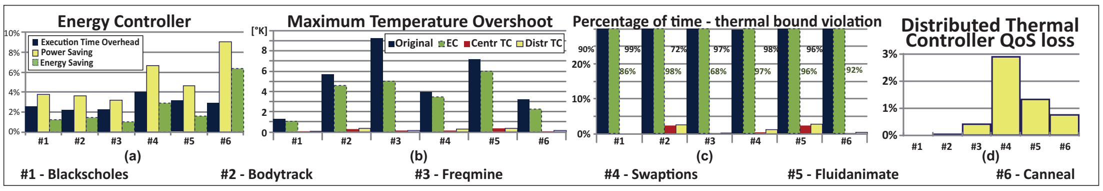


Figure 7.6: Virtual platform test results

Controller(EC_i) and the Thermal Controller (TC_i) routines. The Energy Controller internally estimates the CPI every 1ms by using a last value prediction. This interval of time is comparable with modern O.S. Instead the Thermal Controller routine embeds, as presented in Section 7.3.2, the explicit MPC implementation and estimates the full state vector with the state observer. Complexity analysis in Section 7.4.1 demonstrates that the distributed solution has negligible run-time. Thus, the perturbation due to its computations to the program execution flow can be neglected.

7.4.1 Tests Results

In this section we show the results of the proposed solution running on the virtual platform. Each controller runs on each core of the target architecture (18) under different Parsec2.1 (22) benchmarks workloads. All the benchmarks have been executed with a number of tasks equal to the number of cores of the target architecture, and with the input set "simsmall", and constrain temperature ($T_{MAX} = 330^\circ K$)[1]. We run each Parsec benchmark under four possible configurations: Original, only Energy Controller(EC), Centralized Thermal Controller (Centr TC) and

[1]Used thermal model is calibrated on a device with high performance thermal dissipation dynamics, indeed to stress our thermal controller we are forced to use a lower temperature constraint

our Distributed Thermal Controller (Distr TC).

Fig. 7.6a shows the performance of the EC alone, while allowing a performance penalty of $\%_i(k) = 5\%$. We can notice that it is able to maintain the performance overhead under the selected threshold while achieving a significant power and energy saving.

The global solution performance are instead analysed by considering the maximum temperature overshoot with respect to the constraint, the percentage of time the temperatures violate the constraint (we consider the limit violated when temperature exceeds it of $0.1\,^{\circ}K$), and a metric that quantify the controller quality of service (QoS) degradation due to thermal constraint (QoS Loss). We decide to compute it as the mean squared error between the energy controller frequency target (f_{EC}) and the one applied to the system by the controller (f_{TC}). We relativized it against the centralized controller one. Fig. 7.6 shows the results collected. First, we can notice that the proposed distributed solution performs as well the centralized one. Fig. 7.6b shows the maximum overshoot in kelvin degree above the safe thermal threshold (T_{MAX}) whereas Fig. 7.6c shows that both solutions are capable of drastically reducing the portion of time in which each core runs out of the thermal bound. Looking at the QoS Loss performance figure (Fig. 7.6d), we notice that our proposed solution performs at the same level of the centralized one, with a degradation less than 3%. Finally in more symmetrical workloads[1], such as `swaptions`, `fluidanimate`, `canneal`, we noticed that the average frequency applied to the external cores (#1, #4) is kept lower (up to -14%) than the internal cores. This is a sign that the MPC controller is able to optimize the core frequency locally, taking advantage of the difference between the local thermal models. Indeed the external thermal models have less thermal dissipation headroom since thermal model considers the chip lateral boundary adiabatic (26).

7.5 Conclusions

We have presented a novel fully distributed, energy-aware, MPC-based thermal capping controller for modern multicore platforms. It works coupled with a model self-calibration routine, that allows the controller to automatically recognize the thermal properties of the underlying platform. It exploits a fully distributed architecture, that fits very well the multicore parallelism and distributed nature. We show that this approach performs similarly to the state-of-the-art centralized Thermal Model Predictive Controllers, but with a significantly lower computational

[1]The parallel benchmark executes the same code on all the processors.

cost. Indeed the global complexity cost of our solution scales linearly with the number of cores and it is fully parallel, whereas the centralized one scales exponentially and it parallelization is not trivial. We tested our solution in a real uses case scenario by running it in a complete virtual platform. The results show that our controller is capable to satisfy temperature constraints with high precision, while minimizing system energy.

Bibliography

[1] IDC. Worldwide server power and cooling expense 2006, 2010 forecast. http://www.sun.com/service/eco/IDCWorldwideServerPowerConsumption.pdf.

[2] Hanson H. et al. Thermal response to DVFS: analysis with an Intel Pentium m. In ISLPED '07, pages 219-224, 2007.

[3] Tiwari A. et al. Facelift: Hiding and slowing down aging in multicores. MICRO '08, pages 129-140, 2008.

[4] P Chaparro et al. Understanding the thermal implications of multi-core architectures. IEEE Transactions on Parallel and Distributed Systems,18(8):1055-1065, Aug. 2007.

[5] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer's Manual - Volume 3B, June 2009.

[6] Y. Wang, K. Ma and X. Wang, "Temperature-Constrained Power Control for Chip Multiprocessore with Online Model Estimation", *ISCA*, 2009. 124, 125, 130, 131, 132

[7] G. Dhiman, T. S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning", *ISLPED*, August 27-29, 2007, Portland, OR, USA. 124

[8] K. Skadron, T. Abdelzaher, M. R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management". *Technical Report*. UMI Order Number: CS-2001-27., University of Virginia. 124

[9] M. Kadin, S. Reda, A. Uht, "Central vs. distributed dynamic thermal management for multi-core processors: which one is better?". *GLSVLSI* 2009. ACM, New York, NY, 137-140. 124

[10] Z. Wang, X. Zhu, C. McCarthy, P. Ranganathan, and V. Talwar, “Feedback Control Algorithms for Power Management of Servers”. *FeBid*, Annapolis, MD, June 2008. 124

[11] F. Zanini, D. Atienza, L. Benini and G. De Micheli, “Multicore Thermal Management with model predictive control”, *IEEE*, 2009. 124, 130, 131, 132

[12] A. Bemporad, M. Morari, V. Dua and E.N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems”, *Automatica*, Vol. 38, 2002, pp 3-20. 124, 132

[13] R. Cochran, S. Reda, “Consistent runtime thermal prediction and control through workload phase detection”. *DAC* 2010. ACM, New York, NY, 62-67. 125

[14] Thom J. A. Eguia, Sheldon X.-D. Tan, Ruijing Shen, Eduardo H. Pacheco and Murli Tirumala, “General Behavioral Thermal Modeling and Characterization for Multi-core Microprocessor Design”. *DATE* 2010 Dresden 125

[15] J. Howard et al, “A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS”, *ISSCC* 2010. 125, 127, 129, 134

[16] E. Camponogara, D. Jia, H. Krogh and S. Talukdar, “Distributed Model Predictive Control”, *IEEE Ctl. Sys. Mag.* 22 (2002) (1), pp. 44-52. 125

[17] R. Scattolini. Architectures for distributed and hierarchical Model Predictive Control. *Journal of Process Control* (May 2009), 19 (5), pg. 723-731. 125

[18] A. Bartolini, M. Cacciari, A. Tilli, L. Benini and M. Gries, A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-performance Multicores, *GLSVLSI*, 2010. 126, 132, 134, 135

[19] Huang Wei et al. Accurate, pre-RTL temperature-aware design using a parameterized, geometric thermal model. *IEEE Trans. Comput.*, 57(9):1277-1288, 2008. 128

[20] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan. ”Differentiating the roles of IR measurement and simulation for power and temperature-aware design”, *ISPASS*, 2009. 128

[21] E. F. Camacho and C. Bordons, “Model Predictive Control” *Springer*, 1999. 131

[22] C. Bienia, S. Kumar, J. P. Singh and K. Li, “The PARSEC Benchmark Suite: Characterization and Architectural Implications”, *PACT*, 2008. 132, 135

[23] F. Zanini, D. Atienza, G. De Micheli, S. P. Boyd, Online Convex Optimization-Based Algorithm for Thermal Management of MPSoCs, *GLSVLSI*, 2010. 133

[24] L. Ljung, “System Identification - Theory For the User”, *2nd ed, PTR Prentice Hall* Upper Saddle River, N.J., 1999. 134

[25] R. Guidorzi, “Multivariable system identification”, *Bononia University Press*, 2003 134

[26] G. Paci, M. Morari, V. Dua and E.N. Pistikopoulos, Exploring temperature-aware design in low-power MPSoCs, *DATE*, Vol.1, 2006, pp 1-6. 136

Chapter 8

Publications

A. Marongiu, A. Acquaviva, L. Benini, A. Bartolini. (2008). *Analysis of Power Management Strategies for a Large-Scale SoC Platform in 65nm Technology.*
DSD '08. IEEE Proceedings of International Conference on Digital Systems Design. Parma, Italy. 3-5 Sept. 2008. (pp. 259 - 266). ISBN: 978-0-7695-3277-6. IEEE.

M. Ruggiero, A. Bartolini, L. Benini. (2008). *DBS4video: dynamic luminance backlight scaling based on multi-histogram frame characterization for video streaming application.*
Proceedings of the 8th ACM international conference on Embedded software. Atlanta, GA, USA. 2008. (pp. 109 - 118). ISBN: 978-1-60558-468-3. NEW YORK: ACM (UNITED STATES).

A. Bartolini, M. Ruggiero, L. Benini. (2009). *Visual quality analysis for dynamic backlight scaling in LCD systems.*
Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09. Nice, France. 20-24 April 2009. (pp. 1428 - 1433). NEW YORK: IEEE Press (UNITED STATES).

A. Bartolini, Ruggiero M., Benini L. (2009). *HVS-DBS: human visual system-aware dynamic luminance backlight scaling for video streaming applications.*
Proceedings of the 9th ACM international conference on Embedded software. Grenoble, France. October 12 - 16, 2009. (pp. 21 - 28). ISBN: 978-1-60558-627-4. NEW YORK: ACM (UNITED STATES).

A. Bartolini, M. Cacciari, A. Tilli, L. Benini, M. Gries. (2010) *A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores.*
GLSVLSI '10 Proceedings of the 20th symposium on Great lakes symposium on VLSI 2010. Providence, Rhode Island (USA). May 16 -18-2010. (pp. 311 - 316). ISBN: 978-1-4503-0012-4. : ACM (UNITED STATES).

A. Bartolini, M. Cacciari, A. Tilli, L. Benini. (2011) *A Distributed and Self-Calibrating Model-Predictive Controller for Energy and Thermal management of High-Performance Multicores.*
Design, Automation & Test in Europe Conference & Exhibition, 2011. DATE '11. Grenoble, France. 14-18 March 2011. NEW YORK: IEEE Press (UNITED STATES).

A. Bartolini, M. Cacciari, A. Cellai, M. Morelli, A. Tilli and L. Benini. (2011) *Fault Tolerant Thermal Management for High-Performance Multicores.*
Design, Automation & Test in Europe Conference & Exhibition, 2011. DATE '11. Grenoble, France. 18 March 2011. (workshop uPM2SoC)

Chapter 9

Conclusion

Power consumption is one of the major raising concern to ICT technology providers and users. This because carbon emission of the whole ICT sectors significantly accounts for the world-wide one, because portability of mobile clients is constrained by battery storage capacity and finally because server farms and data centers spend yearly a huge amount of money in powering and cooling down the computational components. Unfortunately not all the energy used by these systems translates in higher quality of services (QoS), thus significant saving can be achieved by dynamically scaling the devices performance when not needed. Something like switching off the room light when you are not in the room. Due to the time scale involved this cannot be performed explicitly by the user, but require the system self-adapting to real performance requirements. This is called dynamic power management. In this thesis we analyzed the ICT system identifying scenarios that would benefit from these techniques. Mobile devices display system and high performance computing cooling system.

In the first scenario we develop a framework for assessing the final rendered image quality as really perceived by the human user. We used it to evaluate the performance in terms of QoS, of dynamic backlight scaling techniques (DBS). We discover that none of the techniques used in state-of-the-art is capable of accurately accounting the perceived final user quality. We then propose and prototype a novel approach to do that. Experimental results confirmed the high performance of our approach in terms of both power and QoS. Our HVS-DBS allows finding the optimal trade-off between QoS and power savings. In the second scenario we developed tools and use them to accurately characterize the state-of-the-art server machines. We first characterize future parallel benchmarks highlighting different program phases and variation suitable for system level performance adaptation. Then we combine data extracted by this characterization phase with a functional simulator and a control design software for creating a novel platform for fast prototyping and exploring dynamic power and thermal management (DTM) techniques for chip multiprocessors.

One of the most promising DTM techniques adopts Model Predictive Controller approach to scale the core frequency to constrain the core temperature into a safe range. It uses in the control process future thermal projections to optimize the control action, thus minimizing the performance slow down. Unfortunately MPC has high overhead and does not scale well with increasing numbers of cores. We use the developed virtual platform to develop and test a novel distributed MPC approach for multicore DTM. The proposed solution integrates also a energy minimization stage to reduce energy consumption and a thermal model self-learning stage to adapt the internal MPC thermal projections to ambient changes. I'm currently working, with our industrial partnership to prototype it in a future manycore test chip and in today data-center.