Runtime Management Techniques for Power- and Temperature-aware Computing

---

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Electrical Engineering)

by

Zhijian Lu

January 2007

Approvals

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical Engineering)

Zhijian Lu

Approved:

Dr. John Lach (Advisor)	Dr. Lloyd R. Harriott (Chair)
Dr. Kevin Skadron	Dr. Mircea R. Stan
Dr. Joanne Bechta Dugan	Dr. Zongli Lin

Accepted by the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

January 2007

Abstract

As integrated circuit (IC) technology advances to the sub-100nm region, power and associated thermal effects are becoming a limiting factor in high-performance circuit design. In addition to battery lifetime for mobile devices, power and temperature are emerging as concerns due to the strong temperature-dependence of leakage power, circuit performance, IC package cost and reliability. In traditional design methodologies, a constant worst-case power and temperature are commonly assumed, leading to excessive design margins (resulting in higher cost) or degraded performance due to temperature or power constraints. In reality, circuits exhibit strong workload-dependent variations (e.g. execution time, temperature) at runtime. Therefore, dynamically adapting a circuit to the behaviors of its workloads (trading off between performance, power, temperature and reliability) would enable reclamation of design margins from previous worst-case assumptions.

In this dissertation, improved runtime techniques are presented to attenuate power and thermal constraints on circuits by exploiting dynamic workload variations. These techniques include efficient dynamic voltage/frequency scaling (DVS) techniques, such as using feedback control and statistical information, to reduce circuit power consumption while maintaining performance requirements. This dissertation also explores the effect of temperature variations on circuit reliability, develops a reliability model subject to dynamic thermal stress, and investigates architectural techniques to maximize circuit performance without violating IC lifetime specifications. It is shown that, using these power- and temperature-aware runtime management techniques, substantial power and performance margins can be reclaimed from methodologies using worst-case power and temperature assumptions. In addition, the dynamic models developed in this dissertation can also be used for design time optimization.

To my wife, my parents and my sister

Acknowledgments

It was my great pleasure to have had the opportunity to work with three professors throughout my PhD research. I learned greatly from them in both academic and personal life. I thank Prof. John Lach, my advisor, for his continuous financial support for my graduate study, for his tolerance of my pursuing different research topics on my own interests, and for his excellent help in polishing my publications. With solid funding support from John, I was able to travel worldwide to attend major conferences and to present my work. I owe a substantial amount of gratitude to Prof. Kevin Skadron for his great technical guidance and patient discussions with me. Kevin sets for me an example of an outstanding researcher. My deep appreciation also goes to Prof. Mircea R. Stan. My research couldn't have been as productive as it was without his countless insightful comments and suggestions. Mircea's ideas usually saved me a lot of effort in my struggling for success. He is definitely one of the smartest people I have met.

I am grateful to my wife Haini, who has been very supportive of me since I began my graduate studies at UVa (the same time as I first met her). Besides being a graduate student herself, facing the same pressure as I did, Haini continues to show her unlimited love to me and takes good care of me. I apologize for those hassles I brought to her when I was busy with my research, especially those times when I was catching paper submission deadlines.

I will be forever indebted to my parents and my lovely sister for their endless support and encouragement. As a son and a brother, I wish I could always have been there when they needed me. I can never pay back as much love as my family has given to me.

I thank my fellow students Wei Huang, Yan Zhang, Karthik Sankaranarayanan and Yingmin Li for sharing their ideas with me and giving me help with their expertise. I would like to extend my gratitude to the system administrators in both the ECE and CS departments for their quick responses in solving the problems I met when using the computing infrastructures.

During my six years at UVa, I met many friends and shared my joy and sadness with them. I thank Yangyang Yu, Haijun Fang, Lei Zhu and Dan Hui for their support and help given to me. My days with them are unforgettable.

Finally, my PhD research would not have been completed without various funding sources. My research was supported in part by NSF grants CCR-0133634, CCR-0306404, CCF-0105626, CCF-0429765, CNS-0410526, EHS-0410526 and EHS-0509245; Army Research Office grant W911NF-04-1-0288; a University of Virginia FEST Award; the Woodrow W. Everett Jr., SCEEE Development Fund in cooperation with the Southeastern Association of Electrical Engineering Department Heads; two research grants from Intel MRL; and an IBM Faculty Partnership Award.

Contents

1 Introduction 1

1.1 Motivation 1

1.2 Research Overview 9

1.3 Research Contributions 13

1.4 Dissertation Roadmap 15

2 Background and Related Research 16

2.1 Dynamic Voltage/Frequency Scaling 16

2.2 Application of Control Theory in Computing Systems 20

2.3 Reliability Modeling and Management 20

3 Power-aware Runtime Management using Procrastinating Voltage Scheduling 24

3.1 Introduction 24

3.2 System Model and Problem Formulation 27

3.3 Solutions for Discrete Frequency Sets 30

3.4 Considering Transition Overheads 35

3.5 Experimental Results 38

3.6 Summary 41

4 Power-aware Runtime Management using Control-Theoretic DVS 43

4.1 Introduction 43
4.2 Control System Design 44
4.3 Simulation Experiments and Results 50
4.4 Summary 57

5 Power-aware Runtime Management for a Video Playback System 58

5.1 Introduction 58
5.2 Related work 60
5.3 Overall architecture of a multimedia playback system 62
5.4 Timing characteristics of MPEG workloads 65
5.5 Design and analysis of feedback based speed control 68
5.6 Implementation on a hardware platform 78
5.7 Experimental results 79
5.8 Conclusion 88

6 Temperature-aware Dynamic Electromigration Reliability Model 90

6.1 Introduction 90
6.2 An analytic model for EM 92
6.3 EM under dynamic stress 95
6.4 Analysis of the proposed model 102
6.5 Electromigration under spatial temperature gradients 106
6.6 Design time optimization considering runtime stress variations 117
6.7 Summary 119

7 Temperature-aware Runtime Reliability Management for High Performance Systems 121

7.1 Introduction 121

7.2 Related Work 123
7.3 Lifetime banking opportunities 124
7.4 Dynamic Reliability Management Based on Lifetime Banking 125
7.5 Experiments and analysis for general-purpose computing workloads . . . 129
7.6 Dynamic Reliability Management for Server Workloads 134
7.7 Summary 145

8 Conclusion and Future Directions 146
8.1 Conclusion 147
8.2 Future Directions 149

A Optimal Speed Schedule for Multi-task with Memory Stalls 152

B Electromigration with Free Stress at Both Ends 158

Bibliography 160

List of Figures

1.1 Illustration of “thermal runaway”. 4

1.2 A simulated temperature/power profile for an integer unit running *mesa* Spec2000 benchmark. [61] 8

3.1 (a) PDF of task execution cycles. (b) Procrastinating DVS with 4 operating points. [66] 29

3.2 Graphical interpretations of Equations (3.3), (3.4) and (3.5). [66] 33

3.3 Proof for Theorem 3.2 34

3.4 Energy–delay curves for different processors with DVS capability (The values for each processor are normalized to its energy and delay at the full speed.). [66] 36

3.5 Energy consumption by different scheduling techniques for workloads with different cycle count distributions. [66] 39

3.6 Average task energy consumption at various deadlines when the number of frequencies in a schedule is fixed (The y–axis on the right shows the energy values normalized to those of using the full frequency set (10 fs).). [66] 40

4.1 Basic architecture of our control system. All of the functional blocks are shown with the Z transforms of their respective transfer functions. [59] . 45

4.2 Step response of the system shown in Figure 4.1. Although we use the example frame arrival and decoding rates shown in Figure 4.1, other rates yield similar results. [59] 47

4.3 Frequency scaling curves for both algorithms in one simulation run. [59] 52

4.4 Performance comparison of the two approaches in terms of average delay, energy consumption, and delay variance for all simulation runs in three workload groups (left column: M/M/1 with hybrid CPU utilization; middle column: Exponentially distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization; right column: Uniformly distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization). The control-theoretic approach provides better timing guarantees with comparable energy savings. Similar results are obtained for other workloads including uniformly distributed decoding times. [59] 54

5.1 MPEG decoding power at different CPU speed. [65] 59

5.2 Overall architecture of a multimedia playback system. [62] 62

5.3 Timing characteristics of MPEG video playback. [65] 67

5.4 A generic dynamic system model for online DVS MPEG decoding. [65] . 69

5.5 Modeling the transient behaviors of the proposed closed loop feedback system. A pulse input $D(n)$ models the disturbance caused by the change in the required decode computation power. [65] 73

5.6 Roots with the maximal magnitude as functions of Kl_c. (a) $i = 1$. (b) $i = 3$. (c) $i = 5$. The plots on the left show the locations of the roots in the unit circle and those on the right show the magnitude of the roots. [65] 76

5.7 The prototype platform (the notebook computer on the left side) and the host running the data sampling software (the desktop computer on the right side). 78

5.8 Total system energy consumption for video playback with different DVS schemes. The energy consumptions are normalized to those for the *full speed* schedule. [65] 82

5.9 Segments of the recorded power trace during playback of video clip *fantastic* (H.264) by various DVS schemes. 83

5.10 Decode speed schedule for video clip *mission* (SVQ3) by various DVS schemes. 85

5.11 Decode speed schedule for video clip *fs2004* (MPEG2) by various DVS schemes. 86

5.12 Estimated decode energy consumption by the proposed feedback scheme and the schemes with looking-ahead different number of frames. 88

6.1 EM stress build-up for different boundary conditions and α values. All processes have $\beta = 1$ (α and β are defined in Equation (6.3)). [61] 94

6.2 EM stress build-up under time-dependent current stress. In each EM process, α (defined in Equation (6.3)) oscillates between two values with different duty cycles. The time dependence of α is given in the legend.[1]All curves have the same average value of α. The solid line is the stress build-up with a constant value of α. [61] 96

6.3 EM stress build-up at one end of the interconnect with different time-dependent β functions (square waveform). The solid line is the case with a constant value of β equal to the average value of β in other curve. [61] . . 98

6.4 EM stress build-up at one end of the interconnect with time-varying α (current) and β (temperature) functions (i.e., square waveforms). The circles represent the numerical solution for time-varying α and β. The solid line is with a constant value of α calculated according to Equation (6.8) and a constant value of β equal to the average value of that in the time-varying case. As a comparison, the EM process (dotted line) simply using the average current of the time-varying case is also shown. These results show that EM process under dynamic stresses (circles) can be well approximated by a process with constant stresses (solid line). [61] 101

6.5 Temperature and current waveforms analyzed: (a) in phase current/temperature, (b) out of phase current/temperature. [61] 103

6.6 Comparison of electric current, temperature factor (β) and MTF for different peak to peak temperature cycles. All results are normalized to the average current and/or temperature case. (a) Ratio of reliability equivalent current (our model) to average current. Both cases of current variation (in and out of phase with temperature) are included. (b) Ratios of temperature factor (β) using average temperature, max temperature, and our model. (c) Comparison of MTF for four different calculations: average temperature/average current, maximum temperature/average current, our model for current in phase with temperature, and our model for current out of phase with temperature. 104

6.7 Effects of non-uniform spatial temperature distribution on EM induced void growth. (a) Various temperature profiles along a $100\mu m$ copper interconnect (left end is the cathode). (b) Void growth with different spatial temperature profiles. 108

6.8 Stress build-ups at different time points along the interconnect under spatial thermal gradients. (a) Low to high temperature profile. (b) High to low temperature profile. (c) Parabolic temperature profile. Electrons flow from the left to the right, causing compressive stress (negative in the figures) on the right side of the interconnect. The left end (cathode) is stress free to model the growth of a void. The time points ("t2" through "t10") are corresponding to the time points in the plots with the same temperature profile in Figure 6.9. 110
6.9 Void growth with non-uniform temperature distribution is bounded by those with a uniformly distributed temperature. (a) Low to high temperature profile. (b) High to low temperature profile. (c) Parabolic temperature profile. (d) V shape temperature profile. (e) Inverse V shape temperature profile. 113
6.10 Void growth subject to both temporal and spatial thermal gradients can be bounded by that using uniform temperature and current. 116
6.11 A proposed design flow incorporating runtime stress information. [61] . . 118
6.12 (a) Distribution of wires violating the MTF specification using maximum temperature (data extracted from [108]) with a total of 372 wires. (b) Reduction of the number of wires violating the MTF specification under different temperature variations (maximum temperature: 135^oC). [61] . . . 119

7.1 Temporal temperature variation. (a) Single program workload. (b) Two-program workload with context switching. [64] 124
7.2 Simple dynamic reliability management (S_DRM). [64] 128

7.3 Temperature and clock frequency profiles in different thermal management techniques for benchmark *gcc*. (a) Conventional DTM (threshold temperature =110^oC). (b) Reliability banking based DRM (reliability target temperature =110^oC). 131

7.4 Performance comparison of DTM and the proposed S_DRM. The results for S_DRM are based on high convection thermal resistance configuration. The results for DTM include two different thermal configurations. [64] . 132

7.5 S_DRM performance at different targeted lifetimes. [64] 132

7.6 Average performance comparison of DTM and DRM on a multi-program workload with different context-switch intervals ((a) 50μs, (b) 5ms, and (c) 25ms). [64] 133

7.7 A constructed workload used to mimic the thermal behavior of server workloads. [64] 137

7.8 S_DRM (simple dynamic reliability management) on the synthetic workload shown in Figure 7.7. 138

7.9 Relationship between clock frequency and lifetime consumption rate. . . 139

7.10 Performance comparison of different runtime management techniques on the synthetic workload shown in Figure 7.7 with different duty cycles of the cool phase: (a) 0.5, (b) 0.6 and (c) 0.75. [64] 140

7.11 Modeling thermal behaviors of server workloads using square waveforms. [64] 142

7.12 Performance speed-up due to lifetime banking on different workload characteristics. [64] 143

A.1 Time distribution among task execution. Note that, in general, the memory stall time is interleaved with CPU computation time during task execution. 153

A.2 Comparison of optimal energy with that using uniform speed at different values of r_m, r_1, p_1 and p_2 ($a = b$). 155

List of Tables

3.1 Energy saving by our method over the simple rounding method with different number of available frequencies. 39

3.2 Comparison between brute-force method and our dynamic programming based heuristic method when $r = 10$. 41

4.1 Performance summary for the change-point detection approach [91]. The energy column gives the energy consumed relative to no frequency/voltage scaling. [59] 55

4.2 Performance summary for our control-theoretic approach. [59] 55

4.3 Impact of number of iterations of "virtual training" for a trace of M/M/1 Model with average delay set point = 0.1 sec. [59] 56

4.4 Impact of number of iterations of "virtual training" for a trace of exponentially distributed inter-arrival times and uniformly distributed decoding times with average delay set point = 0.1 sec. [59] 56

5.1 Characteristics of a set of video clips with different compression formats (each clip has 3000 frames). 66

5.2 Available CPU clock speed/voltage pairs in the prototype platform computer (Clock speed is normalized to the maximum frequency 1530MHz). 78

5.3 Number of frames missing deadline with different DVS schemes. [65] . . 82

6.1 Proposed bounding temperatures for void growth in an interconnect with length l subject to a non-uniform temperature distribution. T_{lb} is the lower bound. T_{ub} is the upper bound. $T(x)$ is the temperature profile. $x = -l$ and $x = 0$ are the locations of the cathode and the anode, respectively (as shown in Figure 6.7). 114

Chapter 1

Introduction

1.1 Motivation

With the advance of semiconductor technology scaling, power and thermal issues have been among the limiting factors facing IC (integrated circuit) design. The power consumption on modern high performance chips has reached the limit of heat dissipation capacity of contemporary thermal package design. While battery lifetime is becoming one of the major concerns for mobile devices with the prevalence of mobile computing technology, the growth in IC power consumption has far exceeded that in battery energy capacity. On the other hand, the resulting higher temperatures due to increased power consumption not only degrade system performance, raise packaging costs, and increase leakage power, but they also reduce system reliability via temperature enhanced failure mechanisms. In the traditional design methodology, worst-case assumptions are used to ensure the system operates normally in all corner cases, which results in excessive design margins by imposing extreme design constraints, and, as a consequence, tends to offset the benefits brought by technology scaling. New design approaches are urgently needed to address the power and thermal challenges and to reclaim the design margins.

1.1.1 Power and Thermal Trend with Technology Scaling

In the past three decades, the performance (clock frequency) and functionality (transistor count) of integrated circuits have grown exponentially as pushed by Moore's law. However, increasing on-chip power consumption tends to become the major roadblock for people to realize continuing technology scaling [73], due to lack of cost-efficient ways to remove heat from the chip.

This problem can be more clearly explained from the expression of dynamic (switching) power consumption [81]:

$$P_d = \alpha \frac{1}{2} C_{eff} V^2 f \tag{1.1}$$

where α is the average transistor switching factor in each clock cycle, a value between 0 and 1, C_{eff} is the effective total capacitance on a chip, V and f are operating voltage and frequency respectively. Circuit delay can be estimated by [81]:

$$D = \frac{C_{load} V}{I_{av}} \tag{1.2}$$

where C_{load} denotes the output load capacitance, and $I_a v$ the average transistor conducting current.

With technology scaling, the dimensions (e.g. transistor channel length and width) of transistors are scaled by S ($S \approx 0.7$) at each new technology generation, and the supply voltage is scaled by U ($U \leq 1$). As a result, load capacitance is scaled by S and, in deep sub-micron (DSM) region ($< 0.25\mu m$ feature size), I_{av} is scaled by U [81]. Thus, the clock frequency $f \propto \frac{1}{D}$ is scaled by $\frac{1}{S}$ according to Equation (1.2). At the same time, individual transistor capacitance becomes smaller (i.e. scaled by S), and transistor density (transistor count per unit area) is increased by $\frac{1}{S^2}$ (i.e., new technology enables more transistors integrated on a chip.). It follows that the total effective capacitance is the sum of all transistor capacitance on a chip, which is scaled by $\frac{1}{S}$, assuming the chip size is fixed. According to Equation (1.1), total dynamic power consumption is scaled by $(\frac{U}{S})^2$. With ideal scaling where $U = S$, one might expect slight power increase in each new generation due to the increase of chip size. However, in practice, $U > S$ as certain noise margin of voltage swing has to be reserved (in ultra-deep sub-micron technology, in order to suppress the leakage current, threshold voltage is scaled very slowly, which also reduces the scaling speed for supply voltage). For example, it is predicted that the power supply voltage for high performance logic circuitry will be scaled from $1.1V$ at 2005 for $90nm$ technology to $1.0V$ at 2008 for $59nm$ technology [6]. In addition, the request for higher circuit performance favors deeper pipeline architectures which enable greater clock frequency scaling. For example, Intel's Pentium 4 processors adopted a 20+-stage pipeline to achieve multi-GHz performance at $0.18\mu m$ technology at 2001 [109]. The combination of new technology and the adoption of deeper pipeline architecture brings exponential growth in performance at the expense of similar increase in on-chip power consumption.

Consequently, chip temperature, which is proportional to power density, also increases exponentially. On the other hand, the adoption of low-k interlayer dielectrics in future technology to reduce the interconnect capacitance will further push upwards the temperature envelope because these materials usually have worse thermal conductance. Furthermore, higher temperature will in turn increase another power component–leakage power–significantly, thus the total power. It is even estimated that leakage power could be larger than dynamic power in future technology nodes [38]. There are two major sources for leakage power: sub-threshold leakage and gate leakage [81]. Sub-threshold leakage is a leakage current flowing through the reversed biased diode junctions when the transistor is in off state, while gate leakage is the current flow due to electron tunneling through the gate oxide. Gate leakage is independent of temperature and can be reduced using high-k gate dielectrics, while sub-threshold leakage is an exponential function of temperature. In addition, leakage current is also exponentially dependent on threshold voltage (i.e. transistors with smaller threshold voltage are leak), which constraints the further scaling of threshold voltage, thus further limiting the scaling factor U of supply voltage.

All these trends suggest that the time has come that we can no longer afford the high on-chip power consumption and high temperature for better circuit functionality and performance. For instance, in less than 5 years after the debut of Pentium 4 processor, Intel canceled the development plan of Pentium 4 architecture due to its extreme high power consumption and the inability to remove heat efficiently [109].

1.1.2 Power and Thermal Challenges

Power, thermal and reliability challenges are highly related. Higher power consumption results in higher temperature, which enhances many physical failure mechanisms. Furthermore, there exists a positive feedback loop between power and temperature–higher temperature leads to higher leakage and therefore total power. With careless design, it is possible that the increase in both power and temperature can be unbounded. This is called "thermal runaway" [38]. Figure 1.1 illustrates this phenomenon. The straight line

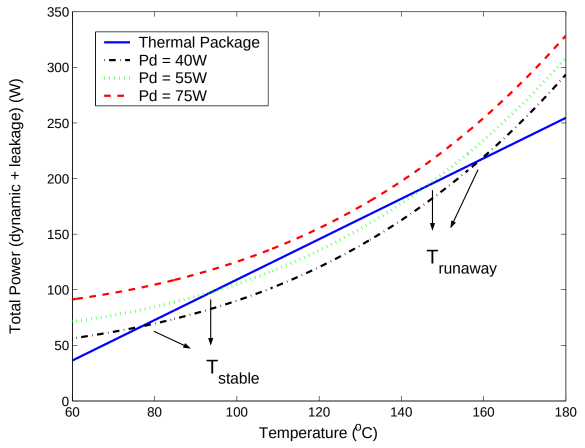


Figure 1.1: Illustration of "thermal runaway".

shows the amount of heat (power) can be removed by the thermal package at each temperature. All other curves show the dependency of total power on temperature. For a given dynamic power P_d, the interception between its power-temperature dependency curve and the thermal package line determines the final power and temperature on the chip. When the dynamic power is small, there are two interception points as the cases in $P_d = 40W$ and $P_d = 55W$. The interception point with the lower temperature dictates the final steady state of the chip. As can be seen from the figure, the final total power could be much larger than the dynamic power due the coupling between temperature and leakage. The temperature of the other interception point is the thermal runaway temperature, because the temperature and power will become infinite if the chip temperature/power is accidentally brought to above this point. Note that if the dynamic power is large enough, as the case for $P_d = 75W$ in the figure, any temperature is virtually the thermal runaway temperature, which should be prevented. In reality, runtime guarding mechanisms such as dynamic thermal management (DTM) can be deployed to avoid "thermal runaway". When the chip temperature is found to be higher than certain predefined threshold, the operations on the chip are slowed down, forcing the power consumption within certain envelope. Of course, performance is sacrificed.

High temperature will also degrade the circuit speed by reducing the charge carrier (i.e. electron or hole) mobility, increasing the interconnect resistance. The combination of high power and high temperature also create significant voltage drop across the on-chip power distribution network, thus reducing transistor conducting current and signal noise margin.

Power and thermal issues have become the limiting factors for both low-end and high-end applications. Most low-end computing systems such cell phone, multimedia player, portable medical devices are powered by battery. Thus, longer battery lifetime, more functionality and comfort (e.g. simple form factor) are the keys for market success. Unfortunately, the increasing power and temperature are opposing these goals. In high performance computing systems such as data centers, power and temperature are directly translated into operation cost – power equipment, cooling equipment and electricity consumed in both computing and cooling have occupied up to 63% of the total cost of a data center [12]. On the other hand, overheating has been identified as one of the major causes for hardware failures [41], as circuit lifetime is exponentially decreased as temperature is increased. For instance, reserchers in Los Alamos National Lab observed, when the air temperature was around 70-75^oF, a Beowulf cluster composed of 128 processors failed once a week and the failure rate doubled when the temperature was around 85-90^oF [41]. Thus, overheating has limited the actual utility delivered by the computing systems.

1.1.3 Existing Solutions

Many new low power design methodologies have been proposed to reduce circuit power consumption at design time. In general, any low power design technique trades circuit speed for energy savings. For example, designers can identify the non-critical paths in the circuit, and build that part of circuit using low swing voltage and high threshold transistors [80]. However, the critical paths could be workload dependent. Consider a general purpose processor with a floating point unit and an integer unit in it. When floating point applications are executed, the performance bottleneck is the floating point unit, and the integer unit for an integer application. In design, designer has to treat both units as critical components. Therefore, design time optimization is still based on worst-case in nature. As the transistor size shrinks further, PVT (process, voltage, and temperature) variations become more prominent and further attenuate the effectiveness of design time optimization, as designers have to ensure correct operations in all corner cases.

In addition to low power design techniques by which temperature can be reduced, many solutions are also proposed to directly address the thermal constraints, and they can be generally divided into two categories from a processor's point of view: 1. external cooling mechanisms, and 2. internal cooling mechanisms. In the first category, designers pay an extra price for more efficient cooling packages, such that the temperature is guaranteed to be below some threshold temperature all the time. However, since the cooling cost has been expensive nowadays, with increasing power consumption in future systems, using physical cooling solution alone is cost-inhibited. In the second category, people sacrifice a certain amount of performance to maintain reliability by reducing circuit speed (resulting in temperature reduction) whenever necessary. Recently developed dynamic thermal management (DTM) techniques (discussed in more depth in the next chapter) belong to this category. However, these techniques rely on a worst-case temperature-based reliability model. Under such pessimistic assumptions, DTM cooling mechanisms may often be engaged and performance penalties incurred unnecessarily.

On architectural level, the increasing power and thermal challenges also push the recent shift in the industry trend from the pursuit of higher clock frequency to that of chip multiprocessing (CMP). In a CMP architecture, multiple functional units (cores) or processing elements (PEs) are placed on a same die. By exploiting computation parallelism, the aggregate performance of the chip is increased while individual PE is operating at a relatively low speed thus consuming less power. The success of CMP architecture is dependent on the inherent computation parallelism provided by the applications.

In this dissertation, we investigate techniques to overcome some of the drawbacks of the existing solutions by exploiting application runtime variations.

1.1.4 Opportunities for Runtime Adaptation

At runtime, many workloads display strong phased behaviors and associated dynamic variations (e.g. execution time, temperature), providing opportunities for runtime optimizations which can hardly be achieved solely on design time decisions. As an example, simulated power and temperature profiles for a high performance processor running a Spec2000 benchmark are plotted in Figure 1.2. The temperature and the power of the hottest block (i.e., the integer unit) are presented. In this case, the substrate temperature varies between 110^oC and 114^oC, and the maximum power is more than 1.5 times the minimum power. One can see that for only a small portion of time is the program running at the worst-case temperature. The advantages of adaptive runtime management are two-fold. On one hand,

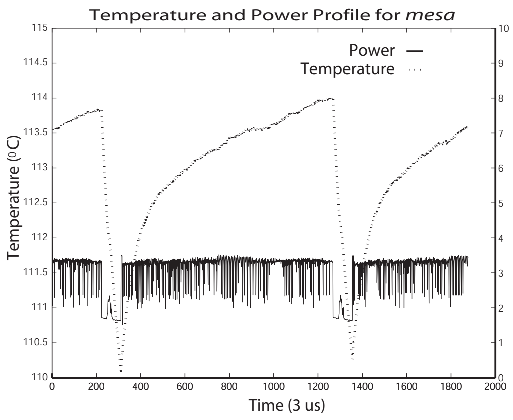


Figure 1.2: A simulated temperature/power profile for an integer unit running *mesa* Spec2000 benchmark. [61]

runtime management can adapt the system to the workload variations, reaching higher efficiency by reclamation of design margins. On the other hand, by monitoring the operating conditions continuously, runtime management can avoid those extreme cases which rarely happen, thus relieving the all corner case design constraints. The importance of runtime management has recently begun to be recognized. As a matter of fact, in Intel's next generation Itanium processor (coded Montecito) there is an embedded on-chip micro-controller which monitors on-chip power and temperature, and adjusts the voltage supply accordingly during runtime, known as Foxton technology [69].

In this dissertation, we study runtime management techniques to address the power and thermal challenges by exploiting runtime workload variations. In order to reduce the power constraint, we need to identify opportunities in runtime to aggressively apply effective low power technique. We also need to ensure that the quality of service (QoS) of the application is not harmed by our runtime management. In order to reduce the thermal constraint, we need not only reduce power consumption (thus temperature) but also investigate the effect of temperature variations on circuit reliability in order to reclaim design margins imposed by the existing worst-case assumption based DTM techniques. Our results confirm that substantial design margins can be reclaimed from methodologies using worst-case power and temperature assumptions and the increasing power and thermal constraints along technology scaling can be significantly relieved. We argue that by combining our improved techniques with existing optimization techniques, we should be able to continuously enjoy the advantages brought by technology scaling.

1.2 Research Overview

In the following, we briefly describe the approaches and methodologies applied in this dissertation research.

1.2.1 Low Power Design using Feedback Control

A system is usually designed in such a way that it can provide the target performance even in the worst-case scenario (e.g. a task to be executed has the longest execution time). However, most tasks in the runtime will finish much earlier than the worst-case execution time, which provides the opportunities to slow down the device processing speed, thus reducing the energy consumption. Dynamic voltage scaling (DVS), which will be discussed in depth in the next chapter, is a powerful technique to exploit these opportunities for energy saving while still satisfying performance requirements (e.g. decoding throughput of frames in a multimedia system), because of the super-linear relationship between power and voltage, as revealed by Equation (1.1) and (1.2). The key in deploying DVS techniques is to determine when and how much the supply voltage should be changed during operations, such that energy(power) is minimized while QoS is not sacrificed. While many researchers apply ad-hoc techniques to scale the voltage, we believe that this problem is naturally suitable for formal feedback-control theory. In our method, a PI (proportional-integral) controller is used to control the voltage dynamically, while the user specified system latency in stream processing is used as the set-point for the controller. Simulations, using traces from synthetic and real MPEG workloads, show that the proposed scheme can effectively maintain the average frame delay within 10% of the traget more than 90% of the time, while greatly reducing the energy consumption [59], [60].

While the above technique can guarantee the average throughput of a multimedia system and reduce the energy consumption, users generally prefer real-time guarantees for multimedia playback. By modeling a multimedia system as a soft real-time system, we extend our aforementioned technique and apply feedback controller to adjust the decoder's speed according to the occupancy of the buffer between the decoder and the display device. The advantages of the proposed scheme have two folds. First, by controlling the display buffer occupancy, one can avoid the explicit prediction of frame decoding time, which is a commonly used approach by other researchers and has been shown non-ideal [23, 40, 67]. Secondly, our scheme allows the decoder to decode frames across the display interval boundaries, with the slack times shared by the following multiple frames. This is an enhancement of those frame-based DVS techniques [62]. Furthermore, we implement our technique in a DVS capable platform, and evaluate various low power multimedia decoding techniques with a set of multimedia streams in various video compression formats. Experimental results reveal that our technique achieves the best trade-off among energy, playback quality, hardware resource and playback latency.

1.2.2 Low Power Design using Stochastic Workload Information

Although a lot of research efforts have been spent on DVS techniques for real-time systems, we observe that the effectiveness of a particular technique is quite dependent on the execution time distribution of the workload. And the previous feedback control based techniques have yet provided a global optimal voltage scaling solution. Therefore we are also interested in searching for a realizable optimal DVS solution using statistical information about the workloads. The optimal DVS solution for single-task systems has been proposed by other researchers [52]. However, their technique has various drawbacks in practical systems because of their ideal assumptions (i.e. continuous DVS scaling). We propose a practical DVS technique for hard real-time systems called “procrastinating DVS”, in which a task begins its execution with a low frequency, and increases the frequency gradually as the task progresses, such that the task can be finished before deadline even in the worst-case [66]. Interestingly, an independent research performed by Xu *et al.* [113] tries to solve a similar problem. With their problem formulation, the exact solution is NP hard, while our approach is analytic in nature and can be solved with very low overhead.

1.2.3 Reliability-aware Design for High Performance Systems

Higher temperatures reduce system reliability via temperature enhanced failure mechanisms such as electromigration (EM). Electromigration is an aging phenomenon of metal interconnects on the chip, due to self-diffusion of metal atoms by the momentum exchange between electrons and atoms. Since atom diffusion rate is an exponential function of temperature, increasing temperature by 10 degrees will approximately double the atom diffusion rate, thus reducing the interconnect lifetime by half.

In addition to using more expensive cooling package for the chip, two strategies are commonly used in current design flow to address EM issues for system reliability. In the first one, circuit designer designs the interconnects using the worst-case temperature, ensuring reliability specification in the worst-case scenario. However, due to increasingly higher temperature, the feasible design space for circuit designers shrinks, only allowing more conservative designs (for instance, increasing the wire width, which may however cause other problems such as power consumption and timing closure). The second approach applied to deal with the temperature-related reliability issue is DTM, which tries to trade off performance with reliability.

It is observed that system temperature varies along time, due to workload variations, such as the case shown in Figure 1.2. We first investigate the effect of temperature temporal gradients on reliability. However, existing EM models assumes constant temperature and is not suitable for dynamic analysis. Thus an efficient dynamic reliability model accounting for runtime temperature variations is needed. In this research, we derive a physics-based dynamic EM model, which can estimate interconnect lifetime in any time-varying temperature/current profile [61]. This model is verified against numerical simulations, and it reveals that design decisions and DTM techniques using worst-case models are pessimistic and result in excessive design margins (i.e. system lifetime) and unnecessary runtime engagement of cooling mechanisms. For example, calculations according to our model show that substantial design constraints on metal interconnects could be relieved [61]. This model is also useful for temperature-aware dynamic runtime management and it leads to a novel view on system reliability by modeling expected lifetime as a resource that is consumed over time at a temperature- and voltage-dependent rate. As a result, in stead of controlling temperature as in DTM, one can directly manipulate system reliability on runtime. Based on this dynamic reliability model, We propose a dynamic reliability management (DRM) technique for high performance systems. Simulation results using various workloads demonstrate that, using DRM, the performance penalties associated with existing DTM techniques can be reduced while the required expected chip lifetime is maintained [63, 64].

1.3 Research Contributions

In this dissertation, we propose improved runtime management techniques to address the increasing power and thermal challenges. Our results show that substantial performance and power margins can be reclaimed by exploiting runtime workload variations rather than worst-case assumptions. At the same time, we also develop design-time models suitable for reclaiming design margins. Specifically, we make contributions in the following three areas.

1. **Architectural power managements using dynamic voltage scaling (DVS).**

   - Develop optimization algorithms to find the optimal intra-task DVS scheduling using statistical workload information [66].
   - Develop a modeling framework for multimedia playback systems, and propose a feedback control technique to adapt the decoder speed to the required workload throughput [59, 60, 62, 65]. To our knowledge, our result is among the first who apply control-theoretic techniques to reduce energy(power) in real-time systems [59, 60].
   - Implement an energy-efficient software multimedia player on a Linux platform equipped with DVS-capable processor, and evaluate various low power decoding techniques using a set of multimedia streams in different video compression formats [65].

2. **Chip interconnect electromigration reliability modeling in deep sub-micron (DSM) design.** Chip interconnect electromigration (EM) is one of the major temperature enhanced failure mechanisms in DSM design. As technology advances, on

chip interconnects are subject to substantial both temporal and spacial thermal gradients. We are among the first to investigate the impact of thermal gradients on EM reliability.

- Propose a new electromigration (EM) lifetime model for chip interconnects subject to temporal temperature and current variations. The model reveals that lifetime can be modeled as a resource to be consumed at a temperature dependent rate [61].
- Propose a method to estimate EM lifetime for interconnects subject to non-uniform temperature distribution across the interconnects.

3. **Dynamic reliability management for high performance computing systems.**

   - Evaluate the thermal impacts of different workloads on a high performance processor using architectural performance and power simulators [63, 64].
   - Propose runtime lifetime banking techniques for dynamic reliability management for high performance systems such that the performance of the system is maximized without harming the expected lifetime [63, 64].

Note that, in the above research outcomes, the techniques for power management and reliability management are complementary to each other. Reducing power (thus, temperature) can relieve the thermal constraint on performance. With low power techniques existing, reliability management techniques can further reclaim performance loss due to traditional DTM techniques.

1.4 Dissertation Roadmap

The rest of the dissertation is organized as follows. Chapter 2 introduces some technology background of this research as well as the state of the art in the related research found from the literature. Chapter 3, 4 and 5 detail our power-aware runtime management techniques for real-time systems, where battery lifetime is the major concerns. Chapter 6 presents a new electromigration model to predict interconnect lifetime subject to temporal and spatial thermal gradients, which becomes increasingly important in DSM technology. Chapter 7 applies this new EM model in temperature-aware dynamic reliability management techniques for high performance systems where temperature is becoming the limiting factor for performance. Finally Chapter 8 discusses the possible extensions of the techniques proposed in this dissertation.

Chapter 2

Background and Related Research

In this chapter, we introduce the concept of dynamic voltage/frequency scaling (DVS) which serves as the basic vehicle for low power runtime management in this dissertation. We will describe some related researches using DVS in different applications. Then we will introduce the application of control theory in hardware optimizations. As for temperature induced circuit reliability issue, we will introduce electromigration (EM) which is one of the most important hard failure mechanisms facing modern ICs. We will also describe current approaches for reliability managements.

2.1 Dynamic Voltage/Frequency Scaling

Circuit-level techniques have been a mainstay of power reduction for some years, but recently much research attention has focused on system-level techniques. The benefits of approaching power reduction at the system-level are typically synergistic with circuit-level techniques, and higher design abstraction levels have more direct knowledge about the workload and can control large portions of the computer system accordingly. Yet for real-time systems, performance should still be guaranteed even when power reduction techniques are in place. Since task scheduling in computing systems is a key lever for both performance and power, it is a natural target for research on power-aware computing. There are two well studied power reduction techniques that have impacts on system scheduling [49]: DVS (dynamic voltage scaling) and DPM (dynamic power management). In DVS, different computational tasks are run at different voltages and clock frequencies while still providing an adequate level of performance. DPM aims to shut off or change the power/performance state of system parts (inside or outside the CPU) that are not in use at any given time [11]. Execution bandwidth throttling has also recently been proposed, in which the number of instructions fetched per cycle is reduced [87] or the frequency of fetch operations is varied [30, 93]. Structure resizing for power reduction has also been considered [78].

Let's define a variable r, the frequency scaling factor with a value within $[0,1]$, which represents the actual speed at which the processor will run (each speed (working frequency) is associated with an optimum operating voltage). For example, $r = 1$ means the decoder will run at its full speed, while $r = 0.5$ is half speed. We adopt the approximate energy calculation proposed in [92] in this research. And the dynamic energy consumption of a task can be estimated by [92]:

$$E(r) = CV_0^2 T_s f_{ref} \left[\frac{V_t}{V_0} + \frac{r}{2} + \sqrt{r\frac{V_t}{V_0} + \left(\frac{r}{2}\right)^2} \right]^2 \tag{2.1}$$

where C is the average switched capacitance per cycle, T_s is the sample period of a DSP system, f_{ref} is the operating frequency at V_{ref}, r is the frequency factor, V_t is the threshold voltage and $V_0 = (V_{ref} - V_t)^2$. For a DVS system, C, T_s, V_t and V_0 are unchanged, and one can obtain the following reduced quadratic power consumption model [92]:

$$E(r) = r^2 E_0 \tag{2.2}$$

where E_0 is the energy consumption for the task executed at full speed, r is the frequency scaling factor, and $E(r)$ is the actual energy consumption at a speed specified by r with the corresponding minimum operating voltage. Equation 2.2 indicates that the energy consumption for a task can be reduced using a slower clock frequency.

Since DVS is very effective in reducing energy consumption, the applications of DVS have recently become a very active research field and various techniques have been proposed. There are several ways to classify different DVS approaches. Depending on when the DVS decisions are made, DVS techniques can be classified as static or dynamic. With the aid of compiler analysis, some techniques identify the code portions in a program where the CPU clock can be slowed down [42, 112] before the actual program execution. While in many applications, detailed off-line analysis is luxurious, DVS decisions have to be made on line [104, 119]. DVS techniques can also be divided by the performance models assumed. Some techniques assume that the task execution time can be predicted accurately [83], others take a more conservative approach by only using the worst case execution time (WCET) [77]. Given that the former approach is not realistic and the latter is too conservative, techniques using statistical information on task execution time are proposed [35, 52, 115]. DVS techniques can also be distinguished by the granularity of DVS decisions. Inter-task DVS techniques [91] execute the task with a single speed and may change the speed at the task boundary while Intra-task DVS techniques [35,42,52,112,115] change the clock speed along task execution. With coarser DVS granularity, DVS overheads (both energy and performance) are generally negligible compared to the task execution [62]. While in fine-grain DVS techniques, DVS overheads become significant and need to be addressed carefully [66, 72].

In a multimedia system, the decode time for each frame in a multimedia stream is not necessarily uniform. For example, MPEG frames come in three different coding types (intra (I), bi-directional (B), and predictive (P)), each of which requires different decoding effort. Even within these coding types, frame decode time varies. Therefore, DVS is quite suitable for multimedia systems. Simunic *et al.* [91] proposed a *change point detection algorithm* based scheme, which uses online statistical maximum-likelihood analysis to detect changes in aggregate behavior for a stream with an exponential distribution in packet arrival and frame decoding time. They evaluate the algorithm using a system model that consists of an SA-1100 processor with an MP3 and MPEG workload. This work is similar to that of [79], which implements the workload of an H.263 video benchmark with frequency scaling on an SA-1100. Another technique for saving energy in multimedia applications using architectural adaptation and frequency scaling was introduced in [44]. However, that technique uses profiling to predict energy per instruction and instructions per frame statistics. Architectural and frequency adaptations are then set to ensure frames meet their deadlines by adjusting the frequency to reduce slack time between frames. Other techniques focus on prediction accuracy on frame decoding time among which Chung *et al.* [26] suggested the server side of the multimedia stream provides timing information about decoding, and Choi *et al.* estimated decoding time for each frame by filtering [23] and later with the help of performance counters [24]. In many of these schemes, the efficiency of DVS is largely dependent on the prediction accuracy.

The above approaches change the voltage/frequency at the frame (task) boundary. In general, the actual task execution time is hard to predict, and therefore, the worst case execution time (WCET) is usually used to find the inter-task voltage/frequency schedule for real-time systems. However, this approach does not yield good energy saving, because the task actual execution time is much smaller than WCET. To address this issue, Yuan *et al.* [115] applied the intra-task DVS idea to a low power multimedia system in which the distribution of frame decoding time is obtained in runtime during video playback.

2.2 Application of Control Theory in Computing Systems

The design of control systems is a mature field with a history dating back at least as far as the 1600s. Numerous textbooks exist that describe basic control principles, e.g. [33, 34]. Control-theoretic approaches have been applied to a variety of computer system design aspects outside the computer architecture realm, including CPU scheduling [56, 100], web server quality-of-service management [53, 58], Internet congestion control [39], and data migration [54]. At the circuit level, feedback control is used for voltage scaling [31] and current canceling for leakage control [110].

At the time when we tried to apply feedback control technique in a DVS multimedia system, the only use of *formal* feedback control theory that we were aware of in computer architecture literature was some work on temperature regulation [93] and cache decay [105]. After our work [59, 60] demonstrated the benefit of feedback control in DVS, there are several other papers using feedback control to guide DVS in various applications [90, 104, 119]. In a formal feedback control based DVS scheme, a PID (Proportional, Integral and Derivative) controller is usually used for new frequency/voltage decision. Because of the robustness of the PID controller, it can be well applied to those complex systems even without accurate models. In the case for multimedia decoding, no explicit decoding time predictions are needed in the feedback control based scheme, and the controller can adjust the output to runtime variations of the working environment.

2.3 Reliability Modeling and Management

2.3.1 Interconnect electromigration

Due to increasing complexity and clock frequency, temperature has become a major concern in integrated circuit design. Higher temperatures not only degrade system performance, raise packaging costs, and increase leakage power, but they also reduce system reliability via temperature enhanced failure mechanisms such as gate oxide breakdown, interconnect fast thermal cycling, stress-migration and electromigration (EM) [10]. Besides runtime efforts to reduce the power consumption of a system (therefore temperature), special attention has to be paid for runtime management in order to guarantee the expected system lifetime. Using electromigration as an example, we investigate how runtime parameter changes such as voltage and temperature affect the system reliability, and how to guide runtime management to maximize the system performance without reliability violations.

EM is a process of self-diffusion due to the momentum exchange between electrons and atoms in the metal interconnects. As a result of electromigration, short(open)-circuit failures will occur due to the formation of hillocks(voids) in the interconnects. Clement [28] provided a review of 1-D analytic EM models describing the diffusion process. Several more sophisticated EM models are also available [76, 88]. In our research, we adopt the EM-induced stress build-up model of Clement and Korhonen [27, 51], which has been widely used in EM analysis and agrees well with simulation results using more advanced models such as that by Ye *et al.* [114]. Historically, Black [14] proposed a semi-empirical temperature-dependent equation to predict interconnect lifetime due to EM failures:

$$T_f = \frac{A}{j^n} exp\left(\frac{Q}{kT}\right) \tag{2.3}$$

where T_f is the time to failure, A is a constant based on the interconnect geometry and material, j is the current density, Q is the activation energy (e.g., $0.6eV$ for aluminum), and kT is the thermal energy. The current exponent, n, has different values according to the actual failure mechanism. It is assumed that $n = 2$ for void nucleation limited failure and $n = 1$ for void growth limited failure [76]. Black's equation is widely used in thermal reliability analysis and design. For example, Hunter [45, 46] derived a self-consistent allowable current density upper bound for achieving a reliability goal by taking into account interconnect self-heating effects using Black's equation.

2.3.2 Thermal management in computing systems

As overheating induces hardware failures, thereby limiting/under-utilizing the possible performance, the field of temperature-aware design has recently emerged to maximize system performance under lifetime constraints. Dynamic thermal management (DTM) techniques [95, 97] are being developed. Currently, DTM studies assume a fixed maximum temperature threshold, which, for example, is derived from Black's equation to provide a lifetime budget. During runtime, when the predefined temperature threshold is reached, DTM will engages certain cooling mechanisms, such as frequency/voltage scaling and throttling. Therefore operating temperature is always bounded and lifetime is assured, at the expense of degraded performance. However, Black's equation assumes a constant temperature. Thus, a worst-case temperature profile is usually used when predicting interconnect lifetime, resulting in pessimistic estimations and unnecessarily restricted design spaces. For example, circuit designer has to apply more conservative design methodologies, under such pessimistic assumptions, while in DTM cooling mechanisms may often be engaged (and performance penalties incurred) unnecessarily. To better evaluate different thermal management techniques and to explore the design space, designers need better information about the lifetime impact of temperature. To our knowledge, the only solutions currently available to answer this question are simulations, which are very time-consuming and not suitable for runtime reliability management [114]. Recently, Srinivasan *et al.* [97] proposed an architecture-level dynamic reliability model, but their model does not have a solid physical foundation on the impact of time-varying stresses on reliability.

2.3.3 An architectural thermal model

While the dynamic temperature profile of a system is workload-dependent [95, 97], several efficient and accurate techniques have been proposed to simulate transient chip-wide temperature distribution [21, 95, 107], providing design-time and runtime knowledge of the thermal behavior of different design alternatives.

In this research, we use prevalent architectural performance and power simulators (i.e., *SimpleScalar* and *Wattch*) [17, 18] to observe the workload variations, and add the *Hotspot* model to obtain thermal variations during runtime. *Hotspot* [43, 95] is an architectural compact thermal model, which is verified against using finite element method as well as an industry chip design. It uses a thermal RC network to calculate the temperatures at various locations of the chip. It is a highly parameterized model in the sense that it can easily model different combination of materials, layout, or thermal package. It can be easily interfaced with the power/performance model and provide both transient and steady state temperature. Its efficiency, accuracy and flexibility make it especially suitable for architecture-level thermal analysis. Thus, *Hotspot* is chosen as the primary thermal analysis tool in this research.

Chapter 3

Power-aware Runtime Management using Procrastinating Voltage Scheduling [1]

3.1 Introduction

With the scaling of semiconductor technology, power consumption has become a serious issue for both high performance and embedded systems [73]. Dynamic voltage scaling (DVS) has become an efficient technique for power reduction due to the quadratic dependence of circuit switching energy on operating voltage.

However, DVS trades off performance for energy. Many researchers propose various voltage scheduling techniques such that energy is minimized while performance is still guaranteed. One of the major difference among these techniques lies in their workload assumptions. For example, Rao and Vrudhula [83] assume the exact amount of computation is known in advance. In reality, different instances of the same task might have various computation requirements. Therefore some work such as [101] uses the worst-case execution time (WCET) to schedule the voltage profile offline and reclaim the slacks on runtime,

[1]This chapter is based on the published paper titled “Procrastinating Voltage Scheduling with Discrete Frequency Sets” [66].

while others [62] and [119] use formal feedback controller to adapt the system to the workload variations in runtime. On the other hand, the amount of computation for the same task could be well characterized by probability density functions. This observation inspires probabilistic approaches in DVS studies [35, 52, 113, 115, 117].

Lorch and Smith [52] show that using statistical information in DVS is superior to other heuristic DVS techniques whose performances are strongly dependent on the workload distribution. In DVS with probabilistic workload information, a good strategy is to begin with a low frequency, and increase the frequency gradually as the task progresses, such that the task can be finished before deadline even in the worst-case. In other words, high voltage (power) operating points are procrastinated during task execution. Following [117], we call this form of voltage scheduling procrastinating DVS in the rest of the chapter. Jejurikar *et al.* [48] introduce the concept of "procrastination scheduling" which is different from the subject studied here. They focus on inter-task scheduling policies and look for opportunities to delay the execution of tasks such that system shutdown intervals can be increased, thus minimizing leakage energy consumption, while we study intra-task scheduling technique.

The optimal procrastinating DVS for single task is derived in [35, 52, 115] using an ideal processor model. Our previous work [117][2] extends their solutions to deal with multiple tasks. However, there are several limitations on these prior works. First they assume voltage/frequency can be continuously scaled, and have to round their solutions to the available frequencies in the real system. The rounding could result in energy-inefficient design when the number of available frequency is relatively small [101]. Second, the overhead of frequency/voltage transition is ignored, which is dangerous for real-time systems and leads to non-optimality in practical systems [9, 72]. Third, system-wide power consumption is

[2]This work is done jointly by Yan Zhang and Zhijian Lu, with the theoretical analysis carried out by Lu and the experiments implemented by Zhang.

not explicitly modeled. In this work, we try to address these practical issues. Recently, Xu *et al.* [113] try to solve the same problem. They adopt a search-based approach to find the optimal scheme, while our solution is analytic in nature and solves the problem very efficiently.

Specifically, in this chapter, we make the following contributions. First, we derive an analytical solution for systems with non-ideal processors when using procrastinating DVS, enabling fast voltage scheduling with arbitrary workload distribution. Our analysis does not assume any specific frequency-voltage relationship (i.e. analytic models), making it suitable for various systems and different voltage scaling techniques, such as combined supply voltage and body bias scaling [9]. Second, our solutions minimize the total system energy consumption by including both dynamic and static on-chip power as well as off-chip component power. Third, our results indicate that all efficient operating points (i.e. voltage/frequency pairs) in procrastinating DVS must lie on a *convex* energy-delay curve. This interesting observation is helpful in low power system design by avoiding inefficient frequency sets. Finally, we find that, for a given deadline, a small number of frequencies are sufficient for forming a schedule to maintain energy savings comparable to that of using all frequencies while avoiding unnecessary transition overheads.

The rest of this chapter is structured as follows. In Section 3.2, the system model is described and the optimization problem is formulated. We solve the problem in Section 3.3, and extend it to account for frequency switching overheads in Section 3.4. We present simulation results in Section 3.5 and summarize this chapter in Section 3.6.

3.2 System Model and Problem Formulation

3.2.1 Energy Model

In many low-power systems, there are usually two power states: active state and idle state. In the idle state, the system is in a low-power mode with zero clock frequency, and no useful work can be done. Therefore, we model the system power by $P_{sys} = P_f + P_0$, where P_0 is the idle power which is a constant and exists whenever the system is powered on, and P_f is the power used for computation in active state, which is the sum of on-chip and off-chip power consumption, including both leakage and dynamic power. In general, P_f is dependent both on supply voltage and clock frequency. We define energy efficiency $e(f) = \frac{P_f}{f}$, which represents the energy spent on each cycle for computation. It is obvious that energy efficiency is also voltage/frequency dependent. If a task with X cycles is finished before its deadline D, the total energy spent in the period D is $E_{sys} = \int_0^X e(f)dx + P_0 D$. The second term in the total energy is independent on voltage/frequency scaling. In the following discussion, we will ignore P_0 as if it is zero.

The system is capable of operating on variable voltages/frequencies. Let $\{f_1, f_2, f_3, \ldots, f_r\}$ denotes the set of available frequencies, with total number equal to r, and we have $f_1 < f_2 < f_3, \cdots < f_r$. For each frequency point, there is a voltage and power consumption associated with it, and thus a corresponding energy efficiency under that frequency. Therefore, we have a set of energy efficiencies $e_1, e_2, e_3, \ldots, e_r$, with $e_1 < e_2 < e_3 < \cdots < e_r$. That is true because if $f_i < f_j$ and $e_i \geq e_j$, f_j can finish tasks faster while spend less energy than f_i, and f_i becomes inefficient and never used. *Therefore, in a usable frequency set, energy efficiency should be an increasing function of clock frequency, or decreasing function of clock period.*

3.2.2 Procrastinating DVS

A task executed on the system has a deadline D and its actual amount of computation (clock cycles) is randomly distributed between 0 and X_{max} and governed by a probability density function (PDF)[3], which can be obtained, for example, from online profiling. Furthermore, we assume frequency/voltage switching points can be inserted anywhere during task execution. Figure 3.1 gives an example of procrastinating DVS along with its workload distribution.

Theorem 3.1. *In the optimal procrastinating DVS, the operating frequency is non-decreasing as the number of executed cycles increases.*

We omit the proof for Theorem 3.1, because Xu *et al.* [113] presented a similar theorem and proved it in their paper. Theorem 3.1 indicates that in a system with r operating points, the optimal procrastinating DVS scheduling has at most r frequency/voltage transitions, as shown in Figure 3.1(b) for a system with 4 usable frequency/voltage levels. *Therefore the key for the optimal scheduling with discrete frequency sets is to identify the positions where transitions from low frequency to high frequency occur.*

Xu *et al.* [113] assume that frequency transitions can only happen at some fixed points (e.g. every 10K cycles) and try to find the optimal scheduling by searching a space composed of those transition points. We take a different approach. Observing that a task usually has millions of cycles while the number of frequency transitions is much smaller (i.e. equal to the number of the operating points, which is usually less than 50), the number of cycles executed on each frequency has much finer granularity and can be approximated as continuous. For example, in Figure 3.1, we use real variables $X_{max} - X'_4 - X'_3 - X'_2$, X'_2, X'_3 and X'_4 to represent the number of cycles executed at f_1, f_2, f_3 and f_4 respectively. And

[3]We assume the number of cycles executed in each task instance is invariable in spite of the operating frequency. This is a reasonable assumption for CPU-bounded applications. How to model the frequency dependence of clock cycles in memory-bounded applications is left for future investigation.

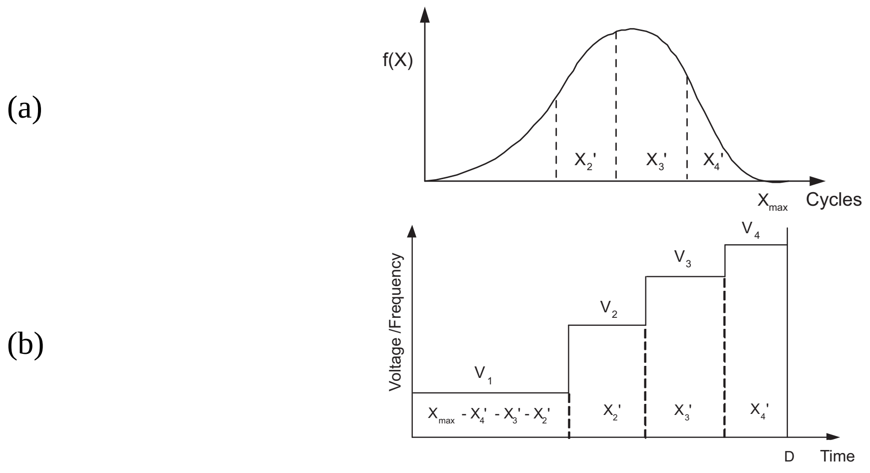


Figure 3.1: (a) PDF of task execution cycles. (b) Procrastinating DVS with 4 operating points. [66]

the expected energy consumption could be modeled as:

$$\begin{aligned}
&E(X_2',X_3',X_4') \\
=\;& e_1 \int_0^{X_{\max}-X_2'-X_3'-X_4'} (1-F(y))\,dy \\
&+e_2 \int_0^{X_2'} \left(1-F(X_{\max}-X_4'-X_3'-X_2'+y)\right) dy \\
&+e_3 \int_0^{X_3'} \left(1-F(X_{\max}-X_4'-X_3'+y)\right) dy \\
&+e_4 \int_0^{X_4'} (1-F(X_{\max}-X_4'+y))\,dy
\end{aligned} \tag{3.1}$$

where e_i is the energy efficiency (i.e. energy/cycle) associated with frequency f_i, as defined at the beginning of this section. $F(x)$ is the cumulative distribution function (CDF) of task execution cycles. Therefore, the optimal procrastinating DVS problem can be formulated as a mathematical optimization problem:

$$\begin{aligned}
&\text{Minimize} && E(X_2',X_3',X_4') \\
&\text{Subject to} && \tfrac{X_{\max}-(X_2'+X_3'+X_4')}{f_1}+\tfrac{X_2'}{f_2}+\tfrac{X_3'}{f_3}+\tfrac{X_4'}{f_4}\le D
\end{aligned}$$

For simplicity, in the following, we use this example in our calculations, but the results are readily generalizable to systems with r operating points.

3.3 Solutions for Discrete Frequency Sets

In this section, the optimal procrastinating DVS with discrete frequency sets is solved without considering frequency transition overheads. In the next section, we extend our discussion to account for those overheads.

3.3.1 Optimal Procrastinating DVS

We apply the Lagrange Multiplier Method to find the number of cycles executed with each operating points. Let

$$\begin{aligned} & L(X_2', X_3', X_4', \lambda) \\ & = E(X_2', X_3', X_4') \\ & + \lambda\left(\frac{X_4'}{f_4} + \frac{X_2'}{f_2} + \frac{X_3'}{f_3} + \frac{X_{\max} - (X_2' + X_3' + X_4')}{f_1} - D\right) \end{aligned}$$

We have

$$\begin{aligned} \frac{\partial L}{\partial \lambda} &= \frac{X_4'}{f_4} + \frac{X_2'}{f_2} + \frac{X_3'}{f_3} + \frac{X_{\max} - (X_2' + X_3' + X_4')}{f_1} - D = 0 \\ \frac{\partial L}{\partial X_4'} &= \frac{\partial E(X_2', X_3', X_4')}{\partial X_4'} + \lambda\left(\frac{1}{f_4} - \frac{1}{f_1}\right) = 0 \\ \frac{\partial L}{\partial X_2'} &= \frac{\partial E(X_2', X_3', X_4')}{\partial X_2'} + \lambda\left(\frac{1}{f_2} - \frac{1}{f_1}\right) = 0 \\ \frac{\partial L}{\partial X_3'} &= \frac{\partial E(X_2', X_3', X_4')}{\partial X_3'} + \lambda\left(\frac{1}{f_3} - \frac{1}{f_1}\right) = 0 \end{aligned} \tag{3.2}$$

By substituting Equation (3.1) into (3.2) and after some simplifications, we obtain:

$$\left(1-F(X_{\max}-X'_4-X'_3-X'_2)\right) = \lambda\frac{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)}{(e_2-e_1)} \tag{3.3}$$

$$(1-F(X_{\max}-X'_4-X'_3)) = \lambda\frac{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)}{(e_3-e_2)} \tag{3.4}$$

$$(1-F(X_{\max}-X'_4)) = \lambda\frac{\left(\frac{1}{f_3}-\frac{1}{f_4}\right)}{(e_4-e_3)} \tag{3.5}$$

$$\frac{X_{\max}-(X'_2+X'_3+X'_4)}{f_1}+\frac{X'_2}{f_2}+\frac{X'_3}{f_3}+\frac{X'_4}{f_4} = D \tag{3.6}$$

Therefore, given a deadline *D*, the optimal procrastinating DVS schedule can be found by solving Equations (3.3-3.6). Transformations of Equations (3.3-3.5) reveal an interesting property of the optimal solution, as illustrated in Figure 3.2. The frequency transition points partition the area of the probability density function such that

$$\begin{aligned}
&\text{area}(II+III+IV) : \text{area}(III+IV) : \text{area}(III+IV)\\
= \;& \left(1-F(X_{\max}-X'_4-X'_3-X'_2)\right)\\
& :(1-F(X_{\max}-X'_4-X'_3))\\
& :(1-F(X_{\max}-X'_4))\\
= \;& \frac{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)}{(e_2-e_1)} : \frac{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)}{(e_3-e_2)} : \frac{\left(\frac{1}{f_3}-\frac{1}{f_4}\right)}{(e_4-e_3)}
\end{aligned}$$

Though it might not be straightforward to solve Equations (3.3-3.6) for any given deadline, once the value of λ is determined, solutions can be readily obtained. Here we discuss several special cases for λ, which will be helpful for finding the value of λ for a given deadline.

$\lambda_1 = 0$**.** It follows that $X'_2 = X'_3 = X'_4 = 0$. Let $D_1 = \frac{X_{max}}{f_1}$. The deadline *D* in this case is equal to D_1, and the areas of region (II–IV) are equal to 0.

$\lambda_2 = \frac{(e_2 - e_1)}{\left(\frac{1}{f_1} - \frac{1}{f_2}\right)}$. As the deadline shrinks from D_1, higher operating points have to be applied. The area of regions (II–IV) will increase from 0 and keep a constant ratio among them. Finally, the sum of these three regions becomes equal to the total area of the PDF. When this happens, there will be no cycles assigned to f_1. Let D_2 denote the deadline in this case.

$\lambda_3 = \frac{(e_3 - e_2)}{\left(\frac{1}{f_2} - \frac{1}{f_3}\right)}$. As the deadline reduces further from D_2, regions (III and IV) keep increasing and finally occupy the whole area of the PDF. In this case, no cycles are allocated to f_1 and f_2. Let D_3 denote the deadline in this case.

$\lambda_4 = \frac{(e_4 - e_3)}{\left(\frac{1}{f_3} - \frac{1}{f_4}\right)}$. When the deadline shrinks more from D_3, the area of region (IV) increases and finally occupies the whole PDF. In this case, only f_4 is scheduled. Let $D_4 = \frac{X_{max}}{f_4}$, which is the minimum deadline this system can satisfy.

In other words, when $D_1 \leq D$, the whole task is executed with f_1, and $\lambda(D) = \lambda_1 = 0$. When $D_2 < D < D_1$, all four frequencies are scheduled along task execution, and $\lambda_1 < \lambda(D) < \lambda_2$. When $D_3 < D \leq D_2$, only f_2, f_3 and f_4 are scheduled, and $\lambda_2 \leq \lambda(D) < \lambda_3$..., and so on. Finally when $D = D_4$, only f_4 is used to execute the whole task. The value of λ increases from its minimum $\lambda_{min} = 0$ to its maximum $\lambda_{max} = \frac{(e_4 - e_3)}{\left(\frac{1}{f_3} - \frac{1}{f_4}\right)}$ as the deadline decreases. In fact, it can be shown that $\frac{d\lambda}{dD} \leq 0$. We will know the numerical range of λ when a deadline is given, by comparing the actual deadline to D_1 through D_4, as well as the operating points to be used in the schedule. In practice, we can represent this numerical range of λ by a set of constant numbers (samples of this range) and find the approximate numerical value of λ very efficiently by the bisection method. Thus, the complexity of our method is O(1) with respect to the number of bins in the workload distribution histogram and linear to the number of available frequencies/voltages.

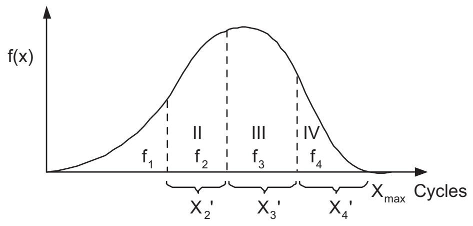


Figure 3.2: Graphical interpretations of Equations (3.3), (3.4) and (3.5). [66]

The significance of λ can be seen as follows. With optimal procrastinating DVS, the expected energy is a function of deadline. It follows

$$\begin{aligned}
&\frac{dE\left(X_2'(D),X_3'(D),X_4'(D)\right)}{dD} = \frac{\partial E}{\partial X_2'}\frac{dX_2'}{dD} + \frac{\partial E}{\partial X_3'}\frac{dX_3'}{dD} + \frac{\partial E}{\partial X_4'}\frac{dX_4'}{dD} \\
&= \lambda\left[\left(\frac{1}{f_1}-\frac{1}{f_2}\right)\frac{dX_2'}{dD} + \left(\frac{1}{f_1}-\frac{1}{f_3}\right)\frac{dX_3'}{dD} + \left(\frac{1}{f_1}-\frac{1}{f_4}\right)\frac{dX_4'}{dD}\right] \\
&(\text{using Equation (3.2)}) \\
&= -\lambda(D)
\end{aligned} \tag{3.7}$$

Because of

$$\left(\frac{1}{f_1}-\frac{1}{f_2}\right)\frac{dX_2'}{dD} + \left(\frac{1}{f_1}-\frac{1}{f_3}\right)\frac{dX_3'}{dD} + \left(\frac{1}{f_1}-\frac{1}{f_4}\right)\frac{dX_4'}{dD} = -1$$

, from Equation (3.2). Equation (3.7) reveals that λ is the energy sensitivity over the deadline. For example, as we discussed earlier, when $D = \frac{X_{max}}{f_1}$, $\lambda = 0$. This means that further increasing the deadline does not save energy anymore, which is true because the lowest frequency is already applied to execute the whole task.

3.3.2 Efficient Operating Points

As shown in Figure 3.2, Equations (3.3-3.5) implicitly require that, for $f_{i-1} < f_i < f_{i+1}$,

$$\frac{(e_i - e_{i-1})}{\left(\frac{1}{f_{i-1}} - \frac{1}{f_i}\right)} < \frac{(e_{i+1} - e_i)}{\left(\frac{1}{f_i} - \frac{1}{f_{i+1}}\right)} \tag{3.8}$$

In fact, we have the following theorem.

Theorem 3.2. *In a system with a set of usable operating points $\{f_1/e_1, f_2/e_2, f_3/e_3, \ldots, f_r/e_r\}$, where $f_1 < f_2 < f_3, \cdots < f_r$ and $e_1 < e_2 < e_3, \cdots < e_r$, if there exists an operating point f_i/e_i such that Equation (3.8) is not hold, there will be no cycles allocated to this operating point in the optimal procrastinating DVS.*

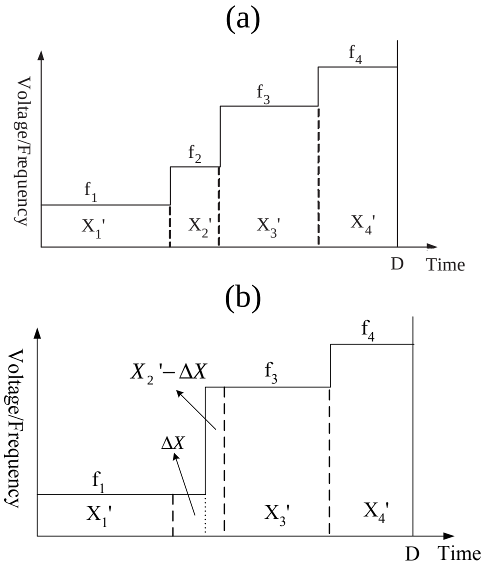


Figure 3.3: Proof for Theorem 3.2

Proof. For simplicity, assuming that we have 4 possible operating points: $f_1 < f_2 < f_3$ and $\frac{(e_2-e_1)}{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)} \geq \frac{(e_3-e_2)}{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)}$, one finds a voltage scheduling scheme including all operating points as shown in Figure 3.3(a). This scheduling scheme allocates X_2' cycles for f_2/e_2. Alternatively we can find another scheduling scheme which allocates ΔX cycles from X_2' to f_1/e_1, and $X_2' - \Delta X$ cycles to f_3/e_3, as shown in Figure 3.3(b). In order to avoid the timing violation, the transformation from Figure 3.3(a) to Figure 3.3(b) also requires that the time needed to finish X_2' cycles is unchanged, and it follows $\Delta X\left(\frac{1}{f_1}-\frac{1}{f_2}\right)=\left(X_2'-\Delta X\right)\left(\frac{1}{f_2}-\frac{1}{f_3}\right)$.

Due to this transformation, the energy consumed by the ΔX cycles is reduced by (denoted by E_{de}): $E_{de}=\int_{X_1'}^{X_1'+\Delta X}(1-F(x))(e2-e_1)dx \geq \left(1-F(X_1'+\Delta X)\right)(e_2-e_1)\Delta X$. On the other hand, the energy consumed by the $X_2' - \Delta X$ cycles is increased by (denoted by E_{in}): $E_{in} = \int_{X_1'+\Delta X}^{X_1'+X_2'} (1-F(x))(e_3-e_2)dx \leq \left(1-F(X_1'+\Delta X)\right)(e_3-e_2)\left(X_2'-\Delta X\right)$. And finally $\frac{(e_2-e_1)}{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)} > \frac{(e_3-e_2)}{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)} \Rightarrow \frac{E_{in}}{E_{de}} \leq 1$.

In summary, if there is an operating point for which Equation (3.8) is not true, one can always find a better scheduling by eliminating this operating point in procrastinating DVS. ■

Theorem (3.2) says that a usable operating point is not necessarily an efficient one. As an example, in Figure 3.4, we plot the energy (per cycle) vs. delay ($1/f$) curves for some existing processors with DVS capability including IBM PPC405LP ([113]), Intel Xscale ([113]), ARM8 ([47]), AMD Mobile Athlon (our own measurements), and an ideal processor using the "α current model" [86]. $\frac{(e_i-e_{i-1})}{\left(\frac{1}{f_{i-1}}-\frac{1}{f_i}\right)}$ represents the slope between two consecutive operating points in the curves. Equation (3.8) specifies that, in order for all operating points to be efficient, the slope of the energy–delay curve should decrease as the delay increases, or, all operating points should lie on a convex curve.

As one may expect, the energy efficiency decreases as the delay increases for all processor models, as illustrated by the curves in Figure 3.4. In the ideal processor model, we also include the static power by assuming that it is about one third of the dynamic power. Figure 3.4 shows that the operating points of the ideal processor are strictly on a convex curve. Xscale and ARM8 processors follow a similar trend with all operating points being efficient. While PowerPC and Mobile Athlon have some operating points that will never be used in the optimal procrastinating DVS, as indicated in the figure.

3.4 Considering Transition Overheads

In this section, we extend our analytical solutions to account for both energy and time overheads in frequency transitions. We first discuss energy overhead alone and then combine both time and energy overhead.

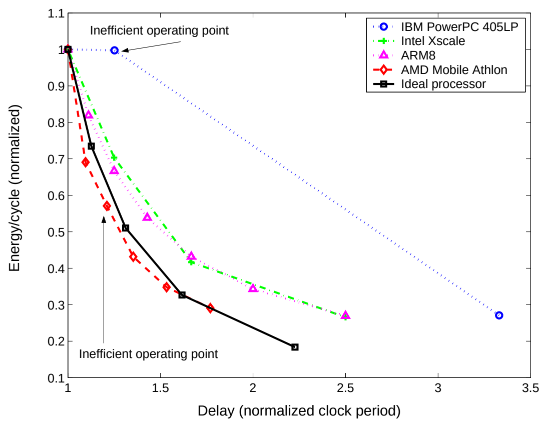


Figure 3.4: Energy–delay curves for different processors with DVS capability (The values for each processor are normalized to its energy and delay at the full speed.). [66]

3.4.1 Modeling Energy Overheads

As pointed out in [72], the energy overhead due to frequency switching includes idle energy and capacitance charging energy. The idle energy part is already captured by P_0 in Section 3.2. For capacitance charging energy, it is true that $E_{ij} = E_{ik} + E_{kj}$, where E_{ij} is the energy overhead when switching from f_i to f_j and $f_i < f_k < f_j$. Let E_{total} denote the average energy consumption with the existence of energy overheads, and, using a 4-frequency system as an example, it follows

$$\begin{aligned}
&E_{total} = E_{ideal} + E_{12}[1 - F(X_m - X_4' - X_3' - X_2')] \\
&+ E_{23}[1 - F(X_m - X_4' - X_3')] + E_{34}[1 - F(X_m - X_4')] \\
&+ E_{34}[1 - F(X_m - X_4')]
\end{aligned}$$

where E_{ideal} is the expression for expected energy without overheads (i.e. Equation (3.1)). Again, using the Lagrange Multiplier Method, the energy-optimal voltage scheduling is obtained by

$$\begin{aligned}
&\left(1-F(X_{\max}-X_4'-X_3'-X_2')\right)+\tfrac{E_{12}}{e_2-e_1}f(X_{\max}-X_4'-X_3'-X_2')\\
&=\lambda\frac{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)}{(e_2-e_1)}\\
&(1-F(X_{\max}-X_4'-X_3'))+\tfrac{E_{23}}{e_3-e_2}f(X_{\max}-X_4'-X_3')=\lambda\frac{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)}{(e_3-e_2)}\\
&(1-F(X_{\max}-X_4'))+\tfrac{E_{34}}{e_4-e_3}f(X_{\max}-X_4')=\lambda\frac{\left(\frac{1}{f_3}-\frac{1}{f_4}\right)}{(e_4-e_3)}\\
&\frac{X_{\max}-(X_2'+X_3'+X_4')}{f_1}+\frac{X_2'}{f_2}+\frac{X_3'}{f_3}+\frac{X_4'}{f_4}=D
\end{aligned}\tag{3.9}$$

Usually the energy overhead E_{ij} is small, e.g., using the data in [72], $\frac{E_{ij}}{e_j-e_i}\approx 7K(cycles)$, thus $\frac{E_{ij}}{e_j-e_i}f(x)\approx F(x)-F\left(x-\frac{E_{ij}}{e_j-e_i}\right)$. Letting $\phi(x)=F(x-\frac{E_{ij}}{e_j-e_i})$, Equation (3.9) can be reduced to the same form as Equations (3.3–3.6) by substituting $F(x)$ with $\phi(x)$.

3.4.2 Combined Overheads

Transition time overheads are hard to model using continuous functions. Eliminating an operating point in the scheduling might increase the energy, but it also increases the allowed execution time by releasing the transition time, thus reducing energy. When the number of available operating points is small. We can adopt a brute-force approach. Using the 4-frequency system as an example, we first assume all frequencies are applied and the actual deadline is adjusted by $D_{actual}=D-3t_o$, where t_o is the time overhead for one transition. We can find the optimal scheduling in this case by Equation (3.9). Then we allow only two transitions to happen in task execution, and we solve Equation (3.9) for all combinations of three operating points, similarly for one transition or even no transitions. Finally we choose the schedule with the lowest energy consumption. This brute-force approach requires $O(2^r)$ (r is the number of potential operating points) iterations of solving Equation (3.9) and is therefore not effective when r is large.

When the number of efficient operating points is large, we propose a heuristic search algorithm based on dynamic programming. We explain the key ideas behind our algorithm. First, we find a schedule using all frequencies by Equation (3.9), with the deadline adjusted by the transition times. This is the baseline schedule. Second, we define $energy\_reduction(f_i)$ as the energy reduction after the elimination of f_i from the baseline schedule. When f_i and f_j are not consecutive in the baseline, it is approximated that $energy\_reduction(f_i, f_j) = energy\_reduction(f_i) + energy\_reduction(f_j)$. Therefore, if $energy\_reduction(f_i) < energy\_reduction(f_k)$, it is established that $energy\_reduction(f_i, f_j) < energy\_reduction(f_k, f_j)$ when f_j is not next to either f_i or f_k in the baseline schedule. Thereby, we can reduce the search space without examining the elimination of both f_i and f_j from the baseline. The complexity of our algorithm is $O(r^3)$. In the next section, we use this algorithm to find schedules with a subset of frequencies when the number of transitions is fixed.

3.5 Experimental Results

In this section, we compare the proposed method with previous techniques. Since transition overheads were not considered in most prior work, in our experiments, we also ignore energy/time overheads for all schemes in comparison. The simulations use a processor model that provides finite frequencies linearly spaced between $500MHz$ and $1.5GHz$, and a power model $P = af_{op}^3$. We assume the maximum number of cycles in a task is $1,000,000$, and generate 5000 task instances for each different workload distribution. The PDFs for the voltage scheduling are obtained from profiling those 5000 task instances using a histogram with $10K$ cycles in each bins.

Figure 3.5 presents the average task energy of different task types (distributions) by simulating the execution of each task with different scheduling techniques. The deadline for all tasks is set to $0.00125s$. Three techniques are compared: 1) DVS with continuous voltage scaling between the maximum and minimum frequencies [35, 52], 2) Continuous

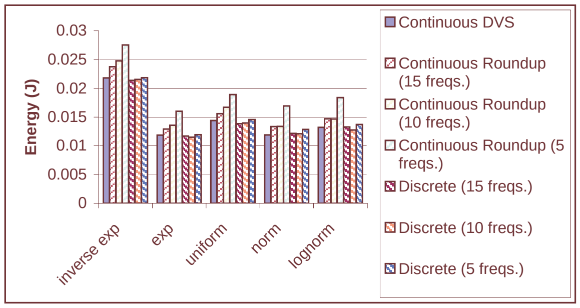


Figure 3.5: Energy consumption by different scheduling techniques for workloads with different cycle count distributions. [66]

scaling rounded up to the available frequencies, used in prior work [115, 117] to apply procrastinating DVS in practical systems, 3) Optimal procrastinating DVS with discrete frequency sets as proposed in this chapter. As indicated by Figure 3.5, the performance of simple rounding method degrades very quickly as the number of available frequencies is reduced. On the other hand, our solution is quite robust across different numbers of available frequencies. On average, the energy savings brought by our method over the previous rounding method range between 10% and 24% from a 15-frequency set system to a 5-frequency set system. Table 3.1 lists the energy saving by our method over simple rounding method.

workload distribution	15 freqs.	10 freqs.	5 freqs.
inverse exp	9.8%	13.0%	20.7%
exp	9.4%	15.3%	25.5%
uniform	11.3%	16.4%	23.0%
norm	8.9%	9.7%	24.2%
lognorm	9.8%	13.0%	25.5%

Table 3.1: Energy saving by our method over the simple rounding method with different number of available frequencies.

For a processor with r efficient operating points, the optimal procrastinating DVS will use by default all frequencies if necessary. A heuristic way to account for switching overheads is to limit the number of frequencies used in the schedule by, say, i frequencies.

Figure 3.6 plots the average energy consumption for tasks of normal distributions at different deadlines with the number of frequencies used in a schedule being fixed to 6, 4, 3, and 1, respectively, though the processor is capable of 10 different frequencies. This figure implies that, for a given deadline, a small number of carefully selected frequencies can form a schedule achieving energy savings very close to that of a schedule consisting of the full frequency set (as shown by the right y–axis). Similar results are observed for other workload distributions.

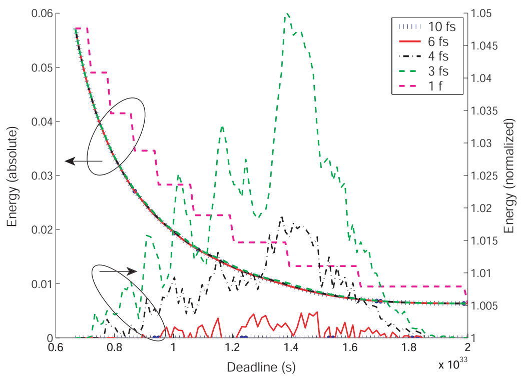


Figure 3.6: Average task energy consumption at various deadlines when the number of frequencies in a schedule is fixed (The y–axis on the right shows the energy values normalized to those of using the full frequency set (10 fs).). [66]

When the total number of efficient frequencies is large, it is computationally expensive to find a good subset of frequencies for scheduling when the allowed number of transitions is fixed. We compare our dynamic programming based heuristic method (subsection 3.4.2) to a brute-force approach. For a system model with 10 total frequencies, experimental results show that our heuristic method is 5 times faster to find “good” schedules for different numbers of allowed frequencies. The average task energy consumption using the schedules found by our method is very close to that found by the brute-force method, and the maximum deviation we observed is within 2%. In fact, the data presented in Figure 3.6 are based on the schedules found by our heuristic method. Table 3.2 presents the computation time needed for the brute-force method and our huristic method to find a voltage schedule for different task workload distributions, as well as the difference in task energy consumption using the schedules found by these two methods. Our heuristic method is very efficient and can find a schedule in which the task energy consumption is very close to that of optimal schedule.

	Runtime (s)			
Workload distribution	Brute-force	Heuristic (DP)	Runtime speedup	Energy diff. (max)
inverse exp	870.1	169.3	5.1	0.000%
exp	505.0	90.3	5.6	0.025%
uniform	1493.9	265.9	5.6	0.151%
norm	663.9	128.0	5.2	0.274%
lognorm	651.4	120.0	5.4	1.390%

Table 3.2: Comparison between brute-force method and our dynamic programming based heuristic method when $r = 10$.

3.6 Summary

In this chapter, we introduced the optimal procrastinating DVS for systems with discrete frequency sets. Our techniques efficiently solve the practical issues encountered by previous techniques. The complexity of our method is O(1) with respect to the number of bins in the task workload histogram, and linear to the number of operating points involved in the schedule. We also find that not all available operating points are necessarily efficient even though they consume less power. We extend our method to deal with transition overheads and find that, for a given deadline, a small number of carefully selected frequencies may be sufficient to form a good schedule achieving energy savings comparable to that using all efficient operating points. We develop an efficient heuristic algorithm to find good schedules composed of only a subset of available frequencies, thereby avoiding unnecessary transition overheads. Thus, our techniques are readily applicable to various practical systems.

Chapter 4

Power-aware Runtime Management using Control-Theoretic DVS [1]

4.1 Introduction

Low power techniques in real-time (RT) embedded systems become more important as the performance expectations of these embedded devices continue to rise. RT embedded systems such as on-board satellite controls contain hard and soft RT deadlines and a strict limit on available battery power. A complicating factor for many applications is that tasks or service requests are often unpredictable, and current solutions for power aware computing cannot provide the guarantees necessary for RT. Over-provisioning is an appropriate solution for some systems but usually not for battery-operated systems where the battery has substantial cost or interferes with the desired form factor.

Closed-loop feedback control is an attractive way to attain soft-RT guarantees. Feedback control defines target metrics and error terms for the system being controlled, monitors the error, and continuously adapts the system to minimize the error. Feedback control can adapt to a wide range of behaviors and hence responds well to unanticipated work-

[1]This chapter is based on the published paper titled "Control-Theoretic Dynamic Frequency and Voltage Scaling for Multimedia Workloads" [59].

loads or behaviors. We use a simulated multimedia system to demonstrate the benefits of feedback-control. Our technique is motivated by the approach of Stankovic *et al.* [99], where a feedback control RT CPU-scheduling scheme is proposed; the design and performance evaluation is shown in [55]. In our feedback-based system, the CPU operating frequency and voltage are scaled based on the current frame throughput.

Following [91], this chapter models an MPEG portable multimedia-playback system with the content received over a wireless network. These multimedia systems require soft RT guarantees on frame delay to maintain the throughput necessary for avoiding choppy playback. We develop a formal feedback-control system to hold the CPU at the minimum possible operating frequency while still continuously adapting to provide the desired throughput. We evaluate the energy savings of our system using an analytic model for a variety of synthetic workloads representing exponential, uniform, and normal distributions of frame-arrival and decoding rates, and also real workloads on a Compaq iPAQ using StrongArm SA-1100 processor.

The rest of the chapter is organized as follows. Section 4.2 discusses our control-theoretic system, and Section 4.3 shows our results on both actual and synthetic workloads. Finally, we summarize our work in Section 4.4.

4.2 Control System Design

In order to simplify our control system design, we model the multimedia decoding system as an M/M/1 queue, as suggested in [91]. Thus, the following equation holds:

$$T = \frac{1}{\mu - \lambda} \tag{4.1}$$

where T is the average delay for the frames, μ is the frame decoding rate and λ is the frame arrival rate.

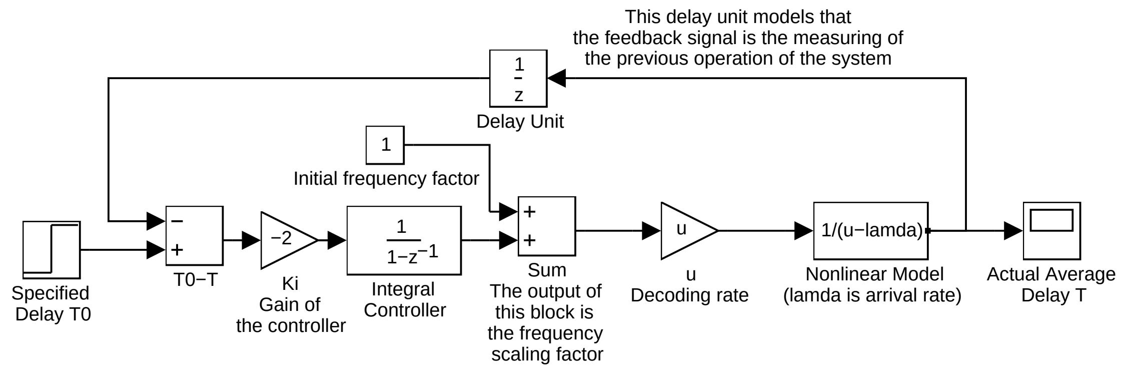

This delay unit models that the feedback signal is the measuring of the previous operation of the system
1/z
Delay Unit
1
Initial frequency factor
−
+
T0−T
−2
Ki
Gain of the controller
1/(1−z^−1)
Integral Controller
+
+
Sum
The output of this block is the frequency scaling factor
u
u
Decoding rate
1/(u−lamda)
Nonlinear Model (lamda is arrival rate)
Actual Average Delay T
Specified Delay T0

Figure 4.1: Basic architecture of our control system. All of the functional blocks are shown with the Z transforms of their respective transfer functions. [59]

Our goal is to adjust the CPU frequency and voltage such that the user-specified delay (referred to as the controller set-point) will be satisfied with the minimum amount of energy and the controller will stabilize quickly to provide a fast response time to variations in delay.

The basic architecture of our control system is illustrated in Figure 4.1. A generic control system architecture usually consists of an input set-point, a controller, a system or "physical plant" that is being controlled, and a feedback path. Our system includes the following components:

Input set-point: The desired average frame delay specified by the user.

Controller: An integral controller and its gain K_i. The output of the *integral* controller is the current (new) decision for the frequency scaling factor, which will go to the decoder system. At the beginning of the operation, we always let the frequency scaling factor equal unity, i.e, we run the decoder at the maximum speed. We force this initial value by adding a constant "1" to the output of the controller. We use a simple integral controller in our system because it can guarantee that the final output of the system will be equal to the set point when the system is stable. The value of K_i will be discussed later.

Controlled system: The multimedia frame decoder which accepts as its input the frequency scaling factor from the controller. A non-linear system model specified by Equation 4.1 is used to model the decoder and it will be linearized for controller analysis.

The actual frame arrival rate and decoding rate in the run time are unknown to the system and can change vary rapidly. We only assume that these values are less than 100 frames per second.

Feedback path: Because what we actually measure from the system is the average frame delay under the previous frequency scaling decisions, we model this feedback signal with a delay unit.

The controlled system is non-linear, but it can be linearized using first order Taylor expansion about the desired decoding rate μ by the following linear equation:

$$T(k+1) = T(k) - T_0^2 * (\mu * r(k+1) - \mu * r(k)) \tag{4.2}$$

where T_0 is the user specified delay and $r(k+1)$ is the frequency factor (a value between 0 and 1) adjusted by our controller at time $k+1$. For example, the decoder will run at its full speed when r is 1. This linear equation assumes that the controlled system works at operating points very close to the set-point. As long as the stability of our control system is guaranteed, this assumption will hold. Note that this assumption is only needed for the control-theoretic analysis.

Using the system model shown in Figure 4.1 and substituting our linear model with the non-linear one, we can find the stability criteria for the whole system based on control theory: $-1 < 1 - |K_i|T_0^2\mu < 1$ or $|K_i| < \frac{2}{T_0^2\mu}$ where K_i is the gain of the integral controller. While a large value of K_i would make the system more responsive, the system would tend to be unstable. Given that in our system T_0 is about 0.1 second and the frame decoding rate is less than 100 frames/sec, and considering the sign of the gain value, we let $K_i = -2$. To show that the system avoids unstable oscillation, we use Matlab to simulate the system shown in Figure 4.1 with a step input signal. The step response of the system is shown in Figure 4.2.

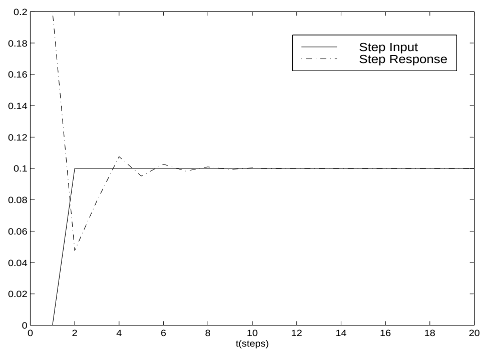


Figure 4.2: Step response of the system shown in Figure 4.1. Although we use the example frame arrival and decoding rates shown in Figure 4.1, other rates yield similar results. [59]

As we can see from Figure 4.2, the system output converges to the set-point very quickly (after about 5 samples(iterations)) We then make use of this fast-converging property in our system to satisfy the RT constraints.

Besides the basic control architecture shown in Figure 4.1, we observe that the number of frames in the queue waiting to be decoded is a good indicator of future traffic. Thus, we feed the number of frames in the queue to a separate differential controller; the output of this controller will also be used to adjust the frequency scaling factor assigned to the decoder.

The feedback signal to our controller is the average delay over a certain number of previous frames (specified by our *window size*). Since Equation 4.1 holds true only for average values over a long time interval, larger window sizes can provide a higher-quality feedback signal (i.e., provide a better estimate of the actual average frame delay in the system). However, a larger window may increase the controller's reaction time to a changing situation (i.e., a different arrival or decoding rate). By simulation, we empirically determined that an average window size of 70 frames provides a good design trade-off. Our *sample* period is five frames (i.e., the controller will determine a new frequency factor every 5 frames), and frames in the same period will be decoded with the same frequency.

In order to make the best determination of the working frequency for the next sample period, we not only use the actual delay statistics from the previous frames in the window size, but we also use them as training samples in our control loop. At each sample, the controller will produce a new frequency factor based on the difference between the set-point and the actual average delay from the previous frames. We then back-calculate for each frame in the entire window how long that frame would have taken to decode had it been processed using the freshly-calculated frequency/voltage setting. The new average delay (a virtual value) is then fed to the controller again in an iterative relaxation process. Due to the fact that our controller converges quickly, it can arrive at a very good decision after only a few iterations in our implementation. There are two reasons for this approach. First, we believe that the frames in the next sample period will have very similar statistics to the previous frames in the window. Second, the controller can apply better decisions directly on the future frames, thus providing very rapid response to changes in the environment. This "virtual training" scheme makes the controller fast; otherwise, it would take many sampling periods for the moving average to reflect the new CPU frequency.

In order to keep the overall average frame delay as close to the user-specified set-point as possible, we use a counter to accumulate the difference between individual frame delay times and the set-point. This provides a global picture of how well the controller is performing and allows us to make the set-point adaptive. At each sampling point, if the controller finds that the previous average delay is much higher than the set-point, it lowers the actual set-point used for the next period to obtain better performance. On the other hand, if the previous average delay is in fact lower than the set-point, the controller increases the actual set-point to save more energy. This adaptive set-point scheme provides the best way to meet the user-defined requirements. This mechanism is equivalent to adding another control loop outside the whole system shown in Figure 4.1.

Though it seems that our scheme may consume a large amount of computation time and energy, the computation required for our controller is less than that of the change-point detection approach proposed in [91]. We compare the amount of computation needed for both algorithms in the average case. In the change-point detection scheme, the algorithm is executed for every new frame to be decoded. This algorithm assumes that there is a set of predefined arrival/decoding rates. For example, there may be 10 possible arrival/decoding rates. For each frame, the algorithm checks the previous 100 frames to identify the possible *change point* (i.e., the frame among the last 100 from which the arrival/decoding rates appears to have changed) by comparing the statistical maximum likelihood with a certain threshold. The amount of computation that this algorithm requires for each frame is about (including both arrival and decoding rate detection):

$2*10*10*\frac{1}{2}*$(2 adds + 3 multiplications + 1 logarithms) = 200 adds + 300 multiplications + 100 logarithms

In our control-theoretic scheme, the bulk of the computation is for the virtual training. Since our algorithm is invoked every 5 frames and (for our initial implementation) we use 5 iterations for virtual training, we have an average of one iteration per frame. The computation needed to calculate the average delay for the previous 70 frames when a new frequency is virtually applied is:

70*(2 multiplications + 5 adds) = 350 adds + 140 multiplications

Our later simulations show that the number of iterations can be even further reduced while maintaining performance.

Once the new frequency scaling factor is obtained, we scale the system voltage to the minimum that is required to maintain the new working frequency. The relation function between the working frequency and the respective minimum system voltage can be measured in advance for any system under study.

4.3 Simulation Experiments and Results

We evaluate the performance of our control-theoretic approach on both actual and synthetic workloads.

4.3.1 Evaluation Using Actual Workloads

We implement our control-theoretic system on a Compaq iPAQ running an SA-1110 StrongARM processor. The iPAQ runs the Windows CE operating system. Running the control based model requires the scaling of core processor frequency, which is discussed in [19]. With the direct hardware access techniques discussed in [19], frequency scaling is accomplished by writing the desired frequency index to the Phase Locked Loop Configuration Register (PPCR). This allows for on the fly frequency scaling during program execution.

An open-source MPEG codec is provided by the MPEG Software Simulation Group (MSSS). Four MPEG-1 video files were used as input to the MPEG decoder to produce frame decoding rates for an actual workload. The four MPEG-1 videos (three 900 frame video files and one 1800 frame video file) yield frame rates of less than 1 frame/sec. This is much lower than typical MPEG frame rates of between 15 and 20 frames/sec. However, we expect lower frame rates from a device running Windows CE on a slow processor with a maximum speed of 221.2 MHz as well as a slower bus speed. Slow decoding rates are not expected from commercial MPEG players such as Windows Media Player, however commercialized MPEG players are highly optimized and source code is proprietary information not available to the public. Such commercial MPEG players provide a much higher and predictable frame rate. Possible future work is to obtain source code for a commercial multimedia player to obtain additional MPEG decoding rate trace data.

Using actual MPEG-1 trace files containing actual decoding rates, we tested our control-theoretic algorithm on the iPAQ device. Evaluating the efficiency of our control algorithm showed a total overhead of approximately 0.371 milliseconds per frame. Since this is less than 1 percent of a typical decoding time, which is normally around 40 milliseconds, we find that our control-theoretic approach is very efficient when running on an actual system.

Implementing our control-theoretic algorithm on such a system shows that our model is robust and capable of running on a handheld device with little overhead. Dynamically scaling frequency on a handheld device allows the portable system to have extended battery life while maintaining performance. A comprehensive evaluation of the real-world battery savings and quality of service characteristics on the iPAQ as a result of these approaches is the subject of future work; however, preliminary results indicate a strong correlation to the simulation results shown in the next section.

4.3.2 Evaluation Using Synthetic Workloads

For experiments on synthetic workloads, we used Matlab to generate traces matching the desired distributions (exponential, Gaussian, etc.) We have 1000 frames in each simulation run, and the statistics of the frames (i.e. arrival rate and decoding rate) may or may not change after every 250 frames.

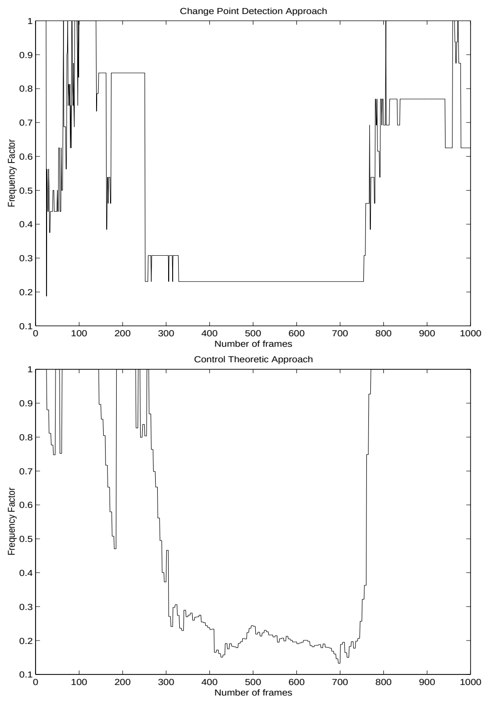


Figure 4.3: Frequency scaling curves for both algorithms in one simulation run. [59]

Figure 4.3 shows the frequency adjustment for one simulation run. In this simulation, all frames have the same decoding rate–70 frames/second[2]. The arrival rate is reduced from 45 frames/second to 5 frames/second after the 250th frame. From the 750th frame, the arrival rate changes sharply to 50 frames/second. The frequency scaling curves in Figure 4.3 show that both the change-point detection algorithm and our control-theoretic approach can adapt to the workload changes very quickly.

Mattavelli *et al.* [67] have shown that multimedia frame-decoding rates cannot be modeled by any simple distribution. In order to further evaluate the performance of the two approaches, we use different random distributions to create our synthetic workloads and divide them into 6 groups:

1. Both inter-arrival times and decoding times are exponentially distributed with high

[2]High decoding rates and arrival rates are used in this experiment to show the adaptive nature of the two approaches.

CPU utilization.

2. Both inter-arrival times and decoding times are exponentially distributed with low CPU utilization.

3. Both inter-arrival times and decoding times are exponentially distributed with hybrid (mixture of high and low) CPU utilization.

4. Inter-arrival times are exponentially distributed, and decoding times conform to Gaussian distribution, with hybrid CPU utilization.

5. Inter-arrival times are exponentially distributed, and decoding times are uniformly distributed, with hybrid CPU utilization.

6. Inter-arrival times are uniformly distributed, and decoding times conform to Gaussian distribution, with hybrid CPU utilization.

High CPU utilization means the CPU will mostly operate at a speed close to its fastest speed. On the other hand, low CPU utilization means the CPU need only operate at a slow speed in order to satisfy the delay requirement. For example, in Figure 4.3, the last 250 frames impose a high utilization on the CPU of the decoder, while the frames between 250th and 750th impose only a light workload for the CPU.

Each workload group is composed of 100 different simulation runs of 1000 frames. We measure, for each simulation run, the average delay, total energy, and frame-delay variance. We do not make measurements of the energy on the actual system due to the difficulties and inaccuracies of doing so on the iPAQ. Instead, our work adopts the approximate energy calculation proposed in [92] (i.e. Equation (2.2) in Chapter 2) and use it throughout our simulations.

At the same time, for each workload group, we calculate how many simulation runs will lead to an average frame delay within $\pm 10\%$ of the user specified delay (delay set-

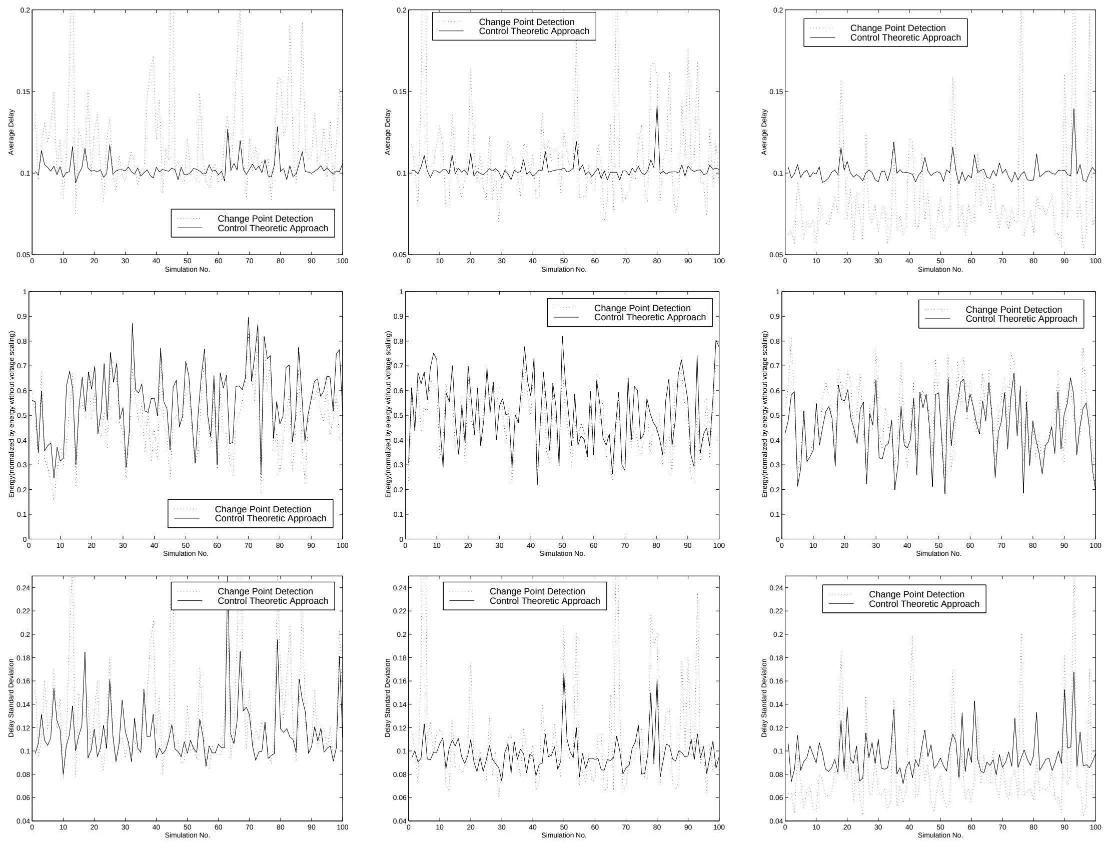


Figure 4.4: Performance comparison of the two approaches in terms of average delay, energy consumption, and delay variance for all simulation runs in three workload groups (left column: M/M/1 with hybrid CPU utilization; middle column: Exponentially distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization; right column: Uniformly distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization). The control-theoretic approach provides better timing guarantees with comparable energy savings. Similar results are obtained for other workloads including uniformly distributed decoding times. [59]

point). Results for each simulation run from 3 workload groups are presented in Figure 4.4, and similar results are obtained from the other 3 workload groups. The statistics (average energy consumption and average delay standard deviation) for all 6 workload groups are summarized in Tables 4.1 and 4.2.

It is clearly shown in all six cases that our control-theoretic approach outperforms maximum likelihood detection in quality of service while providing comparable energy

Workload groups	Ratio of sample runs near set-point (+/-10%)	Average energy over all runs in the same group (ratio)	Average standard deviation over all runs in the same group (sec/frame)
Group 1	0.40	0.68	0.12
Group 2	0.71	0.23	0.11
Group 3	0.41	0.48	0.13
Group 4	0.38	0.47	0.11
Group 5	0.35	0.43	0.08
Group 6	0.04	0.50	0.08

Table 4.1: Performance summary for the change-point detection approach [91]. The energy column gives the energy consumed relative to no frequency/voltage scaling. [59]

Workload groups	Ratio of sample runs near set-point (+/-10%)	Average energy over all runs in the same group (ratio)	Average standard deviation over all runs in the same group (sec/frame)
Group 1	0.90	0.73	0.11
Group 2	1.00	0.28	0.11
Group 3	0.92	0.56	0.12
Group 4	0.94	0.52	0.10
Group 5	0.97	0.42	0.08
Group 6	0.94	0.45	0.10

Table 4.2: Performance summary for our control-theoretic approach. [59]

savings. With a minimal relative variance, our approach is able to provide smooth transitions to yield very consistent frame delays. Consequently, the average frame delay for the control-theoretic approach oscillates around the set-point of 0.1 seconds with minimal deviation.

Tables 4.1 and 4.2 show that our control-theoretic approach can guarantee the user specified set-point with very high probability. Energy consumption increases only slightly compared to the change-point detection algorithm. In contrast, the change-point algorithm tracks the user-specified frame delay rather poorly. Furthermore, notice that although our control system is designed with an M/M/1 queuing model as the system model, and that assumption is only required for control-theoretic analysis, so it still performs well in the cases where the decoding time is Gaussian or uniformly distributed. This demonstrates the robustness of the formal feedback-control approach.

Iteration number	Average Frame Delay	Frame Delay Standard Deviation
1	0.099	0.100
2	0.102	0.099
3	0.102	0.093
5	0.102	0.092
8	0.102	0.095
10	0.103	0.097

Table 4.3: Impact of number of iterations of “virtual training” for a trace of M/M/1 Model with average delay set point = 0.1 sec. [59]

Iteration number	Average Frame Delay	Frame Delay Standard Deviation
1	0.105	0.102
2	0.107	0.110
3	0.11	0.110
5	0.107	0.106
8	0.098	0.094
10	0.103	0.106

Table 4.4: Impact of number of iterations of “virtual training” for a trace of exponentially distributed inter-arrival times and uniformly distributed decoding times with average delay set point = 0.1 sec. [59]

Tables 4.3 and 4.4 illustrate examples of varying numbers of iteration cycles for the ”virtual training” scheme. In the M/M/1 model and in the exponential/uniform model, the average frame delay remains steady with little variation. The number of iterations ranges from 1 to 10. Therefore, the number of iterations in the ”virtual training” back-calculation does not affect frame delay, and multiple iterations are used to achieve a responsive controller.

Frequency scaling is done in a continuous fashion, though we have briefly explored the method of quantized dynamic frequency scaling described in [89]. Frequency settings are linearly subdivided into 32 discrete levels in order to decrease the number of frequency transitions needed. We found that quantizing frequency scaling factors into discrete levels has a negligible effect on performance since frequency transition time is small compared to frame decoding time.

4.4 Summary

In this chapter, we describe a control-theoretic dynamic frequency/voltage scaling scheme for a simulated multimedia portable device. Our control design uses virtual training and an adaptive set-point to provide fast response and stable multimedia throughput, regardless of workload behavior/distribution. This system provides comparable energy savings to the change-point detection technique [91] but dramatically improves stability in average frame delay. It is also not dependent on the assumption of M/M/1 queuing behavior. Overall, we have shown that formal feedback control is a robust and responsive way to control DVS settings for real-time response. The next step to extend this work is to fully integrate our controller into a QoS framework, such as the case presented in the next chapter.

Chapter 5

Power-aware Runtime Management for a Video Playback System [1]

5.1 Introduction

While the continuously increasing computation power provided by technology scaling enables more complex mobile applications, energy consumption becomes a limiting factor. Since multimedia playback is among the dominant applications in mobile devices, it is important to design energy efficient multimedia playback systems. Fortunately, due to the variations in multimedia decoding complexity, the required computation power changes during the playback, and dynamic voltage/frequency scaling (DVS) is a powerful technique to exploit this opportunity to reduce runtime power by lowering down the circuit speed (and thereby power) whenever possible. However, DVS multimedia systems should be carefully designed to avoid causing large degradation in playback quality. In this chapter, we systematically analyze the design of such a system, propose a new design technique and validate our design through a prototype system on a DVS enabled laptop computer.

[1]This chapter is an excessive extension from the published paper titled "Design and Implementation of an Energy Efficient Multimedia Playback System" [65].

Designing a good multimedia playback system using DVS involves trade-offs among multiple contradicting objects and is subject to practical constraints. Techniques proposed in [23, 24, 102] require perfect predictions for decode execution timing, which is not possible in some multimedia compression formats. The methods in [47, 62] exploit buffering to improve the energy efficiency of DVS. However, smaller buffer sizes might increase the frame deadline miss rate, while larger buffer sizes not only increase hardware costs but also increase the playback latency, which is unacceptable in many real-time applications. In general, a practical solution has to seek the best trade-off between power consumption, playback quality and hardware resources and, in general, has to assume no specific information about the incoming multimedia streams.

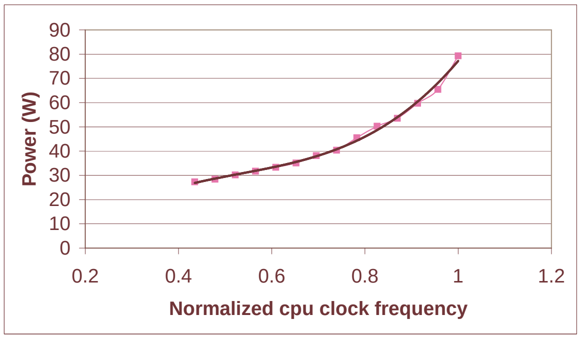


Figure 5.1: MPEG decoding power at different CPU speed. [65]

The DVS design presented in this chapter provides a good balance between all these competing factors. We first create a model for the available design space. Then we search the design space with two desired properties in mind: speed schedule uniformity and system stability. Power consumption is a convex function of circuit speed. As an example, in Figure 5.1, the MPEG decode power consumption of a laptop computer is shown at different CPU speeds (i.e., frequency and voltage settings). Though each frame can be decoded at a different speed due to computation variations, decoding multiple frames at a uniform speed (i.e., the average decoding speed) provides better energy efficiency according to Jensen's inequality [85]. Therefore, in an energy efficient DVS design, a uniform speed schedule is preferable. On the other hand, due to the complexity in many modern multimedia coding formats, a good DVS solution should not assume any prior knowledge or rely on any accurate predictions about the decoding process. Therefore, the decode speed decisions should be solely dependent on the results of the previous frames. Any system working in this way is in fact a closed loop feedback system for which stability is an important property.

The rest of the chapter is organized as follows. Section 5.2 discusses related work and Section 5.3 presents the architecture of the multimedia playback system studied in this work. In Section 5.4, we report some video playback timing characteristics from actual measurements of video clips with various compression formats. Based on these observed timing characteristics, in Section 5.5, we propose a generic architecture to model the dynamics of a variable speed video decoding system. Then we discuss the design of the speed controller based on speed uniformity and system stability, and we propose an enhanced feedback based speed control scheme. In Section 5.6, we describe our implementation of various DVS techniques for video playback on a DVS enabled platform. Finally, Section 5.7 presents the performance comparison of different DVS techniques on a set of clips with various video formats and shows the effectiveness of the new technique proposed in this chapter.

5.2 Related work

Low power design in multimedia systems is currently a very active research field. The major power saving techniques applied to multimedia systems include DVS and DPM (dynamic power management). In DVS, different computational tasks are run at different voltages and clock frequencies while still providing an adequate level of performance.

DPM shuts off system parts (inside or outside the CPU) that are not in use at any given time. Structural adaptations for multimedia systems were recently proposed in [44].

Because a multimedia system has defined QoS requirements, it is usually modeled as a real-time system with soft deadline constraints. Inter- and intra-frame scaling are considered to reduce the deadline miss rate in [44] and [23], respectively. Frame-decode-time prediction techniques are used in these works to achieve a satisfactory trade-off between the power savings and QoS. Chung *et al.* [26] proposed that the decoding time information can be obtained from the multimedia content provider, such that the accuracy of the decode time prediction can be improved, while Yuan *et al.* [115] used the probability distribution of frame decoding time obtained during runtime to help determine decoding speed for each frame. In all these works, the decoder is assumed to decode only one frame for each display interval.

Other work [47, 57, 62, 102] showed that buffering can be exploited to further reduce the decode energy during multimedia playback. Lu *et al.* [57] used an off-line algorithm to schedule the frame decoding rate and respective frequency, and they did not consider multimedia streams that include *B* frames. Im *et al.* [47] focused on the estimation of the input/output buffer size for the decoder, and also proposed an on-line DVS technique. This technique is essentially equivalent to a scheme we call *panic factor scaling* in this chapter. Tan *et al.* [102] proposed a technique to predict the frame decoding time in advance and found the optimal decoding speed according to the predictions. In our previous work [62], prediction is not used, and a PID controller is used instead to adjust decoder speed according to buffer occupancy. A draining buffer indicates that the decoding rate is too slow, and a growing buffer indicates that more energy can be saved by scaling down the decoding rate. The idea behind this chapter is originated from this previous work, and extends it from several aspects: 1. A new feedback control scheme is developed basing on a novel modeling framework for multimedia playback system; 2. The new scheme is implemented and tested in a real hardware platform, while our previous work is based on simulations; 3. Actual measurement results indicate that the new scheme is more advantageous over our previous work in terms of both energy savings and hardware overheads.

5.3 Overall architecture of a multimedia playback system

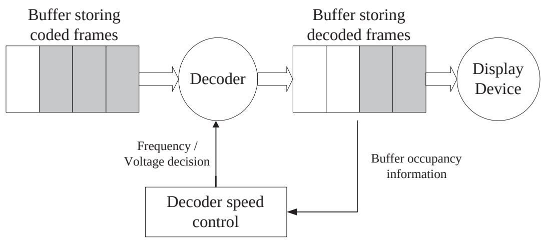

Buffer storing coded frames
Decoder
Buffer storing decoded frames
Display Device
Frequency / Voltage decision
Buffer occupancy information
Decoder speed control

Figure 5.2: Overall architecture of a multimedia playback system. [62]

The general architecture of a multimedia system is illustrated in Figure 5.2. Frames in the decode buffer (or in disks) are fetched and decoded sequentially and displayed by the display device with a constant playback rate denoted by $\frac{1}{T}$, where T is the display interval. Following assumptions are made for this multimedia system:

- Coded frames are decoded in their normal order, and a frame is always available to be decoded whenever the frames before it have already been decoded.
- After a frame is decoded and sent to the display buffer, the decoder is ready for decoding the next frame.
- Each frame has an associated deadline, which is the time to start displaying the frame. The deadline is always a multiple of the display interval T. For example, frame *i* has a deadline of $i * T$. The time needed to decode frame *i* is denoted by c_i.
- The display buffer has enough space to hold all the frames which are decoded before their deadline.

With these assumptions, we state the following theorem [62].

Theorem 5.1. *Let S be a multimedia stream including frames* $\{F_1, F_2, \ldots, F_n\}$. *If the decoder in the system described above never idles when there are undecoded frames left, S will be schedulable (all the frames meet their respective deadlines) if and only if:*

$$\frac{\sum_{k=1}^{i} c_k}{i} \leq T \qquad \forall i,\, 1 \leq i \leq n \tag{5.1}$$

where c_k *is the time needed to decode frame k, and T is the display interval.*

Proof.

1. "Only If" condition:

   Let us consider a time interval $[0, iT]$, where $1 \leq i \leq n$. Since all frames are decoded before their deadlines, at least the first i frames are decoded during this interval. Thus:

   $$\sum_{k=1}^{i} c_k \leq iT$$

2. "If" condition (proof by contradiction):

   Let i be the first frame which misses its deadline. Therefore

   $$\sum_{k=1}^{i} c_k > iT$$

   which contradicts with Equation (5.1).

■

Theorem 5.1 shows that as long as the average decoding time of the frames (the first frame, the first two frames, the first three frames, etc.) is always less than or equal to the display interval, all the deadlines will be satisfied. This suggests that we can slow down the decoder such that some frames may even have decoding times larger than the display interval, provided that they can make use of the slack time from the frames with shorter decoding times. This gives great flexibility for DVS. In contrast, if a multimedia system has no buffer between the decoder and the display device or the buffer can only hold one frame at a time, the schedulability criteria must be reduced to:

$$c_i \leq T \qquad \forall i, 1 \leq i \leq n \tag{5.2}$$

which is much more restrictive for DVS.

Let k_j denote the decoding time for frame j at full speed. Assuming that the actual decoding time is inversely proportional to the decode speed u (this assumption is not exactly true in practice, and will be adjusted later in the chapter), it follows that $c_i = \frac{k_i}{u}$. Since uniform decoding speed tends to bring maximum energy saving, it is desirable to decode continuous n frames using one single speed, denoted by $u_{uniform}$. According to Theorem 5.1, it can be derived that:

$$\begin{aligned} cu_{uniform} \geq \frac{\sum_{j=1}^{i} k_j}{iT}, \qquad \forall i, 1 \leq i \leq n \\ \text{or} \\ u_{uniform} = \max_{i \in \{1,2,\ldots,n\}} \left(\frac{\sum_{j=1}^{i} k_j}{iT}\right) \end{aligned} \tag{5.3}$$

In our previous work [62], we called the DVS scheme using this uniform speed “*optimum*”, because this scheme sets an achievable lower bound energy consumption provided that all frame deadlines are met. In order to implement this DVS scheme, we also assumed that the actual frame decode time information (i.e. k_j in Equation (5.3)) can be obtained in advance, for example, through profiling. Tan *et al.* [102] derived a similar formula in their work, and they obtained the frame decoding time information by their prediction technique.

5.4 Timing characteristics of MPEG workloads

In some previous work on power-aware MPEG decoding [47, 62, 102], a simple performance model (i.e. frame decoding time is inversely proportional to decoding speed) is assumed. Our measurements from actual decoding implementation indicate that this assumed model is over-simplified. Choi *et al.* [24] reported similar findings. In this section, we present the timing characteristics measured from actual MPEG workloads on a general purpose processor.

Table 5.1 lists the information of a set of video clips of different compression formats used in this work, among which, six are movie trailers downloaded from Internet [3–5], the other two, (*fs2003* and *fs2004*), are home-made movies. For some clips, the normal playback rate (around 30 frame per second) requires little CPU effort. In order to effectively compare different DVS schemes as discussed in Section 5.7, we use a higher than normal frame rate for those clips (the 3rd column in Table 5.1). The new frame rate is determined in such a way that the number of frames missing deadlines when decoded using full CPU speed is within 100 (out of total 3000 frames) at that frame rate.

As shown in Figure 5.2, there are two major tasks in an MPEG playback application: frame decoding and frame display. Though different video compression formats have quite different implementation details, there are several common steps in frame decoding: a. Huffman decoding, b. IDCT and motion estimation/compensation, and c. frame dithering (i.e., converting the decoded frame from the YUV plane to the RGB plane). The decoded frame in RGB format is then sent to a graphics device that finishes the display task, which is essentially the same for all movie formats. There are several frame types depending on the operations needed to encode/decode a frame: I (intra-coded) frame, P (predictive coded) frame, B (bidirectionally predictive-coded) frame and S (sprite) frame (coded using global motion estimation) in MPEG4. Table 5.1 shows the composition of different frame types in each movie. This set of movies also present hybrid examples in frame size (the 10th column in Table 5.1).

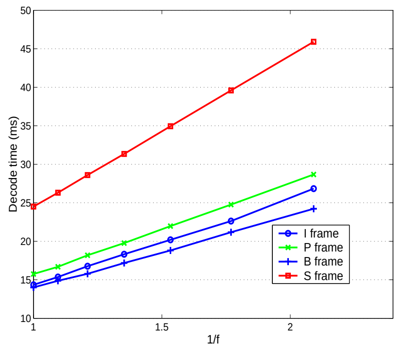
clip name	video format	frame rate	frame type	fraction	Mean (k) (ms)	std (k)	Mean (m) (ms)	std (m)	frame size	average display time (ms)	std. of display time
mission	SVQ3	43.5	I	0.048	14.39	13.62%	1.61	3.34%	640x344	7.35	3.52%
			P	0.452	9.73	24.61%	2.25	6.55%			
			B	0.500	6.72	27.20%	2.74	8.00%			
game	H.264	29.4	I	0.018	16.48	49.10%	1.73	3.88%	640x360	7.69	3.62%
			P	0.545	16.39	22.82%	2.87	14.26%			
			B	0.437	13.29	16.68%	3.96	9.98%			
skeleton	MPEG4	43.5	I	0.096	8.70	6.86%	2.52	2.93%	720x420	9.46	3.68%
			P	0.499	6.85	10.89%	3.05	6.65%			
			B	0.404	6.37	6.37%	3.75	8.56%			
exorcism	MPEG4	43.5	I	0.030	7.69	8.61%	2.47	3.71%	720x384	9.12	2.12%
			P	0.556	6.87	14.52%	2.81	7.52%			
			B	0.413	5.93	7.37%	3.43	9.36%			
fantastic	H.264	29.4	I	0.047	16.55	14.51%	1.92	6.16%	852x356	9.90	2.76%
			P	0.488	15.50	17.51%	3.27	14.44%			
			B	0.464	15.87	14.81%	4.39	12.46%			
kingdom	H.264	28.6	I	0.032	17.98	12.76%	1.97	5.23%	852x360	10.00	2.81%
			P	0.495	15.22	24.32%	3.32	12.46%			
			B	0.473	15.83	18.47%	4.58	10.04%			
fs2003	MPEG4	32.3	I	0.005	11.18	4.21%	2.95	3.04%	720x480	11.29	3.14%
			P	0.175	9.85	5.08%	4.04	3.51%			
			B	0.604	10.65	5.38%	5.18	2.22%			
			S	0.217	17.54	11.59%	4.43	4.58%			
fs2004	MPEG2	38.5	I	0.063	10.31	2.18%	2.40	2.39%	720x480	11.24	3.38%
			P	0.313	9.23	1.45%	3.14	2.86%			
			B	0.624	7.90	1.15%	4.44	2.30%			

Table 5.1: Characteristics of a set of video clips with different compression formats (each clip has 3000 frames).

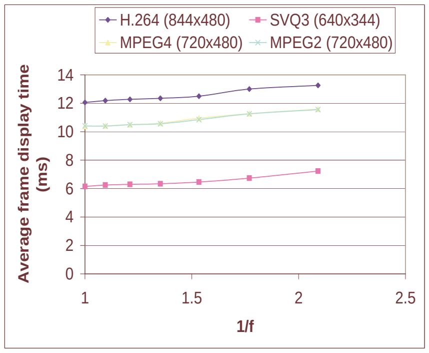
Figure 5.3(a) plots the decoding time (including dithering) of different frame types versus the inverse of decode speed (i.e., CPU clock frequency). This figure indicates that the decoding time of a frame is strongly dependent of the decoding speed and can be modeled as

$$t = \frac{k}{u} + m \tag{5.4}$$

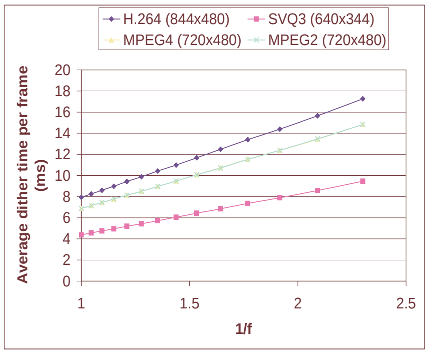
where t is the actual decode time at CPU speed u, and k and m are model parameters representing CPU computation time and memory stall time, respectively, during decoding process. Table 5.1 also lists the mean and variation of k and m for each frame type in each movie (the 6th–9th columns). As shown in the table, the average value of k is usually several times larger than that of m, which implies that MPEG decoding is a computation-dominated application. Within one clip, I frames consume more computation time and less memory time than other frame types, because decoding I frames doesn't need any information from other frames and involves most IDCT operations which are CPU intensive. B frames consume more memory time than other frames since motion estimation/compensation relies on other reference frames. P frames consumes comparable computation time as that of B frames. For all frame types, there exist large variations in both computation time and memory stall time, especially for the former.



(a)



(b)



(c)

Figure 5.3: Timing characteristics of MPEG video playback. [65]

On the other hand, frame display time is almost insensitive to CPU speed as illustrated by Figure 5.3(b) and only a function of frame size, which implies that display task is memory-bounded. Table 5.1 also indicates that there are small variations in display time at full CPU speed (the last column) suggesting that display time is highly predictable. These two observations leave a hint on that energy saving opportunities are available during frame display since one can lower the CPU frequency and voltage when the frame is sent to the screen without affecting playback quality. Choi*et al.* [24] made a similar observation in their study. They also reported that the time spent in frame dithering is also insensitive to CPU speed. We plotted the frame dithering time at different CPU speeds in Figure 5.3(c) and found that the dithering time in the video clips we tested is actually strongly dependent on CPU speed.

These timing characteristics for MPEG workloads need to be considered in seeking the best DVS policy. For example, since the time for frame display task is highly predictable, and a fixed low frequency/voltage can be assigned to this task to save energy, in the rest of discussion, we are only interested in finding the best CPU speed for the decoding process. Therefore we would like to redefine T to denote the actual frame display interval subtracted by the frame display time. Another interesting question raised is whether using uniform speed to decode multiple frames is still the best way to save energy. Our analysis (see Appendix A) shows that, given the MPEG decode timing characteristics listed in Table 5.1, decoding frames using uniform speed still achieves near-to-optimal energy savings (within 5%).

5.5 Design and analysis of feedback based speed control

5.5.1 A dynamic model of DVS multimedia decode systems

Most existing online DVS decoding schemes [23, 24, 47, 62] determine the required decoding speed for future frames according to the decode timing information of previous frames.

Basing on the system architecture illustrated in Figure 5.2, we propose a dynamic system model, shown in Figure 5.4, to represent a design space containing many of these existing schemes.

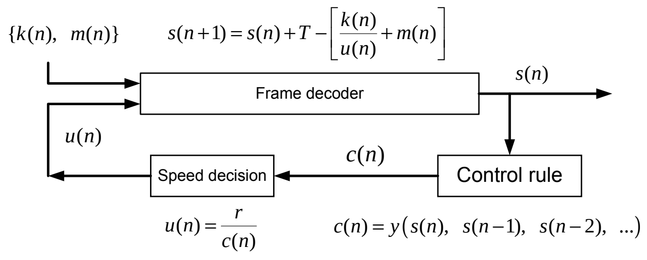

$\{k(n),\ m(n)\}$
$s(n+1) = s(n) + T - \left[\frac{k(n)}{u(n)} + m(n)\right]$
Frame decoder
$s(n)$
$u(n)$
$c(n)$
Speed decision
Control rule
$u(n) = \frac{r}{c(n)}$
$c(n) = y(s(n),\ s(n-1),\ s(n-2),\ ...)$

Figure 5.4: A generic dynamic system model for online DVS MPEG decoding. [65]

In order to explain this model more clearly, we introduce the concept of frame slack time, defined as the time interval between when the frame decoding begins and the frame deadline (the time to display the frame). The slack time for frame n is the maximum available time for its decoding, denoted by $s(n)$. As discussed in Section 5.4, the decoding time of a frame can be expressed as $\frac{k(n)}{u(n)} + m(n)$. Let T represent the frame interval during playback (As discussed at the end of Section 5.4, in the rest of this chapter, T actually represents the frame display interval subtracted by the display time), and we can update the slack time for the next frame by $s(n+1) = s(n) + T - [\frac{k(n)}{u(n)} + m(n)]$, as represented by the “Frame decoder” block in Figure 5.4. By modeling the decoding time for a frame as a random variable, an MPEG stream can be represented by a sequence composed of random variables $k(n)$ and $m(n)$. Therefore, the MPEG decoding process can be modeled as a discrete time system excited by a stochastic input sequence as shown in Figure 5.4. The feedback path in Figure 5.4 represents a generic approach to decode future frames based on previous frames, as used in many existing online DVS MPEG decoding schemes. This generic model makes it possible for us to explore the design space more systematically.

The major difference in various online schemes lies in the parameter r and the control function $y()$. For example, in the frame based schemes [23, 24], $c(n) = \frac{s(n)-m'(n)}{k'(n)}$ and

$r = 1$, where $k'(n)$ and $m'(n)$ are the predictions for $k(n)$ and $m(n)$ of frame n, based on previous decoding results of $s(n), s(n-1), \ldots$. In the scheme proposed by Im and Ha [47], $r = WCET$ and $c(n) = s(n)$ where $WCET$ denotes the worst-case frame decoding time. When a formal PI controller is adopted as the control rule as the one proposed in our previous work [62], the control function $y()$ has the form: $c(n) = \frac{1}{k_p(s(n)-s_0)+k_i\sum_{i=1}^{n} s(i)-s_0}$ and $r = 1$, where k_p and k_i are the proportional and integral gains, respectively, and s_0 is the set-point of the controller input.

5.5.2 Design of an enhanced feedback control scheme with speed uniformity

Since uniform speed provides better energy savings when the decoding throughput is matched with that of display, speed uniformity is an ideal property for a good DVS scheme. Keeping this in mind and using the dynamic model shown in Figure 5.4, it is possible to design a better feedback based speed control scheme for MPEG decoding.

Assuming that the system described in Figure 5.4 is operated under a constant speed, the expected value of random variable $s(n)$ (i.e., the frame slack time), denoted by $S(n)$, satisfies the following relation:

$$\begin{aligned} S(n+1) &= E[s(n+1)] = E\left[s(n) + T - [\tfrac{k(n)}{u(n)} + m(n)]\right] \\ &= E[s(n)] = S(n) \end{aligned}$$

since $E\left[T - [\frac{k(n)}{u(n)} + m(n)]\right] = 0$, which is required by the system steady state condition that the decoding throughput is matched with the display throughput. Therefore, in the desired DVS schedule, both the decoding speed and the average value of frame slack time are constant. Since the frame slack time is observable in the system, we propose to use the average slack time to determine the decoding speed, i.e., a control rule $c(n) = y(S(n))$ and

$r = 1$.

Using this control rule, the feedback system can be described by a difference equation:

$$s(n+1) = s(n) + [T - [y(S(n))k(n) + m(n)]] \tag{5.5}$$

Assuming that $\{k(n), m(n)\}$ are IID (independent, identical distribution) random sequences[2] and $E[k(n)] = K, E[m(n)] = M$, one can apply the expected value function on both sides of Equation (5.5).

$$S(n+1) = S(n) + [T - [y(S(n))K + M]]$$

In the steady state, $S(n+1) = S(n) = S$ and $T = y(S)K + M$, and it follows that the steady decoding speed is

$$U = \frac{1}{y(S)} = \frac{K}{T - M} \tag{5.6}$$

The expected value of $s(n)$ can only be estimated from the decoding results $\{s(n), s(n-1), \dots\}$. We adopt a simple moving average function, $s'(n) = \frac{s(n)+s(n-1+\cdots+s(n-i+1))}{i}$ as the estimation of E[s(n)], in which i is the window width of the moving operation and $s'(n)$ is also a random variable. During runtime, even if the decoding speed is correctly chosen for frame n (i.e., $u(n) = \frac{K}{T-M}$), $u(n+1)$ could be different due to the difference between $s'(n+1)$ and $s'(n)$. The variation of $s'(n)$ can be estimated using the standard deviation of $\Delta s'(n) = s'(n) - s'(n-1)$. Let $\sigma()$ denote the standard deviation

[2]Frames in a clip are decoded in a predefined order specified by GOP (Group of Pictures). Therefore two consecutive frames are not necessarily statistically independent. The IID assumption only serves to reduce the analysis complexity.

function, and it can be shown that:

$$\begin{aligned}\sigma(\Delta s'(n)) &= \tfrac{1}{\sqrt{i}}\sigma(T - [\tfrac{k(n)}{U} + m(n)]) \\ &= \tfrac{1}{\sqrt{i}}\sqrt{\tfrac{\sigma^2(k(n))}{U^2} + \sigma^2(m(n))} \approx \tfrac{\sigma(k(n))}{\sqrt{i}}\tfrac{1}{U}\end{aligned} \tag{5.7}$$

because $\sigma(k(n))$ is usually much larger than $\sigma(m(n))$, as found from the timing measurements of some MPEG clips in Section 5.4. Therefore, the variation in speed decision $\Delta u(n) = u(n) - u(n-1)$ due to the variations in frame decoding time can be estimated according to $u(n) = \frac{1}{y(s'(n))}$:

$$\sigma\left(\Delta u(n)\right) \approx \frac{-y'(s'(n))}{y^2(s'(n)}\sigma\left(\Delta s'(n)\right) \tag{5.8}$$

If we can assume that the variations of the amount of computation in frame decoding is proportional to the average amount of computation, we have $\sigma(k(n)) \propto K$. It follows that $\sigma(\Delta s'(n))$ is independent of $s'(n)$ (i.e., current speed) because of Equation (5.6) and (5.7). We would like to choose a function $y(x)$ such that $\sigma\left(\Delta u(n)\right)$ is also independent of the current decoding speed or $s'(n)$. According to Equation (5.8), this requires that:

$$\frac{y'(x)}{y^2(x)} = C$$

where C is a constant. Thus, the control function $y()$ in Figure 5.4 has the form $c(n) = y(s'(n)) = \frac{1}{as'(n)+b}$, where a and b are two constant parameters to be determined. In other words, we have derived a simple linear control rule to determine the next frame decoding speed as

$$\begin{gathered}c(n) = y(s'(n)) = \tfrac{1}{as'(n)+b} \\ \text{and} \\ u(n) = \tfrac{1}{c(n)} = as'(n) + b.\end{gathered} \tag{5.9}$$

Essentially, this is a P (proportional) controller.

The parameters a and b can be determined according to practical constraints such as available buffer size and decoding speed (clock frequency) range. For example, we can conservatively set the full speed when the frame slack time is less than the display period and set the minimum speed when the display buffer is full. Let B denote the number of available display buffer sizes, we have $Max(s'(n)) = (B+1) * T$. These settings require that:

$$\begin{cases} a * T + b = u_{max}; \\ a * [(B+1)T] + b = u_{min}. \end{cases} \tag{5.10}$$

where u_{max} and u_{min} are the maximum and minimum decoding speeds provided by the system. The values of a and b can be determined accordingly and obviously $a < 0$.

5.5.3 Stability analysis of the proposed feedback control system

In the above analysis, we assume the input sequences $\{k(n), m(n)\}$ are stationary (i.e., K and M are independent of time). During movie playback, we find that some clips show strong phased behaviors. In order to provide both good playback quality and low energy consumption, the feedback system should be able to adapt the speed decision to the new set of K and M. This requires the system to have a fast response time and be stable.

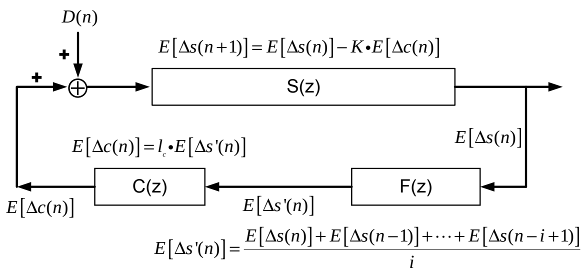

$D(n)$
+
+
$E[\Delta s(n+1)] = E[\Delta s(n)] - K \bullet E[\Delta c(n)]$
S(z)
$E[\Delta s(n)]$
$E[\Delta c(n)] = l_c \bullet E[\Delta s'(n)]$
C(z)
F(z)
$E[\Delta c(n)]$
$E[\Delta s'(n)]$
$E[\Delta s'(n)] = \frac{E[\Delta s(n)] + E[\Delta s(n-1)] + \cdots + E[\Delta s(n-i+1)]}{i}$

Figure 5.5: Modeling the transient behaviors of the proposed closed loop feedback system. A pulse input $D(n)$ models the disturbance caused by the change in the required decode computation power. [65]

The proposed feedback control system can be analyzed using well established linear system analysis techniques (e.g., transfer functions). In order to do so, we first linearize the proposed control function (Equation (5.9)) using the first order Taylor expansion:

$$\begin{gathered} c(n) \approx c(n-1) + l_c\left[s'(n) - s'(n-1)\right] \\ \text{and} \\ l_c = y'(E\left[s'(n-1)\right]) = \frac{-a}{E^2[u(n-1)]}. \end{gathered} \tag{5.11}$$

Second, instead of directly analyzing the signals shown in Figure 5.4, we focus on the signal difference at consecutive time steps, e.g., $\Delta s(n) = s(n) - s(n-1)$, $\Delta c(n) = c(n) - c(n-1)$, etc. In the steady state, the decoding rate matches the display rate, and $E[\Delta s(n)] = 0$. When the required computation power is changed during the playback (i.e., the values of K ($E[k(n)]$) and M ($E[m(n)]$) change), $E[\Delta s(n)]$ will become non-zero and break the steady state. If the system is stable, $E[\Delta s(n)]$ will converge to zero again and the system reaches a new steady state. The block diagram in Figure 5.5 models this dynamic process. The changes in K and M are modeled by injecting a disturbance signal $D(n)$ (usually a pulse signal) into the system, as shown in Figure 5.5.

One important difference between Figure 5.4 and Figure 5.5 is that the sequences in Figure 5.5 are formed by applying the expected value function to those stochastic sequences in Figure 5.4, because only expected values of those random variables are amiable for analysis using transfer functions. Other differences between these two figures include: 1. the blocks "Frame decoder" and "speed decision" in Figure 5.4 are combined into one block denoted by $S(z)$, and 2. one more block is added to model the moving average function introduced in the proposed feedback system.

It can be verified that the transfer functions in Figure 5.5 are: $S(z) = \frac{K}{1-z}$, $C(z) = l_c$, and $F(z) = \frac{1}{i}\frac{z^{i-1}+z^{i-2}+\cdots+1}{z^{i-1}}$. Consequently, the relation between $D(n)$ and $E[\Delta s(n)]$ can be created using their $\mathcal{Z}$ transforms.

$$\begin{aligned}\frac{\mathcal{Z}\{E[\Delta s(n)]\}}{\mathcal{Z}\{D(n)\}} &= \frac{S(z)}{1-S(z)C(z)F(z)} \\ &= \frac{-Kz^{i-1}}{z^i-(1-\frac{Kl_c}{i})z^{i-1}+\frac{Kl_c}{i}(z^{i-2}+z^{i-3}+\cdots+1)}\end{aligned}$$

Therefore the characteristic equation for the closed loop system in Figure 5.5 is

$$z^i-(1-\frac{Kl_c}{i})z^{i-1}+\frac{Kl_c}{i}(z^{i-2}+z^{i-3}+\cdots+1)=0 \tag{5.12}$$

The system is stable if and only if the roots of Equation (5.12) are within the unit circle. The magnitude of the roots determines the converging speed (i.e., response time) of the system.

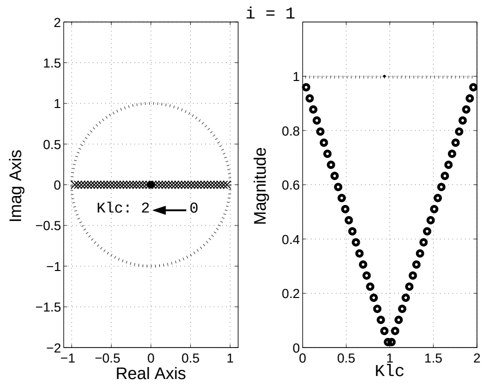
Figure 5.6 shows the root with the maximal magnitude of Equation (5.12) at different values of Kl_c and i. As indicated by the figure, as the moving averaging window size (i) increases, the system tends to be unstable and reacts more slowly. This is expected as the new feedback signals are weighted less with larger window sizes.

From (5.11), we have $Kl_c \le K\frac{-a}{u_{min}^2} \le \frac{-aT}{u_{min}^2} = \frac{u_{max}-u_{min}}{Bu_{min}^2}$, because of $a = -\frac{u_{max}-u_{min}}{BT}$ from (5.10) and assuming $K \le T$. Thus, the stability condition for the system under study can be written as:

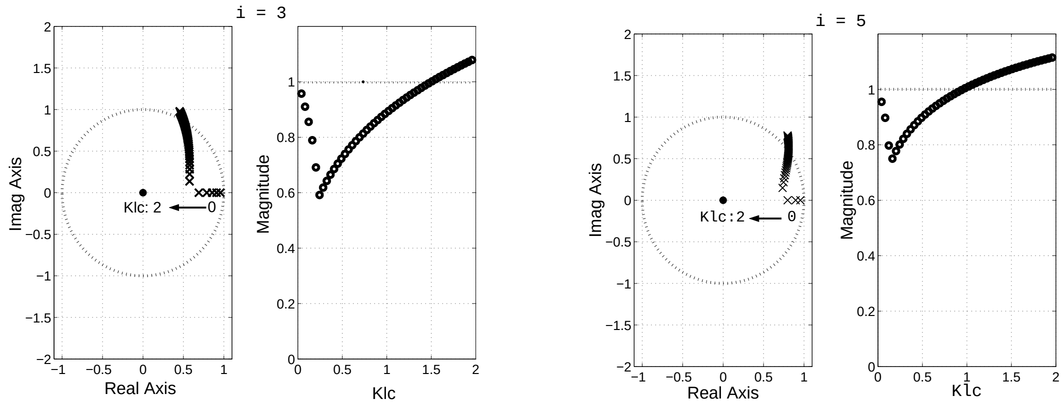
$$\frac{u_{max}-u_{min}}{Bu_{min}^2} < P_i \tag{5.13}$$

where there are roots on the unit circle when $Kl_c = P_i$ in Equation (5.12).

Strictly speaking, condition (5.13) only guarantees the stability of the linearized closed loop system shown in Figure 5.5 and, consequently, the stability of the original system when the system is operating near the steady state. However, condition (5.13) does not necessitate the global stability of the original system due to the non-linear control function (Equation (5.9)). The study of the global stability of the closed loop non-linear system is



(a)



(b) (c)

Figure 5.6: Roots with the maximal magnitude as functions of Kl_c. (a) $i = 1$. (b) $i = 3$. (c) $i = 5$. The plots on the left show the locations of the roots in the unit circle and those on the right show the magnitude of the roots. [65]

outside the scope of this chapter. In practice, we found that condition (5.13) is sufficient to ensure stability in our experimental system.

5.5.4 Real-time guarantee

High video playback quality is accomplished by ensuring that frames are decoded before their deadlines. In a feedback based DVS decoding system (e.g., the system shown in Figure 5.4), real-time guarantees are achieved by the control function $c(n)$. Specifically, for the proposed control function in Equation (5.9), we find that when $i = 1$, no frames will miss their deadlines if the worst-case decoding time at full speed is less than the display interval T and:

$$B \geq \frac{u_{max} - u_{min}}{u_{min}} \tag{5.14}$$

in which B is the display buffer size.

When $i > 1$, condition (5.14) no longer provides hard real-time guarantee. But it still provides helpful reference for choosing the correct design parameter.

5.5.5 Trade-off analysis on design parameters

For a practical DVS system providing variable speed between u_{max} and u_{min} and a target display rate (or display interval T), there are two design parameters for the proposed feedback based decoding system: display buffer size B and moving average window size i. According to Equations (5.8) and (5.9), the decoding speed variation can be re-written as $\sigma(\Delta u(n)) \approx a\sigma(\Delta s'(n))$ where a is the control function parameter in Equation (5.9) and $\sigma(\Delta s'(n))$ is defined in Equation (5.7). Equations (5.10) and (5.7) indicate that larger B and i result in smaller a and $\sigma(\Delta s'(n))$, respectively, and therefore smaller speed variation, which is preferable for energy savings. In addition, larger display buffer sizes help satisfy the stability condition (5.13) and reduce frame deadline misses as implied by Inequality (5.14).

On the other hand, there are disadvantages for larger values of B and i. Larger display buffer sizes not only increase hardware costs, but they also increase the playback latency because $B+1$ frames are decoded ahead of displaying in the worst-case. This is especially undesirable in some real-time multimedia applications. The disadvantages of larger values of i can be seen from Figure 5.6. A larger i tends to increase the magnitude of the roots of the characteristic equation of the feedback system, thereby increasing the system response time. It also reduces the stability range of Kl_c, causing the system to be unstable.

5.6 Implementation on a hardware platform

Our prototype platform is a Compaq notebook computer equipped with a mobile AMD Athlon XP DVS enabled processor. The CPU clock frequency can be dynamically scaled to 14 discrete speeds between 665MHz and 1530MHz by writing the voltage/frequency setting to a model specific register (MSR). The available CPU speed/voltage pairs provided by our testbed computer are listed in Table 5.2. During the experiments, we pull out the notebook battery and put a small sense resistance ($10m\Omega$) in series with the laptop. The voltage drop on the sense resistance is amplified and sampled at a $5K$ sample rate by a desktop computer equipped with a data acquisition card and LabVIEW software. The overall hardware set-up is shown in Figure 5.7.

Speed	1.000	0.957	0.913	0.87	0.826	0.783	0.739
Voltage (V)	1.500	1.400	1.350	1.300	1.275	1.225	1.150
Speed	0.696	0.652	0.609	0.565	0.522	0.478	0.435
Voltage (V)	1.125	1.075	1.050	1.024	1.000	0.950	0.925

Table 5.2: Available CPU clock speed/voltage pairs in the prototype platform computer (Clock speed is normalized to the maximum frequency 1530MHz).

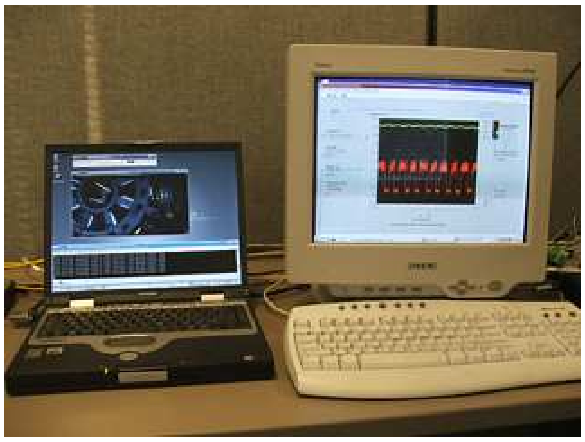


Figure 5.7: The prototype platform (the notebook computer on the left side) and the host running the data sampling software (the desktop computer on the right side).

The video playback software is implemented using two processes and runs on Linux operating system. One process of the software is responsible for fetching coded frames from the hard disk, decoding the frame and putting the decoded frame in the display buffer. The other process is responsible for displaying frames on the screen by fetching a frame picture from the buffer and sending it to the X Window server. The display process is in sleep mode most of time and is periodically woken up by a timer that repeatedly times out at the display interval (i.e., the inverse of the playback rate). Frame deadline misses occur when the display process tries to fetch a frame picture from an empty buffer. When the display process is woken up, the decode process is suspended and resumed after the display process is finished. At the beginning of decoding a new frame, the decode process determines the decoding speed according to the specific DVS scheme and changes the CPU frequency/voltage using a system call modified from the open source *CPUfreq* driver [2]. The measured latency to change frequency/voltage on the fly is less than $100\mu s$. The buffer management routines in the software created by Cen [20] are borrowed to manage the display buffer of our software. The display process is similar as that used in *mpeg_play* [68]. The open source codec library *ffmpeg* [1] is used in the decode process. Thus, video clips with a wide range of different formats can be showed in our software.

5.7 Experimental results

We implemented various DVS MPEG decoding schemes in our hardware platform and tested them on a set of eight video clips with different compression formats as listed in Table 5.1. The various DVS schemes implemented are similar to those examined in [62] plus the new feedback scheme introduced in this chapter:

• *Full speed* The CPU always decodes frames at full speed just as in the systems without DVS capabilities. The CPU becomes idle until the decoded frame is displayed. Therefore the display buffer size in the scheme is 1.

• *Ideal period* Using profiling information for the video streams, the processor calculates the correct decoding speed such that the decoding time for each frame is exactly equal to the display period and rounds up this calculated speed to the available discrete speed provided by the hardware. This scheme represents the best-case of the scheme proposed in [24] where predictions about frame decode timing are always accurate.

• *Optimum* As presented in Section 5.3, if n continuous frames can be decoded using one speed without missing deadlines, the uniform speed is determined by Equation (5.3). When the decoding process is pure CPU-bounded (i.e. decoding time is inversely proportional to CPU clock frequency), decoding frames using this uniform speed results in minimal energy consumption. Given that, in reality, some portion of the decoding time is spent on memory stalls, decoding using uniform speed is not the true optimal solution. However, as analyzed in Appendix A, given the timing characteristics shown in Table 5.1, decoding multiple frames using uniform speed still achieves near-to-optimal energy saving. However, Equation (5.3) needs to be adjusted to account for memory stall times:

$$u_{uniform} = \max_{i \in \{1,2,\ldots,n\}} \left(\frac{\sum_{j=1}^{i} k_j}{iT - \sum_{j=1}^{i} m_j} \right) \tag{5.15}$$

Therefore, using Equation (5.15) and the profiled timing information for all frames (i.e. k_j and m_j), one can still find a speed schedule in which multiple frames are decoded using a uniform speed without missing deadlines. We use this scheme to approximate the optimal schedule and call it *Optimum*.

• *Panic* This scheme is similar to that proposed by Im *et al.* [47], which tries to spend the available slack time in decoding the next frame and assumes the decoding time of the next frame is equal to $WCET$. However, the accurate worst-case decode time is usually unknown in practice. Therefore, in our implementation, we use the longest decode time seen so far during movie playback as the prediction for $WCET$ in future frames. Again the calculated speed is rounded up to the next available discrete speed. The display buffer size for this scheme is 5.

• *Dead-zone based feedback PI controller* This scheme specifies a region (dead-zone) for the number of decoded frames in the buffer. A PI controller is used to pull the system back to this region if the current number of frames in the buffer is out of the dead-zone. Following the design in [62], in our implementation, the dead-zone is between 3 and 8 frames. When the system is within the dead-zone, the CPU is operated at a speed calculated using the decode timing information from the previous several frames [62]. The display buffer size for this scheme is 10.

• *Linear slack speed control* This is the enhanced feedback DVS control scheme proposed in this chapter in which the decoding speed for the next frame is a linear function of the averaged available slack time as indicated by Equation (5.9). In our implementation, the averaging window size i is fixed to be 3, as it is a good trade-off between speed variation and system response time as discussed in Section 5.5. We also tested this technique with different display buffer sizes and found that a five-frame buffer would be a good design trade-off. Using $i = 3$, $B = 5$ and $u_{max} = 1.0, u_{min} = 0.435$ (the frequency range of our hardware platform), one can verify that both the stability condition (5.13) and real-time condition (5.14) are satisfied.

As revealed in Section 5.4, the time needed to display a frame on the screen is insensitive to the CPU speed. Therefore, for the sake of fair energy consumption comparisons, in all of the above schemes, we scale the CPU speed to a fixed low frequency when the display process tries to send the decoded frame to the screen.

Figure 5.8 plots the measured total system energy consumption during video playback. As one might expect, all DVS schemes can save significant energy consumption compared to the case without DVS, and the *optimum* scheme achieves minimum energy consumption.

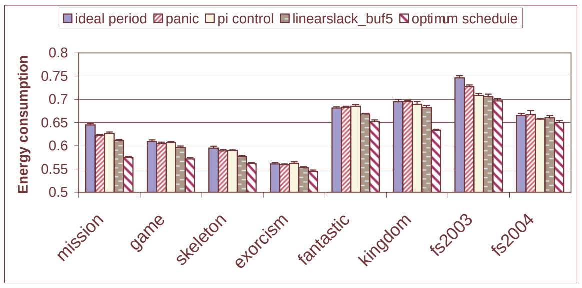


Figure 5.8: Total system energy consumption for video playback with different DVS schemes. The energy consumptions are normalized to those for the *full speed* schedule. [65]

The proposed *linear slack* scheme with buffer size 5 performs roughly equal to or better than the rest DVS schemes in terms of energy saving.

video clips	*panic* (buf5)	*linear slack* (buf5)	*PI controller* (buf10)
mission	94	69	38
game	53	34	19
skeleton	29	0	0
exorcism	10	0	0
fantastic	95	37	0
kingdom	77	4	0
fs2003	30	3	0
fs2004	172	6	0

Table 5.3: Number of frames missing deadline with different DVS schemes. [65]

In the theoretical analysis carried out in Section 5.5, we assume that the worst-case full speed frame decoding time is less than the display interval. However, during practical video play back, there are some frames exhibiting extremely long decoding times. In addition, in the *panic* scheme, accurate worst-case decode time is unavailable in practice. Therefore, a significant number of frames might miss their deadlines. Table 5.3 lists the number of frames missing their deadline in each video clip for different runtime DVS schemes. Though both schemes have the same display buffer size, the *linear slack* scheme does a much better job than the *panic* scheme to hide those extreme large frames. The *PI controller* scheme can further reduce the number of deadline misses at the cost of twice buffer size. Given that the total number of frames in each clip is 3000, the deadline miss rate of the *linear slack* scheme is around or less than 1% for most clips.

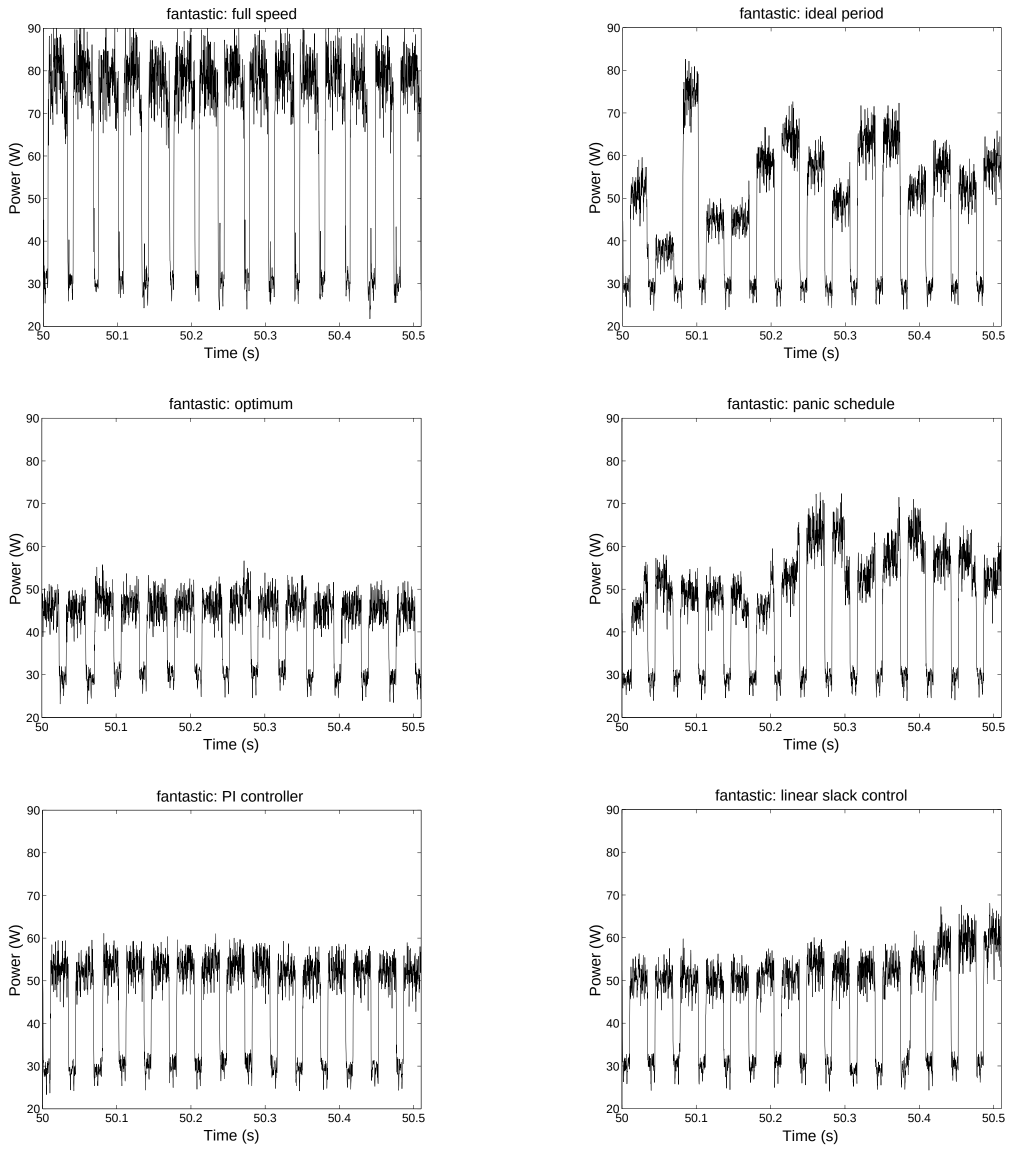


Figure 5.9: Segments of the recorded power trace during playback of video clip *fantastic* (H.264) by various DVS schemes.

Figure 5.9 shows the power trace of a 15-frame segment by different DVS schemes during the video playback. The valleys in the power traces are due to the low CPU operating frequency for the frame display task. The *ideal period* and *panic* schemes create large inter-frame decode power variations which is energy inefficient. On the other hand, *optimum* and *PI controller* schemes produce rather flat decode power curves which are desirable for energy saving. The decode power consumed by the *linear slack control* scheme is gradually increased in the second half of the clip segment because $E[k]$ in the second half of the clip segment becomes larger as indicated by the power trace of the *ideal period* scheme. This figure seems to suggest that the *PI controller* scheme might be superior to the *linear slack control* scheme, which might be true only when the number of frames in the display buffer is within the "dead-zone". The disadvantage of the *PI controller* scheme could be found in the speed schedule of a much longer frame sequence such as those in Figure 5.10.

Figure 5.10 plots the speed schedule by different DVS schemes for video clip *mission* (SVQ3). By knowing all frame decode timing information in advance, the *optimum* schedule can decode a large number of frames at constant speed without causing frame deadline missing, thereby achieving minimum energy consumption. The *ideal period* schedule decodes frame n at speed $u(n)$ such that $\frac{k(n)}{u(n)} + m(n) = T$. Given that the variation of $m(n)$ is much smaller than $k(n)$, we have $u(n) \propto k(n)$. Thus, the large variations in the speed schedule of *ideal period* scheme reflects directly the rapid changes in $k(n)$ (CPU computation time) of the frames due to strong phased behavior in the required decoding effort during movie playback. The *panic* scheme slightly reduces the variations in its speed schedule, therefore consuming less energy than *ideal period* scheme. The proposed *linear slack control* scheme does a better job to reduce the high frequency noise in the speed schedule. The speed variations in this schedule are largely due to the rapid changes in $E[k(n)]$ at different clip segments as illustrated by the *ideal period* speed schedule in the corresponding clip segments. Indeed, the *linear slack control* speed schedule shows that the feedback system designed in this chapter can effectively keep track of the changes of frame timing statis-

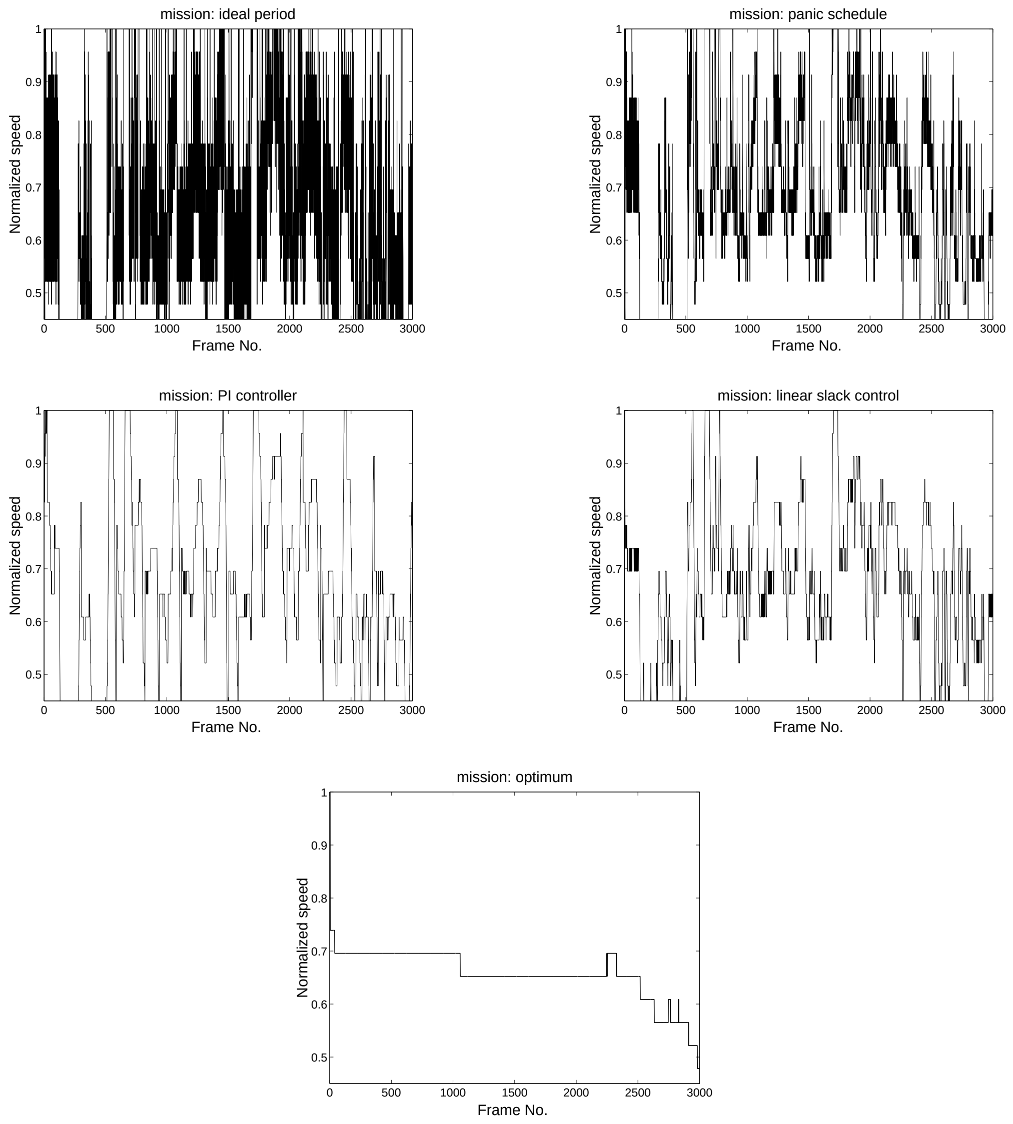


Figure 5.10: Decode speed schedule for video clip *mission* (SVQ3) by various DVS schemes.

tics in different phases during the movie playback. The *PI controller* scheme generates flat speed schedule for small segments of frames when the system is within the specified "dead-zone". However, when the system is outside the "dead-zone", the PI controller plays its role and try to pull the system back into the "dead-zone" with *over-compensated* speed decisions. As a result, too aggressive/conservative speed schedules might be formed as exemplified by the schedule for the frames before 1500, where steep change in frame timing statistics happens.

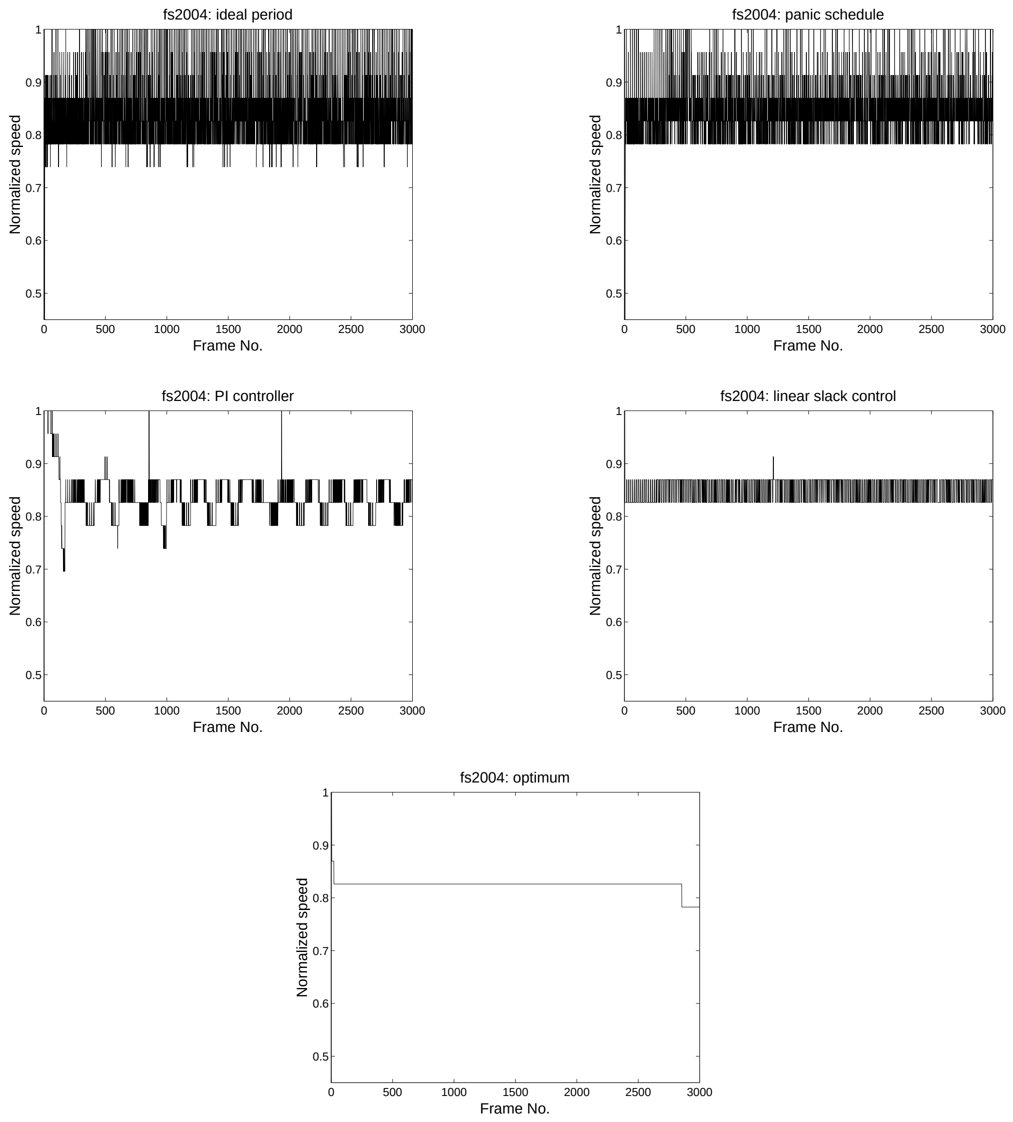


Figure 5.11: Decode speed schedule for video clip *fs2004* (MPEG2) by various DVS schemes.

The decoding process of movie *fs2004* (MPEG2) illustrates other situations in which there are no significant phase variations during the playback, as shown in Figure 5.11. It can be directly interpreted from the speed schedule of the *ideal period* scheme that the average value of $k(n)$ (i.e. the required decoding effort) is rather consistent. This observation can be further confirmed from the small variation in $k(n)$ for this movie clip as indicated in Table 5.1. As expected, the *linear slack control* scheme finds a fairly uniform speed schedule (the actual speed jumps between 0.83 and 0.87 since only discrete speeds are available) indicating that the feedback system is working in the steady state throughout the movie playback. On the other hand, the speed schedule of the *PI controller* scheme is less stable, again due to the over-compensation problem.

5.7.1 Goodness of the proposed scheme

Results from Figure 5.8 show that our new feedback based DVS decoding scheme consumes less energy than the *ideal period* scheme which is assumed to know the full timing information (i.e. k and m) about the current frame. In other words, even perfect predictions on single frame don't help. In some cases, one might be able to predict the timing information for the future several frames [102]. It would be interesting to see how well the proposed feedback scheme performs compared to the situations when several future frames can be looked ahead. As one extreme, the *optimum* scheme gives the best speed schedule when all frames in the movie clip are known in advance. In general, when the timing information of n future frames is available, the best energy-efficient decoding speed can be found by using Equation (5.15). Figure 5.12 compares the estimated decode energy by our proposed feedback scheme and those using different number of look-ahead frames.

The data shown in Figure 5.12 are obtained in the following way. We use our testbed system to measure the averaging decoding power at each CPU speed. We find that the measured power consumption is fairly consistent, independent of the video format being decoded. For each different look-ahead number, the decoding speed for each frame can be derived using Equation (5.15). Given a speed schedule for the movie, we can estimate the decoding time for each frame using the profiled timing information, and further estimate the total decode energy using the pre-measured decode powers for different speeds. Fig-

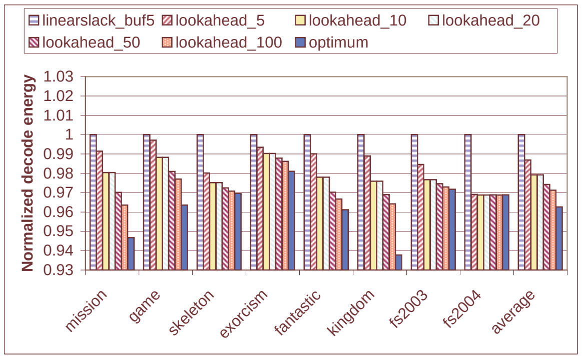


Figure 5.12: Estimated decode energy consumption by the proposed feedback scheme and the schemes with looking-ahead different number of frames.

ure 5.12 reveals that, for most movie clips, the energy difference between our scheme and that knowing the next 5 future frames is less than 2%. This may suggest that unless one can accurately predict the timing information for more than 5 future frames at a time, it is unlikely that one can do a much better job than our scheme which doesn't need predictions at all!

5.8 Conclusion

Although many low-power multimedia playback techniques have been proposed, many of them suffer from idealized assumptions or practical constraints. The work described in this chapter seeks a balanced design by first creating a model for the available design space, then identifying a preferable architecture based on desirable system behaviors, and finally analyzing the design trade-offs using formal methods. The implementation overhead of this new DVS technique is ultra-low and can be easily implemented in both software and hardware based decoding systems.

We implemented the proposed architecture on a prototype hardware platform and compared it with existing techniques. Experimental results show that the proposed new design achieves equal or better energy efficiency than the existing techniques, brings close to ideal playback quality (about a 1% deadline miss rate), and only requires a moderate buffer size (5-frame buffer). Furthermore, our results from the tested real-world video clips suggest that unless one can predict a significant number of future frames at a time, there are unlikely other schemes which can outperform our scheme significantly. Therefore, the proposed new DVS multimedia decoding architecture is indeed a good trade-off among energy, playback quality, hardware cost and playback latency.

Chapter 6

Temperature-aware Dynamic Electromigration Reliability Model [1]

6.1 Introduction

Due to increasing complexity and clock frequency, temperature has become a major concern in integrated circuit design. Higher temperatures not only degrade system performance, raise packaging costs, and increase leakage power, but they also reduce system reliability via temperature enhanced failure mechanisms such as gate oxide breakdown, interconnect fast thermal cycling, stress-migration and electromigration (EM). The introduction of low-k dielectrics in the future technology nodes will further exacerbate the thermal threats [10]. In this chapter, we focus on temperature-related EM failure on interconnects. Other failure mechanisms will be investigated in the future.

The field of temperature-aware design has recently emerged to maximize system performance under lifetime constraints. Considering system lifetime as a resource that is consumed over time as a function of temperature, dynamic thermal management (DTM)

[1]This chapter is based on the published paper titled “Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design” [61].

techniques [95, 97] are being developed to best manage this consumption. While the dynamic temperature profile of a system is workload-dependent [94, 97], several efficient and accurate techniques have been proposed to simulate transient chip-wide temperature distribution [21, 94, 107], providing design-time knowledge of the thermal behavior of different design alternatives. Currently, DTM studies assume a fixed maximum temperature, which is unnecessarily conservative. To better evaluate these techniques and explore the design space, designers need better information about the lifetime impact of temperature.

Failure probability in VLSI interconnects due to electromigration is commonly modeled with lognormal reliability functions. The variability of lifetime is strongly dependent on the interconnect structure geometries and weakly dependent on environmental stresses such as current and temperature [37], while median time to failure (MTF) is determined by current and temperature in the interconnect. In this chapter, we use MTF as the reliability metric and investigate how it is affected by temporal and spatial thermal gradients.

Traditionally, Black's equation 2.3 is widely used in thermal reliability analysis and design. However, Black's equation assumes a constant temperature. For interconnects subject to temporal and/or spatial thermal gradients, two questions need to be answered: 1. Is Black's equation still valid for reliability analysis in these cases? 2. If Black's equation is valid, what temperature should be used? Though in absence of clear answers in the literature, in practice, Black's equation is still widely assumed, and the worst-case temperature profile is usually used to provide safeguard, resulting in pessimistic estimations and unnecessarily restricted design spaces.

In this chapter, we answer the above two questions. We find that, for EM subject to time-varying stresses, Black's equation is still valid, but only with the reliability equivalent temperature/current density that returns from a dynamic reliability model [61]. For EM subject to spatial thermal distribution, Black's equation cannot be applied directly. Instead, we give the bounding temperatures which can be used in Black's equation to bound the actual lifetime subject to non-uniform temperature distribution. Therefore, our results can be seamlessly integrated into current reliability analysis tools based on Black's equation [8]. In addition, while designers are currently constrained by constant, worst-case temperature assumptions, the analysis presented in this chapter provides more accurate, less pessimistic interconnect lifetime predictions. This results in fewer unnecessary reliability design rule violations, enabling designers to more aggressively explore a larger design space.

The rest of the chapter is organized as follows. Section 6.2 introduces a stress-based analytic model for EM, which serves as the base model in this chapter. In Section 6.3, we extend this model to cope with time-varying stresses (i.e., temperature and current) and derive a formula to estimate interconnect lifetime, which we analyze in Section 6.4. In Section 6.5, we analyze the impact of non-uniform temperature distribution on lifetime prediction due to EM. We illustrate how designers can use our analysis to reclaim some design margin by considering runtime variations in Section 6.6. Finally, we summarize the chapter in Section 6.7.

6.2 An analytic model for EM

In this section, we describe the basic EM model used in the chapter. In the following sections, we will extend this basic EM model to predict interconnect lifetime under dynamic thermal and current stresses.

Clement [28] provides a review of 1-D analytic EM models. Several more sophisticated EM models are also available [76, 88]. In this chapter, we only discuss the EM-induced stress build-up model of Clement and Korhonen [27, 51], which has been widely used in EM analysis and agrees well with simulation results using a more advanced model by Ye *et al.* [114].

EM is the process of self-diffusion due to the momentum exchange between electrons and atoms. The dislocation of atoms causes stress build-up according to the following equation [27, 51]:

$$\frac{\partial\sigma}{\partial t} - \frac{\partial}{\partial x}\left(\left[D_a\left(\frac{B\Omega}{kTl^2\varepsilon}\right)\right](\frac{\partial\sigma}{\partial x} - \frac{qlE}{\Omega})\right) = 0 \tag{6.1}$$

where $\sigma(x,t)$ is the stress function, and an interconnect failure is considered to happen when $\sigma(x,t)$ reaches a threshold (critical) value σ_{th}. D_a is the diffusivity of atoms, a function of temperature. B is the appropriate elastic modulus, depending on the properties of the metal and the surrounding material and the line aspect ratio. Ω is the atom volume. ε is the ratio of the line cross-sectional area to the area of the diffusion path. l is the characteristic length of the metal line (i.e., the length of the effective diffusion path of atoms). q is the effective charge. E is the applied electric field, which is equal to ρj, the product of resistivity and current density. The term $\frac{qlE}{\Omega}$ corresponds to the atom flux due to the electric field, while $\frac{\partial\sigma}{\partial x}$ corresponds to a back-flow flux created by the stress gradient to counter-balance the EM flux. And the total atomic flux at a specific location in the interconnect is proportional to the sum of these two components:

$$J = \left[D_a\left(\frac{B\Omega}{kTl^2\varepsilon}\right)\right](\frac{\partial\sigma}{\partial x} - \frac{qlE}{\Omega}) \tag{6.2}$$

Equation (6.1) states that the mechanical stress build-up at any location is caused by the divergence of atomic flux at that point, or $\frac{\partial\sigma}{\partial t} = \nabla J$. If we assume a uniform temperature across the interconnect characteristic length and let $\beta(T) = D_a\left(\frac{B\Omega}{kTl^2\varepsilon}\right)$ (which we refer to as the temperature factor throughout the chapter) and $\alpha(j) = \frac{qlE}{\Omega}$, we obtain the following simplified version, the solution of which depends on both temperature and current density:

$$\frac{\partial\sigma}{\partial t} - \beta(T)\frac{\partial}{\partial x}\left(\frac{\partial\sigma}{\partial x} - \alpha(j)\right) = 0 \tag{6.3}$$

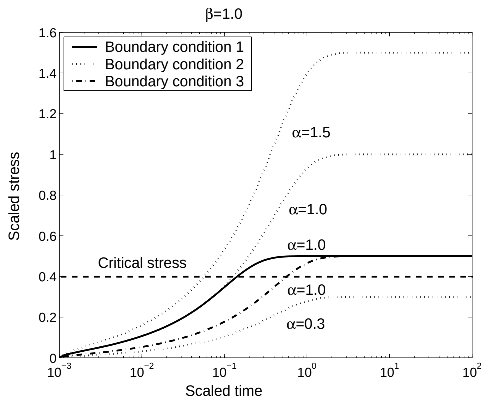


Figure 6.1: EM stress build-up for different boundary conditions and α values. All processes have $\beta = 1$ (α and β are defined in Equation (6.3)). [61]

Clement [27] investigated the effect of current density on stress build-up using Equation (6.3), assuming that temperature is unchanged (i.e., $\beta(T) = constant$), for several different boundary conditions. He found that the time to failure derived from this analytic model had exactly the same form as Black's equation (2.3). The exponential component in Black's equation is due to the atom diffusivity's (D_a's) dependency on temperature by the well-known Arrhenius equation: $D_a = D_{ao} exp\left(\frac{-Q}{kT}\right)$.

Applying the parabolic maximum principles [70] to Equation (6.3), we know that at any time t, the maximum stress along a metal line can be found at the boundaries of the interconnect line. Figure 6.1 shows the numerical solutions for Equation (6.3) at one end of the line (i.e., $x = 0$) for different boundary conditions and α values, all with $\beta = 1$. The three boundary conditions shown here are similar to those discussed in [27] for finite length interconnect lines. It indicates that both boundary conditions and current density (α) affect the stress build-up rate (i.e., the larger the current, the faster the stress builds up.). Also seen from the figure is that the stress build-up saturates at a certain point. This is because, in saturation, the atom flux caused by EM is completely counterbalanced by the stress gradient along the metal line. It is believed that the interconnect EM failure occurs whenever the stress build-up reaches a critical value, σ_{th} (as shown in Figure 6.1). If the saturating stress is below the critical stress, no failure happens. In the following discussion, we assume that the saturating stress in an EM process is always above the critical stress.

6.3 EM under dynamic stress

In this section, we first show that the "average current" model can be used to estimate EM lifetime under dynamic current stress while the temperature is constant. Then we derive a formula to reveal the effect of time-dependent temperature on EM. Finally, based on these two results, we generalize an EM lifetime prediction model accounting for the combined dynamic interplay of temperature and current stresses.

6.3.1 Time-dependent current stress

Clement [27] used a concentration build-up model similar to the one discussed here to verify that in the case in which temperature is kept constant, the average current density can be used in Black's equation for pulsed DC current. As for AC current, an EM effective current is used by the Average Current Recovery (ACR) model [46, 103]. In this study, we do not distinguish between these two cases. We only consider the change of EM effective current due to various causes (e.g., phased behaviors in many workloads). This is because the time scale of the current variation studied here is usually much longer than that of the actual DC/AC current changes in the interconnects.

We numerically solve Equation (6.3) with different time-dependent α functions, and the results are plotted in Figure 6.2. The stress build-ups for all EM processes in Figure 6.2 overlap before saturations (or before reaching the critical stress), since they have the same average current. Thus, the EM process under time-varying current stress can be

[3]For example, the numbers after the circle represent the case in which α is a square-wave function and varies between 3 and 0 with a duty cycle of 0.5. This representation of the time-dependent square-wave function is used in other figures throughout the chapter.

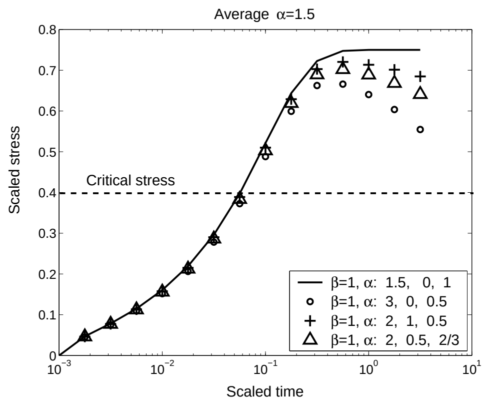


Figure 6.2: EM stress build-up under time-dependent current stress. In each EM process, α (defined in Equation (6.3)) oscillates between two values with different duty cycles. The time dependence of α is given in the legend.[3]All curves have the same average value of α. The solid line is the stress build-up with a constant value of α. [61]

well approximated by average current. Note that the curves in Figure 6.2 diverge after they reach their maximum stress. This is because the time-varying current could not create a stable counterbalancing stress gradient for EM. However, we are only interested in the EM process before reaching the critical stress when EM failure occurs.

6.3.2 Time-dependent thermal stress

If the temperature (β) of the interconnect is time-dependent, we can derive the EM stress build-up expression indirectly based on the following theorem.

Theorem 6.1. *Consider stress build-up Equation (6.3) with constant values for β and α. Let $\sigma_1(x,t)$ be the solution for the equation with $\beta = \beta_1$ under certain initial and boundary conditions and $\sigma_2(x,t)$ be the solution with $\beta = \beta_2$ for the same initial and boundary conditions. If the solutions for Equation (6.3) are unique for those initial and boundary conditions, we have*

$$\sigma_2(x,t) = \sigma_1(x, \left(\frac{\beta_2}{\beta_1}\right) t)$$

Proof. Since $\sigma_1(x,t)$ is the solution for the equation, we have $\frac{\partial \sigma_1}{\partial t}(x, \left(\frac{\beta_2}{\beta_1}\right) t) - \beta_1 \frac{\partial}{\partial x} \left(\frac{\partial \sigma_1}{\partial x}(x, \left(\frac{\beta_2}{\beta_1}\right) t) - \alpha(j) \right) = 0$. On the other hand, let $\sigma_2(x,t) = \sigma_1(x, \left(\frac{\beta_2}{\beta_1}\right) t)$, we have $\frac{\partial\sigma_2}{\partial t}(x,t) = \left(\frac{\beta_2}{\beta_1}\right)\frac{\partial\sigma_1}{\partial t}(x,\left(\frac{\beta_2}{\beta_1}\right)t)$ and $\frac{\partial\sigma_2}{\partial x}(x,t) = \frac{\partial\sigma_1}{\partial x}(x,\left(\frac{\beta_2}{\beta_1}\right)t)$. This leads to $\frac{\partial\sigma_2}{\partial t}(x,t) = \beta_2\frac{\partial}{\partial x}\left(\frac{\partial\sigma_2}{\partial x}(x,t)-\alpha(j)\right)$, which demonstrates that $\sigma_1(x,\left(\frac{\beta_2}{\beta_1}\right)t)$ is the solution for the stress build-up equation with $\beta = \beta_2$, under the same initial and boundary conditions. ■

Theorem 6.1 tells us that the stress build-up processes in the interconnect are independent of the value of β in Equation (6.3). The value of β only determines the build-up speed of the process. For example, at time $\left(\frac{\beta_2}{\beta_1}\right)t$, the stress build-up of an EM process with $\beta = \beta_1$ *sees* the stress build-up of an EM process with $\beta = \beta_2$ at time t. In other words, *it is possible to use the expressions for stress build-up under constant temperature to describe the EM process under time-varying thermal conditions*.

Consider that temperature varies over time, and EM effective current doesn't change. We can divide time into segments, such that temperature is constant within each time segment. In other words, β in Equation (6.3) is a segment-wise function, described as:

$$\beta(t) = \begin{cases} \beta_1, & t \in [0,\Delta t_1] \\ \beta_2, & t \in (\Delta t_1, \Delta t_1 + \Delta t_2] \\ \ldots & \\ \beta_i, & t \in \left(\sum_{k=1}^{i-1}\Delta t_k, \sum_{k=1}^{i}\Delta t_k\right] \\ \ldots & \end{cases}$$

We denote $M0$ as the metal line of interest. Imagine that there is another metal line, denoted by $M1$, having the same geometry and EM effective current as $M0$. $M1$ has a constant value of β equal to $\beta 1$, while $M0$ will experience a time-dependent function of $\beta(t)$. Let $\sigma_0(t)$ and $\sigma_1(t)$ be the stress evolution on metal line $M0$ and $M1$ respectively. During the first time segment, the stress build-ups on both metal lines are the same. Thus, at the end of this time segment, we have $\sigma_0(\Delta t_1) = \sigma_1(\Delta t_1)$. $M0$ will continue to build up stress with β_2 during the second time segment. According to Theorem 6.1, the stress evolution of $M0$ during Δt_2 will be the same as that of $M1$, except that it will take $M1$ a time period of $\frac{\beta_2}{\beta_1}\Delta t_2$ to achieve the same stress. Similar analysis can be applied to other time segments. As a result, at the end of the ith time segment, the stress build-up in $M0$ will be equal to that in $M1$ after a total time of $\sum_{k=1}^{i}(\frac{\beta_k}{\beta_1})\Delta t_k$. In other words, we can convert the stress evolution under time-varying thermal stress into EM stress evolution with constant temperature.

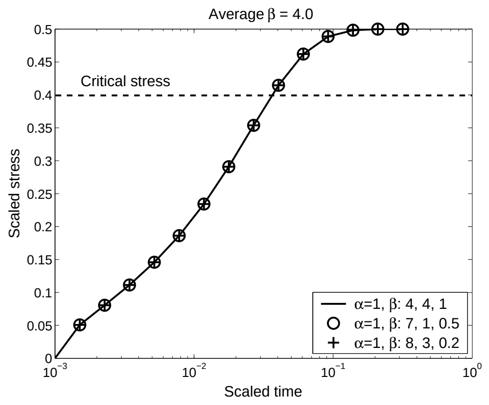


Figure 6.3: EM stress build-up at one end of the interconnect with different time-dependent β functions (square waveform). The solid line is the case with a constant value of β equal to the average value of β in other curve. [61]

It follows that at the end of the ith time segment, the stress in $M0$ is specified as: $\sigma_0(\sum_{k=1}^{i}\Delta t_k) = \sigma_1\left(\sum_{k=1}^{i}(\frac{\beta_k}{\beta_1})\Delta t_k\right)$.

As $\Delta t_i \rightarrow dt$, $\beta_i \rightarrow \beta(T(t))$, we obtain the integral version for the stress build-up function:

$$\sigma_0(t) = \sigma_1\left((\frac{1}{\beta_1})\int_0^t \beta(T(t))dt\right) \tag{6.4}$$

If we assume that the stress build-up reaches a certain threshold (σ_{th}) at which an EM failure occurs, we have:

$$\int_0^{t_{failure}} \beta(T(t))dt = \varphi_{th} \tag{6.5}$$

where φ_{th} is a constant determined by the critical stress (i.e. $\varphi_{th} = \sigma_1^{-1}(\sigma_{th})\beta_1$). If an average value of $\beta(t)$ exists, we obtain a closed form for the time to failure:

$$t_{failure} = \frac{\varphi_{th}}{E(\beta(T(t)))} \tag{6.6}$$

where $E(\beta(t))$ is the expected value for $\beta(t)$, and $\beta(t)$ is the temperature factor, as defined in Equation (6.3), having the form $\beta(T(t)) = A'\left(\frac{exp\left(-\frac{Q}{kT(t)}\right)}{kT(t)}\right)$ where A' is a constant. In comparison with Black's equation, Equation (6.6) indicates that the average of temperature factor β should be used.

One way to interpret Equation (6.5) is to consider interconnect time to failure (i.e., interconnect lifetime) as an available resource, which is consumed by the system over time. Then the $\beta(t)$ function can be regarded as the consumption rate.

Let $MTF(T)$ be the time to failure with a constant temperature T. We have $\beta(T) = \frac{\varphi_{th}}{MTF(T)}$ by Equation (6.6). Substitute this relation in Equation (6.6) again and consider the time-varying temperature, and we obtain an alternative form for Equation (6.6):

$$t_{failure} = \frac{1}{E(1/MTF(T))} \tag{6.7}$$

Equation (6.7) can be used to derive the absolute time to failure provided that we know the time to failure for different constant temperatures (e.g., data from experiments).

By calculating the second derivative of $\beta(T)$ as a function of temperature, it can be verified that $\beta(T)$ is a convex function within the operational temperatures. By applying Jensen's inequality, we have $E(\beta(T)) \geq \beta(E(T))$, which, according to Equation (6.6), leads to an interesting observation: constant temperature is always better in terms of EM reliability than oscillating around that temperature (with the average temperature the same as the constant temperature).

Similar to the methods for verifying the "average current model", we obtain numerical solutions for the stress build-up equation using different square waveforms for β. Figure 6.3 compares these results and shows that the time to failure will be the same as long as the EM processes exhibit the same *average* value of β.

6.3.3 Combined dynamic stress

In reality, both temperature and current change simultaneously. In most cases, the variation of temperature on the chip reflects changes in power consumption, thus directly relating to current flow in the interconnects. In order to describe the EM process in this general case, we can, again, divide time into multiple small segments, and in each time segment, assume that both current and temperature are constant. The temperature and current stresses on the interconnect within time segment Δt_i is denoted by a pair of values (α_i, β_i). Following the same technique as for the time-varying thermal stress, we compare the EM processes in two metal lines ($M0$ and $M1$), and one ($M0$) of which is under time-varying thermal and current stresses. We construct an EM process in the second metal line ($M1$) such that $M1$ is subject to a constant thermal stress ($\beta_{M1} = \beta_1$). Applying Theorem 6.1 reveals that the stress evolution of $M0$ within Δt_i, under (α_i, β_i), is the same as that of $M1$ under stress (α_i, β_1) for a time period of $\frac{\beta_i}{\beta_1}\Delta t_i$. Thus, at the end of the ith time segment, the stress build-up of $M0$ is equal to the stress evolution of $M1$ at the time $\sum_{k=1}^{i}(\frac{\beta_k}{\beta_1})\Delta t_k$. Notice that the current stress on $M1$ is time-dependent (i.e, $\alpha_{M1} = \alpha_i$ for a time period of $\frac{\beta_i}{\beta_1}\Delta t_i$). In order to find the stress of $M1$ at $\sum_{k=1}^{i}(\frac{\beta_k}{\beta_1})\Delta t_k$, the current profile (i.e., α as a function of time) for $M1$ should be considered:

$$\alpha_{M1}(t) = \begin{cases} \alpha_1, & t \in \left[0, \frac{\beta_1}{\beta_1}\Delta t_1\right] \\ \alpha_2, & t \in \left(\frac{\beta_1}{\beta_1}\Delta t_1, \frac{\beta_1}{\beta_1}\Delta t_1 + \frac{\beta_2}{\beta_1}\Delta t_2\right] \\ \cdots & \\ \alpha_i, & t \in \left(\sum_{k=1}^{i-1}\frac{\beta_k}{\beta_1}\Delta t_k, \sum_{k=1}^{i}\frac{\beta_k}{\beta_1}\Delta t_k\right] \end{cases}$$

Since the stress evolution in $M1$ is under constant thermal stress, we may apply the "average current model". As $\Delta t_i \rightarrow dt$, $\beta_i \rightarrow \beta(T(t))$ and $\alpha_i \rightarrow \alpha(t)$, we derive the EM reliability equivalent current for $M0$ (or the average current for $M1$) as:

$$j_{equivalent} = \frac{\int_0^T j(t)\beta(t)dt}{\int_0^T \beta(t)dt} = \frac{E\left[j(t)\beta(t)\right]}{E\left[\beta(t)\right]} \tag{6.8}$$

where T is a relatively large time window, and $j(t)$ is the corresponding current density for $\alpha(t)$. Thus, the EM process in $M0$ can be approximated by an EM process with constant stresses (i.e., $j = j_{equivalent}$ and $\beta = \beta_1$). Using a similar derivation as for Equations (6.4), (6.5), and (6.6), combined with Black's equation, we obtain the time to failure for $M0$:

$$t_{failure} = \frac{C}{j^n_{equivalent} E(\beta(T(t)))} \tag{6.9}$$

where $j_{equivalent}$ is defined by Equation (6.8), and C is a constant.

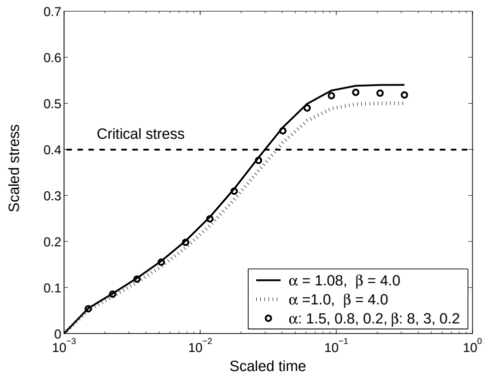


Figure 6.4: EM stress build-up at one end of the interconnect with time-varying α (current) and β (temperature) functions (i.e., square waveforms). The circles represent the numerical solution for time-varying α and β. The solid line is with a constant value of α calculated according to Equation (6.8) and a constant value of β equal to the average value of that in the time-varying case. As a comparison, the EM process (dotted line) simply using the average current of the time-varying case is also shown. These results show that EM process under dynamic stresses (circles) can be well approximated by a process with constant stresses (solid line). [61]

Figure 6.4 compares the stress build-ups for different dynamic current and temperature combinations. These results illustrate that the EM process under dynamic stresses can be well approximated by an EM process with a constant temperature (i.e., $E(\beta)$) and a constant current (i.e., $I_{equivalent}$ as defined in Equation (6.8)). Therefore, for an interconnect with concurrent time-dependent temperature and current stresses, time to failure has the same form as Black's equation, except that the reliability-equivalent current (the actual current modulated by the temperature factor β (i.e., weighted averaging by β)) and the mean value of the temperature factor are used.

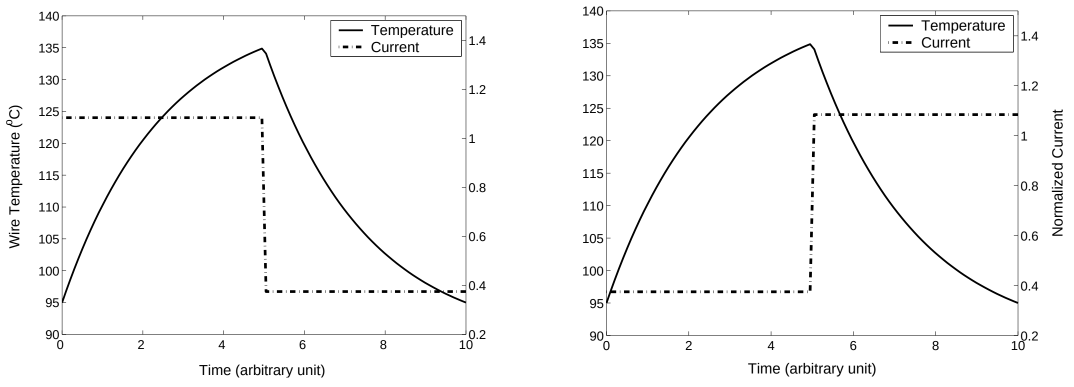
In fact, if the current and the temperature are statistically independent, we have $\frac{E[j(t)\beta(t)]}{E[\beta(t)]} = E[j(t)]$ in Equation (6.8). In this case, the reliability equivalent current will be reduced to the average current and we get back to the "average current model". On the other hand, if the current is constant, Equations (6.8) and (6.9) will lead us to Equation (6.6). Finally, if both temperature and current are time invariant, Black's equation (Equation (2.3)) is obtained.

6.4 Analysis of the proposed model

Equations (6.8) and (6.9) form the basis of our proposed EM model under concurrent time-varying temperature and current stress. In this section, we use these equations to evaluate EM reliability. Specifically, we compare the reliability of constant temperature with that of fluctuating temperature, and we show the difference of lifetime projection between our model and the traditional worst-case model.

For any two temporal temperature and current profiles we can easily compare the EM reliability, using our model, by:

$$\frac{MTF_1}{MTF_2} = \frac{j^2_{equivalent2} E(\beta(T_2(t)))}{j^2_{equivalent1} E(\beta(T_1(t)))}$$

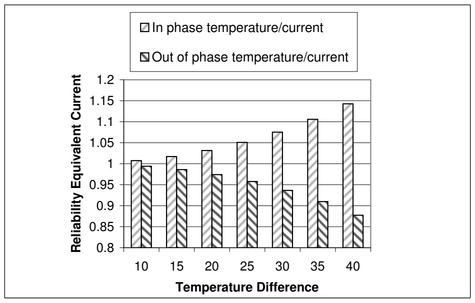


(a) (b)

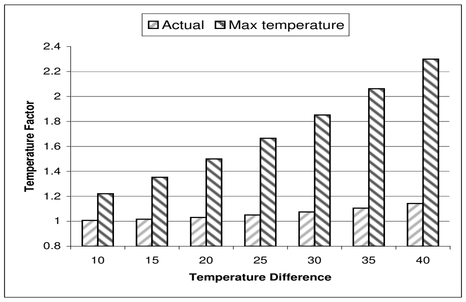
Figure 6.5: Temperature and current waveforms analyzed: (a) in phase current/temperature, (b) out of phase current/temperature. [61]

where MTF_1 is the time to failure under time-varying temperature profile $T_1(t)$ and electric current profile $j_1(t)$.

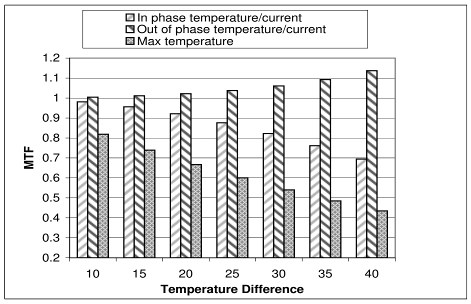
As shown in Figure 1.2, in real workload execution, temperature changes along with the changes in power consumption (i.e. current). It is interesting to see how the interactions between temperature and current profiles affect the interconnect lifetime. The possible interactions between temperature and current form a spectrum, and the plots in Figure 6.5 show the two extremes of this spectrum. In this figure, a simple assumption is made that the current is proportional to the difference between the steady substrate temperature and the ambient temperature (i.e., 40^oC). The temperature difference between the substrate and the interconnects is fixed to be 21^oC, which is a reasonable assumption for high-layer interconnects [22]. Using the data from Figure 1.2, the maximum temperature of the substrate is assumed to be 114^oC (i.e., 135^oC at the interconnects), and we change the minimum temperature to obtain different temperature/current profiles. Using these profiles, we can compare the reliability equivalent current with the average current, compare the temperature factor using our model with those of average and maximum temperatures, and finally compare the MTFs in these cases (i.e., average current/average temperature, reliability



(a)



(b)



(c)

Figure 6.6: Comparison of electric current, temperature factor (β) and MTF for different peak to peak temperature cycles. All results are normalized to the average current and/or temperature case. (a) Ratio of reliability equivalent current (our model) to average current. Both cases of current variation (in and out of phase with temperature) are included. (b) Ratios of temperature factor (β) using average temperature, max temperature, and our model. (c) Comparison of MTF for four different calculations: average temperature/average current, maximum temperature/average current, our model for current in phase with temperature, and our model for current out of phase with temperature.

equivalent current/average temperature factor (β), and average current/maximum temperature).

Our results are reported in Figure 6.6, and we summarize our observations as follows:

- As the peak to peak temperature difference is small, both the reliability equivalent current and the temperature factor predicted by our dynamic stress model are very close to those calculated from using average current and average temperature. That is because the temperature factor function (β), although an exponential function of temperature, can be well approximated by a linear function of temperature within a small temperature range. Thus, the MTF predicted by using average temperature/current provides a simple method for reliability evaluation with high accuracy.

- As the temperature difference increases, we can no longer simply use average temperature/current for MTF prediction. Both the reliability equivalent current and the temperature factor increase (degrading reliability) quickly as the temperature difference increases.

- On the other hand, using maximum temperature always underestimates the lifetime, resulting in excessive design margins.

- One interesting phenomenon arises in the case in which the current is out of phase with temperature variation. Recall that the reliability equivalent current is actually a temperature factor weighted average current, and high temperature increases the weights for the accompanied current. Thus, the reliability equivalent current is reduced compared to the case in which temperature/current are synchronized. This brings a non-intuitive effect on the reliability projection—MTF even slightly increases as the temperature cycling magnitude increases.

In the above discussion, the duty cycle of the current waveform is fixed (i.e., 0.5). We also investigated the effects of different duty cycles, but the data is not shown here due to space limitations. In general, when the temperature change is small (e.g., within $10^{o}C$), using the average temperature to predict lifetime is still a good approximation (less than 5% error) regardless of the duty cycle. While the temperature variation increases, the difference between our model and using average temperature is largest at a duty cycle of about 0.4. On the other hand, the smaller the duty cycle, the larger the difference between our model and using maximum temperature. Thus, using maximum temperature is reasonable only when the duty cycle is large (i.e., higher temperature dominates almost the entire cycle).

6.5 Electromigration under spatial temperature gradients

In addition to temporal temperature variations, large temperature differences across the chip are commonly seen in modern VLSI design. Ajami *et al.* [7] showed that non-uniform temperature has great impacts on interconnect performance. In this section, we will illustrate the importance of considering spatial temperature gradients for interconnect reliability.

6.5.1 EM model with spatial thermal gradients

Due to the exponential dependence of diffusivity on temperature, EM in interconnects with spatial temperature gradients has quite different characteristics than those with constant temperature. Guo *et al.* [36] reported that EM in aluminum interconnect is strongly affected by the relative direction of electron wind and thermal gradients, while Nguyen *et al.* [74] found that temperature gradients greatly enhance EM in aluminum interconnect. Following the stress build-up model introduced in Section 6.2, the atomic flux due to EM can be modeled by $J = \beta(T)\left(\frac{\partial\sigma}{\partial x} - \alpha(j)\right)$, and the stress build-up at a specific location is caused by the divergence of atomic flux at that location, i.e. $\frac{\partial\sigma}{\partial t} = \nabla J$. When the temperature is uniform across the interconnect, i.e. $\beta(T)$ is independent of location, Equation (6.3) is obtained. When the temperature is not uniform, the following equation is derived to describe the stress build-up under thermal gradients:

$$\frac{\partial \sigma}{\partial t} - \beta(T(x))\frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial x} - \alpha(j)\right) - \frac{\partial \beta(T(x))}{\partial x}\left[\frac{\partial \sigma}{\partial x} - \alpha(j)\right] = 0 \qquad (6.10)$$

where σ, β and α are defined in Section 6.2. When compared with Equation (6.3), Equation (6.10) introduces a third term $\frac{\partial \beta(T(x))}{\partial x}\left[\frac{\partial \sigma}{\partial x} - \alpha(j)\right]$, which captures the atomic flux divergence induced by spatial thermal gradients along the interconnect. Though temperature gradient itself will cause migrations of atoms from high temperature to low temperature, a phenomenon called thermomigration (TM), the atomic flux due to TM is generally believed to be much smaller than that due to EM [74]. Therefore TM is not explicitly modeled in Equation (6.10). Jonggook *et al.* [50] investigated EM in aluminum (Al) interconnects subject to spatial thermal gradients. They modeled EM from a different approach but yielded an equation with a form similar to ours. Since dual-damascene Cu interconnects have become the mainstream technology in modern VLSI design and have quite different EM characteristics from Al [75], in the following, we focus on EM failure in copper interconnects.

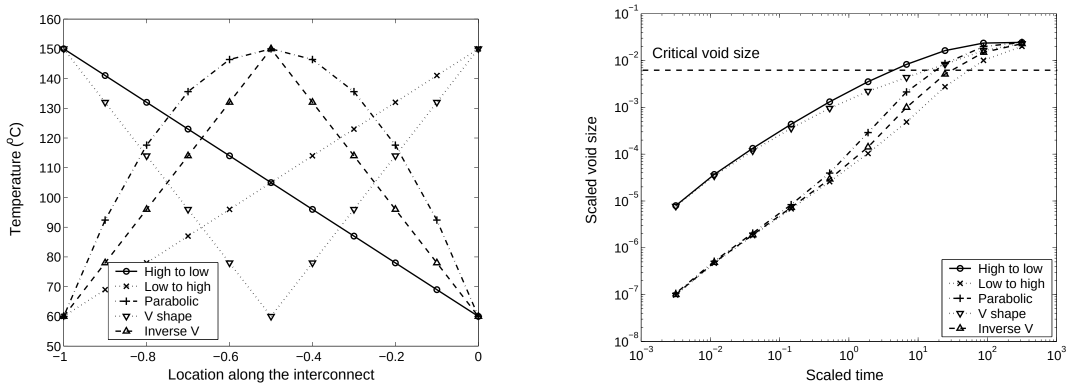
Various experiments [25,37] showed that, in copper interconnect, voids tend to nucleate at the cathode end (near the via), and void growth is the dominant failure process because the critical mechanical stress for void nucleation in copper is much smaller than that for aluminum. With spatial thermal gradients in the interconnect, it is possible that the location of void nucleation is no longer at the cathode end. However, in this case, void growth tends to be slower than that at the cathode, because there are atomic fluxes both going into and coming from the void in the middle of the interconnect [25]. Bearing these observations in mind and assuming a void-growth dominated failure, we choose a boundary condition for Equation (6.10) to model void growth at the cathode such that the mechanical stress at the cathode end is zero (free stress at void) and the atomic flux at the other end is zero (complete blockage for atomic flux), or:

$$\sigma(x=-l,t)=0,\ J(x=0,t)\Rightarrow \left[\frac{\partial\sigma}{\partial x}-\alpha(j)\right]\Bigg|_{x=0}=0$$

where $x=-l$ is the cathode end. This boundary condition is consistent with that used by Clement [28] to model void growth due to EM. The void size at time t can be approximated by [28]:

$$\Delta l \approx \int_{-l}^{0}\frac{-\sigma(x,t)}{B}dx$$

where $\sigma(x,t)$ is the mechanical stress (tensile stress) developed along the interconnect at time t and B is the elastic modulus. Because we are unaware of any closed form solution for Equation (6.10) with the above boundary condition, we use numerical solutions to analyze the impact of thermal gradients on electromigration.

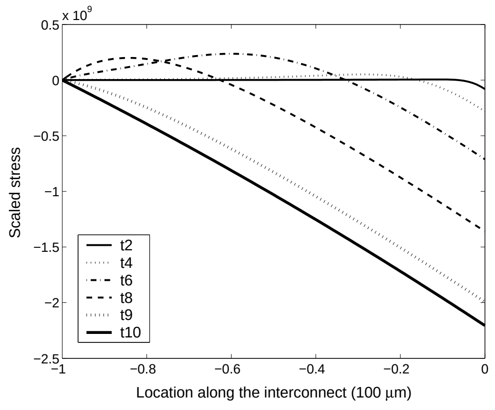


(a) (b)

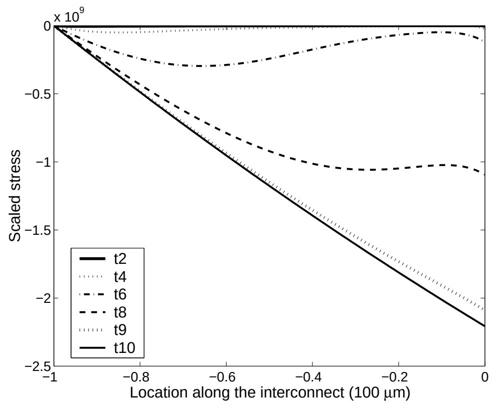
Figure 6.7: Effects of non-uniform spatial temperature distribution on EM induced void growth. (a) Various temperature profiles along a $100\mu m$ copper interconnect (left end is the cathode). (b) Void growth with different spatial temperature profiles.

The temperature spatial profile along an interconnect is the combined effects of joule heating and substrate temperature distributions. Figure 6.7 (a) plots several temperature profiles used in our study and their effects on EM induced void growth. The length of the interconnect is $100\mu m$, and elections are assumed to flow from the left end (cathode) to the right end of the interconnect. Though all temperature profiles have the same maximum and minimum temperatures, their void growth differs greatly due to the different thermal gradients along the interconnect (Figure 6.7 (b)), resulting quite different failure time. In order to investigate how thermal gradients affect EM induced void growth, we also plot, in Figure 6.8, the mechanical stress build-up along the interconnect at different times, with different thermal profiles. In spite of different temperature profiles on the interconnect, in the final EM process stage ("t10" in Figure 6.8), a steady stress gradient is built up to counter-balance the driving force of electron wind, i.e. $\frac{\partial\sigma}{\partial x} - \alpha(j) = 0$, resulting in voids with comparable saturation sizes (Figure 6.7 (b)).

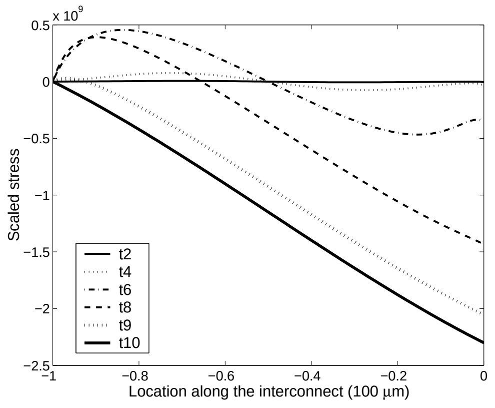
However, as shown in Figure 6.8, the kinetic aspects of stress build-up for different temperature profiles are quite different, especially when the relative direction of electron flow and temperature gradients changes. For a "low to high" temperature profile, the temperature increases linearly from the cathode end to the anode end (shown in Figure 6.7 (a)). At the early EM stage, as indicated by "t2" and "t4" in Figure 6.8 (a), the stress gradient near the cathode is negligible. Therefore the atomic flux at the cathode is only determined by the electron wind at the temperature of that location, because the atomic flux is the sum of the fluxes induced by the stress gradient and the electron wind (Equation (6.2)). Therefore, in this case, the void growth at the cathode is subject to almost the same kinetics as those with a uniform temperature across the interconnect. Later on in the EM process, the effect of thermal gradients begins to play its role. As shown by "t6" and "t8" in Figure 6.8 (a), tensile (positive) stress is built up from the cathode end towards the other end, due to the increasing temperature from the cathode end. The stress gradient created by this tensile stress distribution forms an atomic flux in the same direction as the electron wind. Thus, void growth in the cathode end is enhanced later by the increasing temperature. On the contrary, Figure 6.8 (b) shows quite different kinetics for EM process with "high to low" temperature profile, where the temperature is decreasing linearly from the cathode end



(a)



(b)



(c)

Figure 6.8: Stress build-ups at different time points along the interconnect under spatial thermal gradients. (a) Low to high temperature profile. (b) High to low temperature profile. (c) Parabolic temperature profile. Electrons flow from the left to the right, causing compressive stress (negative in the figures) on the right side of the interconnect. The left end (cathode) is stress free to model the growth of a void. The time points (“t2” through “t10”) are corresponding to the time points in the plots with the same temperature profile in Figure 6.9.

(shown in Figure 6.7 (a)). In the early stage, as in the case for a “low to high”profile, void growth is similarly dominated by the temperature at the cathode end, as illustrated by the stress distributions at “t2” and “t4” in Figure 6.8 (b). Subsequently, compressive (negative) stress is built up from the cathode towards the anode, because of the decreasing temperature from the cathode, as shown by the stress distributions at “t6” and “t8” in Figure 6.8 (b).

The stress gradient due to the compressive stress distribution in this case creates an atomic flux in the opposite direction of the electron wind, retarding the void growth at cathode. The stress distributions induced by EM with a “parabolic” temperature spatial profile at different time points are drawn in Figure 6.8 (c). The kinetics of stress build-up in this case are similar to those in a “low to high” temperature profile, because both temperature profiles have similar temperature gradients near the cathode. On the other hand, in the late stage of the EM process (as indicated by “t9” and “t10” in Figure 6.8), regardless of the temperature distribution across the interconnect, significant stress gradient is formed in the opposite direction of atomic flux and slows down the void growth, and finally the steady state of the EM process is reached (or void growth saturates). In summary, in the early stage of the EM process, the void growth is largely dependent on the temperature at the cathode, while later on, the void growth is enhanced or retarded depending on the temperature gradient near the cathode. Finally, void growth is suppressed by the back-flow stress gradient just like in the case of the EM process with a uniform temperature distribution.

6.5.2 Empirical bounds for void growth with non-uniform temperature distribution

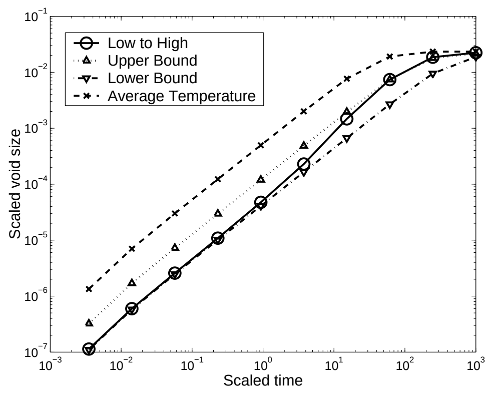
In Section 6.3, our analysis reveals that the EM process with time-varying temperature variations can be approximated by an EM process using a constant reliability equivalent temperature T_{eq}, as long as $\beta(T_{eq}) = E[\beta(T(t))]$. However, in the case where there is a non-uniform temperature across the interconnect, we cannot find a similar reliability equivalent temperature, due to the difference in the EM kinetics in the different stress build-up stages as shown in Figure 6.8. Instead, we try to find two constant temperatures, such that the void growth due to EM with non-uniform temperature can be bounded by the void growth with uniformly distributed temperature equal to these two bounding temperatures respectively.

The reason for our approach is as follows. Because Black's equation is only valid for a uniform temperature distribution, and many existing reliability analysis tools are based on Black's equation, by providing the bounding temperatures for interconnects subject to spatial thermal gradients, one can still use these tools to evaluate the effects of non-uniform temperature distributions.

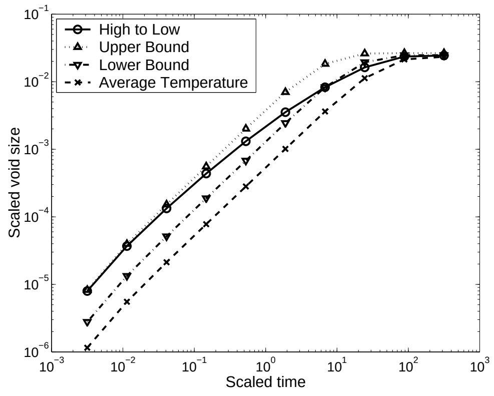
Following our previous discussion on Figure 6.8, one can expect that the cathode temperature can serve as the lower/upper bound temperature for void growth with increasing/decreasing temperature towards the anode end. On the other hand, the void size is proportional to the amount of atoms moved from the cathode end and the void growth rate is determined by the atomic flux at the cathode. We would like to find the other bounding temperature by bounding the atomic flux at the cathode. Consider an interconnect of length l subject to a certain spatial temperature profile $T(x)$, with both ends at zero stress $\sigma(0,t) = \sigma(l,t) = 0$. In the steady state, there is a uniform atomic flux flowing through the interconnect, expressed as (see Appendix B):

$$J_{steady} = -\frac{\int_0^l \alpha(T(x))dx}{\int_0^l \frac{1}{\beta(T(x))}dx}$$

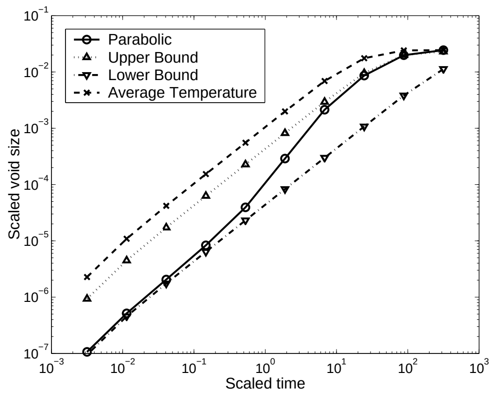
By examining the steady-state stress distribution along the interconnect in the above case, it can be further verified that when the temperature is increasing from the cathode, tensile stress is built up from the cathode, and the atomic flux at the cathode is enhanced by the stress gradients, a situation similar to "t6" and "t8" in Figure 6.8 (a). When the temperature is decreasing from the cathode end, atomic flux at the cathode is retarded by the stress gradient, similar to "t6" and "t8" in Figure 6.8 (b). Therefore we propose to use J_{steady} to bound the atomic flux at the cathode of the interconnect subject to a non-uniform temperature distribution. However, the stress at the anode is not free (Figure 6.8), which does not



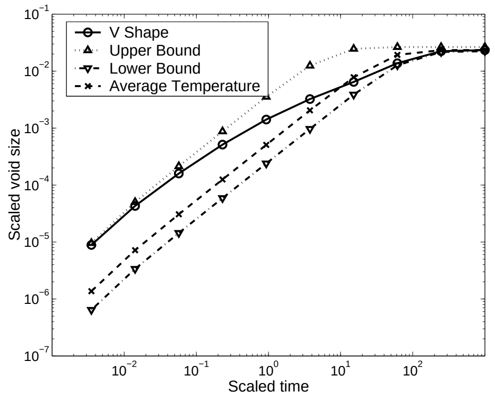
(a)



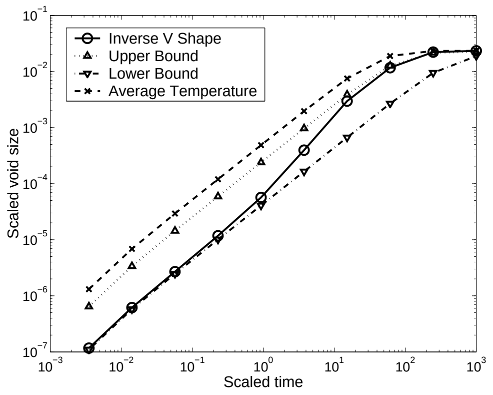
(b)



(c)



(d)



(e)

Figure 6.9: Void growth with non-uniform temperature distribution is bounded by those with a uniformly distributed temperature. (a) Low to high temperature profile. (b) High to low temperature profile. (c) Parabolic temperature profile. (d) V shape temperature profile. (e) Inverse V shape temperature profile.

satisfy the boundary condition for J_{steady}. We instead propose to use half length (the half from the cathode) of the interconnect to calculate J_{steady}, as Figure 6.8 shows that the stress at the middle of the interconnect is close to zero most of the time during the EM process. Finally the bounding temperature is determined in such a way that the atomic flux due to electron wind at this temperature is equal to the calculated J_{steady}.

Temperature gradient at cathode	Lower Bound	Upper Bound
Increasing temperature in the current direction	$T_{lb} = T_{cathode}$	$\beta(T_{ub})(\alpha(T_{ub})) = \frac{\int_{-l}^{-\frac{l}{2}} \alpha(T(x))dx}{\int_{-l}^{-\frac{l}{2}} \frac{1}{\beta(T(x))}dx}$
Decreasing temperature in the current direction	$\beta(T_{lb})(\alpha(T_{lb})) = \frac{\int_{-l}^{-\frac{l}{2}} \alpha(T(x))dx}{\int_{-l}^{-\frac{l}{2}} \frac{1}{\beta(T(x))}dx}$	$T_{ub} = T_{cathode}$

Table 6.1: Proposed bounding temperatures for void growth in an interconnect with length l subject to a non-uniform temperature distribution. T_{lb} is the lower bound. T_{ub} is the upper bound. $T(x)$ is the temperature profile. $x = -l$ and $x = 0$ are the locations of the cathode and the anode, respectively (as shown in Figure 6.7).

We would like to point out that since void growth at the cathode is only dependent on the atomic flux nearby, the temperature gradient near the cathode plays the major role in determining (enhancing or retarding) the void growth, while the temperature distribution far away from the cathode is not as important. This observation is verified by testing with various temperature profiles. (Due to space limitations, we cannot show them all here.) Therefore, in the above discussion, we focus on the temperature gradient near the cathode without assuming any specific temperature distribution along the second half of the interconnect. Table 6.1 numerates the proposed formulas to calculate bounding temperatures for void growth with non-uniform temperature distributions, and only the temperature gradient near the cathode is used to choose the appropriate bounding formula. The void growth with different temperature profiles as well as those with uniform temperatures are compared in

Figure 6.9. In these plots, the void growth with interconnect thermal gradients is closely bounded by the void growths with the proposed uniform bounding temperatures. Blindly using the average temperature to evaluate the EM lifetime will either overestimate (e.g. Figure 6.9 (a)) or underestimate (e.g. Figure 6.9 (b)) the void growth, let alone using the maximum temperature. Wachnik *et al.* [106] demonstrated that it is possible to construct an electromigration resistant power grid by using shorter interconnect segments because of the Blech effect [75]. This finding seems to imply that, under normal operating conditions, the critical void size causing EM failure should be at a similar order of magnitude as that of the saturation void size (e.g., as the case shown in Figure 6.7). Therefore, for increasing/decreasing temperature at the cathode, the upper/lower bound temperature could serve as a good estimation of the interconnect lifetime with a non-uniform temperature distribution.

Joule heating in an interconnect usually results in a symmetric temperature distribution with the maximum temperature in the middle, due to the much lower thermal resistance of the vias on both ends. Therefore, the symmetric temperature distributions along the interconnect are of more practical interest. The“parabolic” and “inverse V shape” temperature profiles shown in Figure 6.7 (a) are used to approximate this kind of temperature distributions. Interestingly, for these temperature distributions, as indicated by Figure 6.9 (c) and (e), even the upper bound temperature for void growth is lower than the average temperature. In the EM measurements of copper interconnects performed by Meyer *et al* [71], they considered the non-uniform temperature distribution due to self-heating, and tried to fit their measurements with Black’s equation by using different temperatures (e.g. maximum, average, weighted average, via (minimum) temperature). They reported that the best fit temperature is strongly weighted to the via temperature. Their findings agree with our analysis presented here.

6.5.3 Effects of combined temporal and spatial temperature gradients

So far we have discussed the interconnect lifetime prediction under temporal and spatial temperature distributions separately. In practice, due to circuit activity variations, one might expect the spatial temperature distribution over an interconnect would change over time. The electromigration diffusion equation (Equation (6.1) or Equation (6.10)) can be extended to capture this situation by assuming that temperature T is a function of both time and interconnect location. However, we cannot obtain a closed form analytic solution in this highly complex scenario. Instead, we propose to combine the results we have found so far in the cases of temporal gradients only and spatial gradients only to estimate the interconnect lifetime subject to both temporal and spatial temperature gradients.

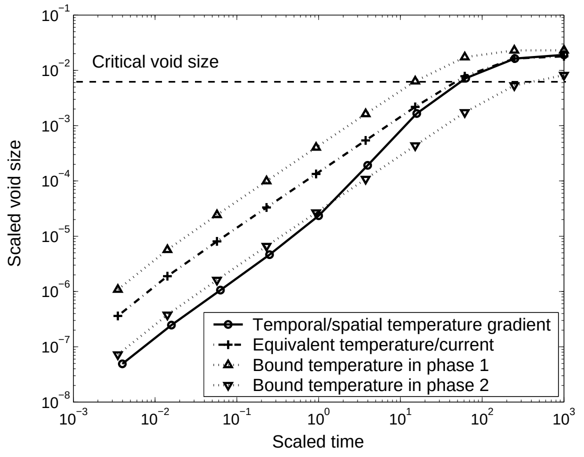


Figure 6.10: Void growth subject to both temporal and spatial thermal gradients can be bounded by that using uniform temperature and current.

At time t, the temperature profile across an interconnect is denoted by $T(t,x)$, and the current density is $j(t)$. Using the formulas in Table 6.1, we can find the bounding temperature at t, denoted by $T_b(t)$. Applying Equations (6.8) and (6.9) for the case of temporal temperature gradients to $T_b(t)$ and $j(t)$, we could find an equivalent uniform temperature and current to approximate the void growth subject to both temporal and spatial temperature gradients. Figure 6.10 shows one example. In this example, the interconnect Cu line is subject to two “parabolic” temperature profiles, with each one for half the time (i.e., 50% duty cycle), denoted by “phase 1” and “phase 2” in the figure. We solve Equation (6.10) numerically with temperature being a function of both time and space, and we plot the void growth. This figure indicates that the void growth subject to the time varying temperature profile can be bounded by that using time invariant uniform temperature and current, as calculated according to the procedures proposed here. As a comparison, we also plot the void growth at the bound temperature of each temperature profile alone. If the critical void size is close to the saturation void size, as shown in the figure, one can use the calculated equivalent temperature and current to estimate the interconnect lifetime subject to both temporal and spatial thermal gradients, using Black’s equation (Equation (2.3)). We have also testes this for other temperature profiles and duty factors, and the results are similar but are not presented here due to space limitations.

6.6 Design time optimization considering runtime stress variations

In the traditional IC design flow, static and dynamic analyses are performed for the initial design to determine current loading information. Then this information is combined with the worst-case temperature to find those design points violating the reliability specification [84]. However, as we have shown above, using worst-case temperature is too conservative and could result in excessive design margins. Here we propose a design flow incorporating runtime stress information as shown Figure 6.11. In this design flow, the actual or projected current and temperature loads are fed into an accurate reliability model, such as the one proposed in this chapter. We expect that the reliability projection from these models will generally enable more relaxed design constraints and provide a wider design space.

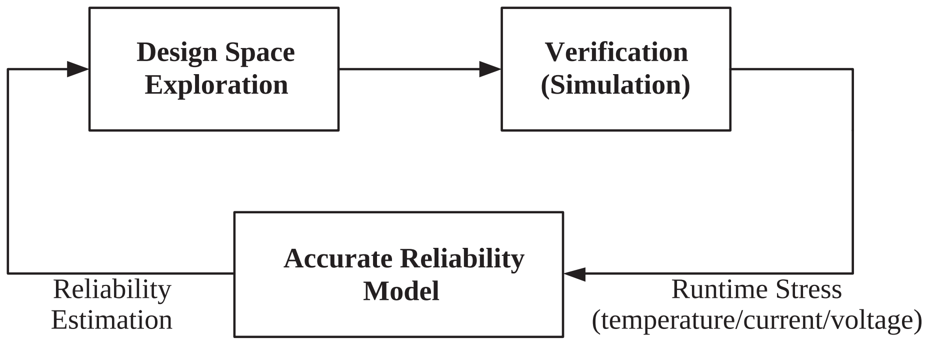
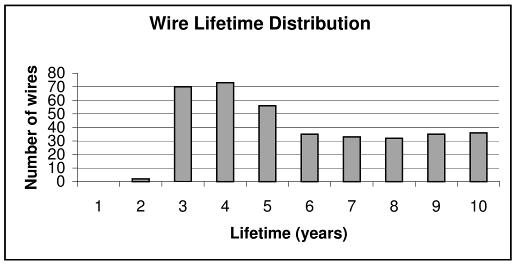

Design Space Exploration
Verification (Simulation)
Accurate Reliability Model
Reliability Estimation
Runtime Stress (temperature/current/voltage)

Figure 6.11: A proposed design flow incorporating runtime stress information. [61]

For instance, when temperature fluctuates within a relatively small range (e.g., 10^oC), our model predicts that using average temperature is good enough for reliability evaluation. Therefore, we could potentially reduce the number of design points falsely flagged for design rule violations when using the worst-case temperature. One example is illustrated in Figure 6.12 using data from a power grid design [108]. In this example, the worst-case temperature of a design is 135^oC, and Wang *et al.* [108] showed that there were a total of 372 wires violating the reliability requirement by using that worst-case temperature. However, if runtime stress information is available at design time, we can move some wires that are outside the specified reliability threshold (10 years of MTF at 135^oC in this example) into the reliable bins by re-calculating the lifetime distribution using our dynamic reliability model. Equivalently, we can shift the reliability threshold towards fewer years on the original wire lifetime distribution diagram. Using the results in Figure 6.6(c), we can estimate the benefits obtained, in terms of design margin reclamation, by considering runtime temperature fluctuations. These results are shown in Figure 6.12(b).

This example only illustrates some potential advantages in design optimization offered by our dynamic reliability model. We expect that our model can be integrated into existing reliability-aware design flows, such as the power grid optimization method proposed by



(a)

Temperature variation (^oC)	New reliability threshold (years)	Number of wires reduced	Percentage reduction
5	9.06	33	8.8
10	8.34	59	15.9
15	7.73	79	21.2
20	7.24	95	25.5
25	6.85	107	29.8

(b)

Figure 6.12: (a) Distribution of wires violating the MTF specification using maximum temperature (data extracted from [108]) with a total of 372 wires. (b) Reduction of the number of wires violating the MTF specification under different temperature variations (maximum temperature: 135^oC). [61]

Wang *et al.* [108].

6.7 Summary

This chapter presented an analysis of interconnect EM failures subject to temporal and spatial thermal gradients. For EM under time-varying stresses (temperature/current), we proposed a dynamic reliability model, which returns reliability equivalent temperatures/currents. For EM under non-uniform temperature distributions, we obtained close bounding temperatures to estimate the actual lifetime. Therefore, the commonly used Black's equation is still applicable by using our constant reliability equivalent temperatures. Our analysis reveals that blindly using the maximum or average temperature to evaluate EM lifetime is inappropriate. Our results not only increase the accuracy of reliability estimates but enable designers to more aggressively explore the design space and to reclaim the design margin imposed by less accurate, more pessimistic models. Existing constant-temperature models require designers to observe a static worst-case temperature limit, but the analysis presented here enables temperature-aware designers to evaluate the system reliability using runtime information, thus increasing the confidence about the actual behavior of the system. The dynamic nature of our reliability model also makes it suitable for DTM, which will be the subject of the next chapter.

Chapter 7

Temperature-aware Runtime Reliability Management for High Performance Systems [1]

7.1 Introduction

The advance of technology scaling (along with resulting increases in power density) has made thermal-related reliability a major concern in modern IC design. For example, in the deep-submicron region, experts widely regard electromigration as a dominant failure mechanism. Designers must therefore rely on temperature-dependent reliability models to derive the expected lifetime of their circuits, increasing design margins (for example, wire width) as necessary to meet requirements.

Worst-case power dissipation and environmental conditions are rare for general-purpose microprocessors. Designing the cooling solution for the worst case is wasteful. Instead, the cooling solution should be designed for the worst "expected" case. In the event that environmental or workload conditions exceed the cooling solution's capabilities and temperature rises to a dangerous level, on-chip temperature sensors can engage some form

[1]This chapter is based on the published paper titled "Improved Thermal Management with Reliability Banking" [64].

of “dynamic thermal management” (DTM) [16, 94, 96], which sacrifices a certain amount of performance to maintain reliability by reducing circuit speed whenever necessary.

During execution, many programs or workloads exhibit temperature fluctuations caused by inherently phased behaviors. Existing DTM techniques ignore the effects of temperature fluctuations on chip lifetime and can unnecessarily impose performance penalties for hot phases. The disadvantages of these techniques become more obvious in systems such as Web servers, in which hot phases usually imply an increased number of service requests. The engagement of cooling mechanisms then affects the server’s quality of service.

In this chapter, using electromigration as the targeted failure mechanism, we propose runtime dynamic reliability management (DRM) techniques based on our dynamic reliability model [64] introduce in the previous chapter. By leveraging this model, one can dynamically track the “consumption” of chip lifetime during operation. In general, when temperature increases, lifetime is being consumed more rapidly, and vice versa. Therefore, if temperature is below the traditional DTM engagement threshold for an extended period, it may be acceptable to let the threshold be exceeded for a time while still maintaining the required expected lifetime. In effect, lifetime is modeled as a resource that is being “banked” during periods of low temperature, allowing for future withdrawals to maintain performance during times of higher operating temperatures. Using electromigration as an example, we show the benefits of lifetime banking by avoiding unnecessary DTM engagements while meeting expected lifetime requirements.

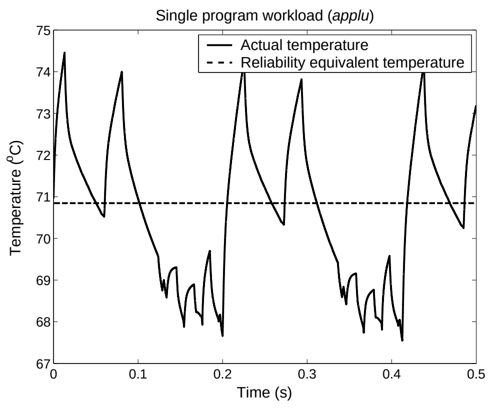
High temperature limits the circuit performance directly by increasing interconnect resistance and reducing carrier mobility. However, it has been shown that ([94]) using DTM to compensate the temperature dependency of clock frequency induces very mild performance penalty. On the other hand, Banerjee *et al.* [10] showed that temperature induced reliability issue tends to limit the circuit performance in future technology generations. In the following discussion, we assume that the temperature threshold is set solely for reliability specification, and circuits can operate correctly above this threshold whenever allowed. Although extreme high temperature may cause immediate thermal damage for IC circuits, we study a range of operating temperatures only with long-term reliability impacts (i.e. temperature induced aging). High temperatures causing immediate or unrecoverable damage are assumed to be far above the range of normal operating temperatures studied here (e.g. the temperature used in accelerated EM test is usually around 200^0C [106]). Therefore, we assume that a monitoring and feedback mechanism is implemented at runtime to ensure that circuits operate well below such temperatures.

7.2 Related Work

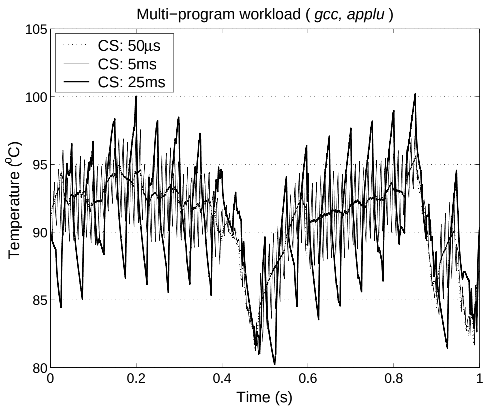
The concept of dynamic reliability management is first introduced by Srinivasan *et al.* [97]. In their work, they proposed a chip level reliability model and showed the potential benefits by trading off reliability with performance for individual applications. They assumed an oracular algorithm for runtime management in their study, and they did not consider the effects of inter-application thermal behaviors on reliability. Later work from the same authors [98] refined their reliability model and showed the improvement in reliability using redundant components. In this chapter, we focus on practical runtime management techniques for the worst-case on-chip component (i.e. hottest interconnect) to exploit both intra- and inter-application temperature variations. The combination of their model and our techniques is expected to bring more advantages and is open for future investigation. Ramakrishnan and Pecht [82] proposed to monitor the life consumption of an electronic system and project the system lifetime based on the monitoring results. We take a similar approach to monitoring the stresses on the circuit continuously, and we also intelligently adapt the circuit operation to maximize the circuit performance without reducing reliability.

7.3 Lifetime banking opportunities

Due to activity variations, the power consumptions of on-chip components (i.e. caches, FP/INT units, branch predictor, etc.) are not constant. Therefore, there exists not only chip-wise spatial temperature gradients but also temporal temperature gradients for each component.



(a)



(b)

Figure 7.1: Temporal temperature variation. (a) Single program workload. (b) Two-program workload with context switching. [64]

Figure 7.1 depicts the temperature profiles for two different workloads that are commonly seen in general purpose computing. Figure 7.1(a) represents a single program workload and Figure 7.1(b) represents a multi-program workload with context switching. In the single program workload, temperature changes over time due to the phased behavior in the executed program. In the multi-program workload, besides the execution variations within each program, inter-program thermal differences also affect the overall thermal behavior of the workload. For example, in Figure 7.1(b), the workload is composed of one cold program (*applu*) and one hot program (*gcc*). Thus the temperature fluctuation in Figure 7.1(b) is quite different with various context switching intervals. Though there are different thermal behaviors for different workloads, one can still find some common characteristics as compared with server workloads, which we will discuss in Section 7.6. In Figure 7.1, temperature variations occur in a manner with small granularity in both magnitude and time interval. More formally, the temperature profile can be decomposed into a constant temperature component (average temperature) and a high frequency component. The analysis in the previous chapter reveals that the constant temperature component in the temperature profile is approximately equal to the reliability equivalent temperature, as shown in Figure 7.1(a). It is the high frequency component that provides opportunities for lifetime banking. When the actual temperature is under the reliability equivalent temperature, the lifetime is consumed with a slower speed, which allows subsequent execution above the reliability equivalent temperature.

7.4 Dynamic Reliability Management Based on Lifetime Banking

In the previous chapter, we derived the lifetime model for electromigration subject to dynamic stresses (Equation (6.8) and (6.9)). Considering void growth limited failures such as those in dual-damascene Cu interconnects [75], let current exponent $n = 1$, we can rewrite MTF by combining Equation (6.8) and (6.9) as:

$$T_f \propto \frac{1}{E\left[j(t)\left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right)\right]}$$

Or equivalently, by eliminating the expected-value function, one can express the MTF in an integral form:

$$\int_0^{T_f} j(t)\left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right)dt = D \tag{7.1}$$

where D is a constant determined by the structure of the interconnect. Equation (7.1) models interconnect time to failure (i.e., interconnect lifetime) as a resource consumed by the system over time. Function $r(t) = \left[j(t) \left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)} \right) \right]$ can be regarded as the consumption rate. In DSM copper technology, void growth failure (e.g. at vias) is the major EM induced failure mechanism [29], and $r(t)$ can be regarded as the void growth rate (i.e. the atom drift rate at the cathode) in this case. Equation (7.1) provides a model to capture the effect of transient behaviors on system lifetime. One interesting case is when $j(t) = 0$, which occurs when the system is inactive as commonly seen in systems with non-server, user-driven workloads. When this happens, the atomic flux becomes zero while the effect of the back-flow diffusion near the cathode created by the EM atomic flux in active periods is worth careful examination. If the inactivity happens in the early stage of the void growth, the back-flow diffusion is negligible and the void simply stops growth during the power-down periods. If the back-flow diffusion is comparable to the normal EM atomic flux, which happens at a very long time after EM begins (e.g. at the order of several years). This back-flow diffusion tends to reduce the void size at the inactivity periods by refilling the void with atoms. However, in order to have significant impact on the void size already formed, this healing process has to last for a duration comparable to the time it took to grow to the current void size, e.g. several years. The inactivity period in normal usage is usually much less than this time scale. Therefore, the void size is essentially unaffected in inactivity. Our simulations confirm this observation and more detailed discussion on this aspect is out of the scope in this paper. In summary, the void size remains unchanged during the inactive period if the inactive period is much less than the total active time. Equation (7.1) accurately models this phenomenon by specifying $r(t) = 0$ during the inactive periods.

Ideally, we would like to monitor the temperature and current for each individual interconnect to build an exact full chip reliability model. In practice, only a limited number of temperature sensors are available on die, and a detailed and complex full chip reliability model is not suitable for runtime management due to the computation overhead. In this study, we use the maximum temperature measured across the chip at runtime, together with the worst-case current density specified at design time, to calculate the dynamic consumption rate. This is a conservative but safe approach. Thus, the results obtained in this study provide a lower bound for the potential benefits delivered by the proposed DRM method. When DVS is applied, the worst-case current density in the IC interconnects should be scaled according to the voltage/frequency setting used. The relationship between current density, supply voltage and clock frequency can be modeled by transferred charges per clock cycle [13]: $j \propto \frac{CV}{T} = CVf$, where C is the effective capacitance.

When a chip is designed, usually an expected lifetime (e.g., 10 years) is specified under some operating conditions (e.g., temperature, current density, etc.). We use $r_{nominal}$ to denote the lifetime consumption rate under the nominal conditions (e.g. reliability constrained temperature threshold). During runtime, we monitor the actual operating conditions regularly, calculate the actual lifetime consumption rate $r(t)$ at that time instance, and compare the actual rate with the nominal rate $r_{nominal}$ by calculating $\int (r_{nominal} - r(t))dt$, which we call the "lifetime banking deposit". When $r(t) < r_{nominal}$, the chip is consuming its lifetime slower than the nominal rate. Thus, the chip's lifetime deposit is increased. When $r(t) > r_{nominal}$, the chip is consuming its lifetime faster than the nominal, and the lifetime banking deposit will be reduced. According to Equation (7.1), as long as the lifetime deposit is positive, the expected lifetime will not be shorter than that under the nominal consumption rate $r_{nominal}$. Figure 7.2 illustrates this S_DRM technique. For example, in the interval $[t0, t1]$, the reliability of the chip is banked, while in $[t1, t2]$, the banking deposit is consumed. At time instance $t2$, the banking deposit becomes less than some threshold, and a cooling mechanism has to be engaged to quickly pull down the lifetime consumption rate to the nominal rate, just as is done in conventional DTM techniques. In other words, our S_DRM technique adopts DTM as a bottom-line guarding mechanism.

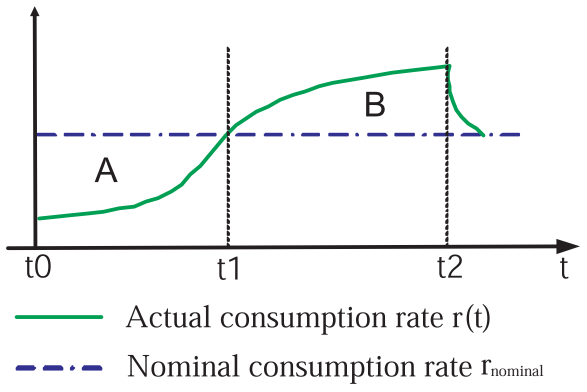


Figure 7.2: Simple dynamic reliability management (S_DRM). [64]

Therefore, the difference between conventional DTM and our S_DRM lies in the case where the chip's instantaneous consumption rate is larger than its nominal rate. In DTM, the lifetime consumption rate is never allowed to be larger than the nominal. In S_DRM, before we engage thermal management mechanisms we first check to see if the chip currently has a positive lifetime balance. If enough lifetime has been banked, the system can afford to run with a lifetime consumption rate larger than the nominal rate. Otherwise, we apply some DTM mechanism to lower the consumption rate, thus preventing a negative lifetime balance. In this study, we use dynamic voltage/frequency scaling as the major DTM mechanism. Since S_DRM only needs to monitor the actual lifetime consumption rate and to update the lifetime banking deposit, the computation overhead is negligible compared to that of DTM.

7.5 Experiments and analysis for general-purpose computing workloads

7.5.1 Experimental set-up

We run a set of programs from the Spec2000 benchmark suite on a processor simulator (SimpleScalar [18]) with the characteristics similar to a 0.13μm Alpha 21364. We simulate each program for a length of 5 billion instructions, and obtain both dynamic and static (leakage) power traces, which are fed as inputs to a chip-level compact thermal model *Hotspot* [94] for trace-driven simulation. In our trace-driven simulations, we include the idle penalty due to frequency/voltage switching, which is about 10us in many real systems [94]. Furthermore, since leakage power is strongly dependent on temperature, we scale the leakage power trace input dynamically according to the actual temperature obtained during runtime, using a voltage/temperature-aware leakage model [118]. Since the *Hotspot* model is highly parameterized, one can easily run experiments on a simulated processor with different thermal package settings. In order to obtain meaningful results, one should carefully choose the initial temperature setting for the *Hotspot* model. For each new thermal package setting, we obtain its initial temperatures by repeating the trace-driven simulations until the steady temperatures of the chip are converged, as suggested in [94].

We implement both DTM and S_DRM in the *Hotspot* model and set 110°C as the temperature threshold for both runtime management techniques. Both schemes use a feedback controlled dynamic voltage/frequency scaling mechanism to guard the program execution. For example, in DTM, when the actual temperature is above a certain temperature threshold, a controller is used to scale down the frequency/voltage, ensuring the program will never run at a temperature higher than 110°C. Our S_DRM scheme uses 110°C as the nominal temperature for the lifetime consumption rate. If the program never runs at a temperature less than that of the nominal (i.e., without banking opportunity), our S_DRM scheme will perform the same as thermal threshold-based DTM as the DTM policy is always engaged. On the other hand, if the program never exceeds the nominal temperature with full CPU speed, neither mechanism is engaged. Finally, we record the simulated execution times for fixed length power traces as the system performances under the two runtime management techniques, and use “performance slow-down” , defined as

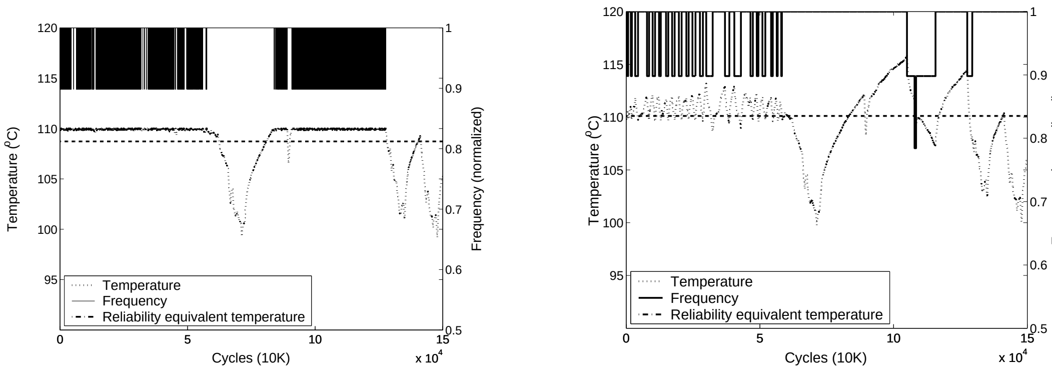
$$\frac{(\text{simulated time w/ runtime management} - \text{simulated time w/o runtime management})}{\text{simulated time w/o runtime management}}$$

as the metric to compare both techniques.

7.5.2 Single-program workload

Figure 7.3 plots the dynamic process for both conventional DTM and the proposed S_DRM techniques for benchmark *gcc*. The feedback controller in DTM effectively clamps the temperature within the target temperature (110^oC) by oscillating the clock frequency between 1.0 and 0.9, resulting in a reliability equivalent temperature less than that, and causing unnecessarily frequent clock throttling (Figure 7.3 (a)). On the other hand, our S_DRM technique can exploit reliability banking opportunities during the cool phase, and delay the engagement of throttling, while maintaining the specified reliability budget, as proved by the reliability equivalent temperature shown in Figure 7.3 (b).

Figure 7.4 shows the performance penalty for both DTM and S_DRM with the same thermal configuration. Only those benchmarks subject to performance penalties due to runtime management are shown here. As clearly indicated in the figure, performance penalty with the S_DRM scheme is always less than that with DTM scheme, when the thermal configuration is the same. On average, the S_DRM technique reduces the performance penalty by about 40% of that due to DTM (from 7% to 4%). Also shown in the figure is the performance of DTM with a more expensive thermal package whose convection thermal resistance is only one third of the other's. As one can expect, a more expensive thermal package can reduce the performance penalty. Figure 7.4 shows that, on average, S_DRM with a higher thermal resistance can achieve a performance very close to that of DTM with a lower thermal resistance. These results imply that, if the tolerable performance lost is fixed, the application of S_DRM allows the usage of a much cheaper thermal package than that required by the conventional DTM technique.



(a) (b)

Figure 7.3: Temperature and clock frequency profiles in different thermal management techniques for benchmark *gcc*. (a) Conventional DTM (threshold temperature =110^oC). (b) Reliability banking based DRM (reliability target temperature =110^oC).

In addition, using the S_DRM technique, one can explicitly trade-off reliability with performance by targeting different lifetime budgets. That is one can increase the nominal lifetime consumption rate when lifetime target is allowed to be reduced. Figure 7.5 plots the performance of S_DRM averaging over all benchmarks at different lifetime budgets, with shorter expected lifetimes enabling faster execution. However, reducing lifetime by 10% only increases the performance by about 1%.

When compared with the conventional thermal threshold-based DTM, a distinct feature

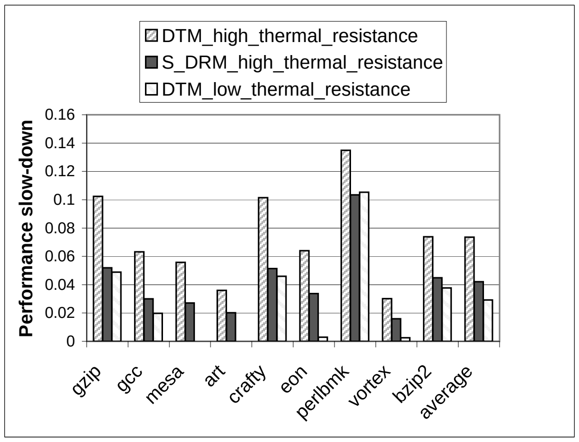


Figure 7.4: Performance comparison of DTM and the proposed S_DRM. The results for S_DRM are based on high convection thermal resistance configuration. The results for DTM include two different thermal configurations. [64]

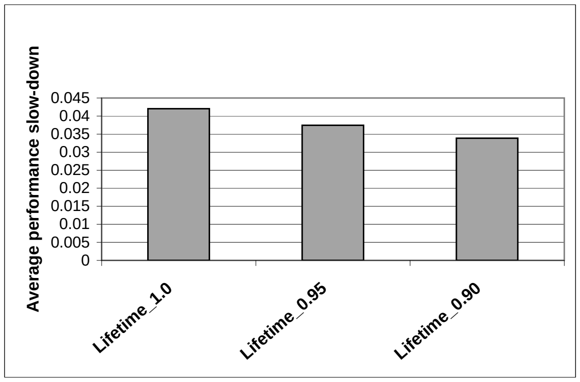


Figure 7.5: S_DRM performance at different targeted lifetimes. [64]

of S_DRM is its ability to "remember" the effects of previous behaviors. If the lifetime balance is high due to previous deposits, S_DRM will be more tolerant of higher operating temperatures for longer time intervals, thus reducing the performance penalties due to conventional DTM slow-down mechanisms. In summary, the advantage of S_DRM over DTM is largely dependent on the inherent variations in the temperature profile of the workload.

7.5.3 Multi-program workload

Another interesting program execution scenario is a workload of multiple programs with context-switching between them. When a hot benchmark and a cold benchmark are executed together, the average operating temperature should be between the individual benchmarks' operating temperatures. For example, *gcc*'s own operating temperature is around 115^{o}C and *applu*'s is around 70^{o}C. Figure 7.1(b) plots the temperature profile of a hybrid workload composed of *gcc* and *applu*, with different context-switch time intervals. Note that, in our simulation, the multi-program workload is constructed using the power trace of the individual program, and the overhead of context-switching is not modeled and simulated. Since we are only interested in the relative performance of different runtime management technique, such simplification should not affect the final conclusions.

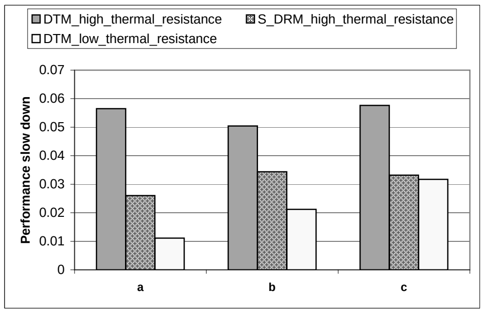


Figure 7.6: Average performance comparison of DTM and DRM on a multi-program workload with different context-switch intervals ((a) 50μs, (b) 5ms, and (c) 25ms). [64]

As one expects, the smaller the context-switch interval, the less temperature fluctuation, with the thermal package of the chip working as if a low-pass filter. When the context-switch interval is increased, individual benchmarks can show their hot/cold properties, and the temperature variation in the workload becomes obvious. In order to investigate how multi-program workloads affect the performance of DTM and DRM, we reduced the temperature threshold of the targeted lifetime from 110°C to 90°C. Figure 7.6 shows the performance penalties of DTM and S_DRM for this multi-program workload with different context-switch intervals. We observe a similar trend shown in the single-program workload. S_DRM outperforms DTM with the same thermal package configurations. As the context-switch interval increases, the performance of S_DRM becomes closer to that of DTM with a much smaller convection thermal resistance (three-fold smaller).

7.6 Dynamic Reliability Management for Server Workloads

We have investigated the application of banking based DRM in workloads for general-purpose computing. As we will see, the disadvantages of this technique become more obvious in server systems such as web servers, in which hot phases usually imply an increased number of service requests while the engagement of active cooling mechanisms then exacerbate the QoS provided by the server. In the following, we first discuss some distinct characteristics of server workloads in terms of both thermal behaviors and performance requirements. We then propose a profile-based dynamic reliability management (P_DRM) technique that can extract more benefits from lifetime banking for those server workloads.

7.6.1 Characteristics of server workloads

In general-purpose computing, the temperature variations of workloads are largely due to the inherent phased behaviors (i.e. phased activities or context-switches). These variations usually occur in a very small-scale time interval, which is comparable to the thermal constant of chip thermal package. Server workloads, in contrast, are dependent on user requests, which vary with a much larger time scale with tens of hours [15, 32]. There are several distinct characteristics in the server workloads like those presented in [15, 32].

First, there is clearly a cool phase (lower request rate) and a hot phase (higher request rate) in the workload distribution. For example, in a workload trace for the 1998 Winter Olympic Games, the request rate increases from around 50 (req./s) in the cool phase to above 400 (req./s) in the hot phase, an eight-fold difference. Second, as a consequence of the workload and associated processor utilization variation, the power consumption of the processor varies greatly (a two-fold difference in the Olympic Games servers), which implies a large variation in temperature. Third, each phase sustains for a very long time interval. Thus, each phase reaches its steady-state temperature and stays at that temperature for most time of the phase interval. This is quite different from general-purpose computing, where the interval for each thermal phase is very short and the steady-state temperature is seldom reached before the next phase arrives. These distinct thermal characteristics make our lifetime-banking-based reliability management promising for server workloads.

In the hot phase, conventional thermal threshold-based DTM clamps the maximum temperature to a predefined threshold by slowing down the processor, thus possibly exacerbating the situation. In contrast, banking-based runtime management can exploit the banking effects of the long cool phase and delay or reduce the performance loss due to engagement of a cooling mechanism. From an average user's point of view, the QoS provided by the server is largely dependent on its performance in the hot phase, as most requests are made during that time. Therefore, in our study, *we use the performance of the hot phase as our performance metric for comparison*.

7.6.2 Dynamic reliability management for server workloads

In order to evaluate our runtime management technique on server workloads, we construct a hybrid workload in a way similar to that of the multi-program workload, but with a much longer context-switch interval. This synthetic workload is composed of a cool phase and a hot phase, running Spec2000 benchmarks *applu* and *gcc* respectively. Figure 7.7 shows the temperature profile of the synthetic workload we use to mimic the thermal behavior of server workloads. From various experiments, we find that the thermal time constants of our simulated system are in the range of tens of milliseconds. Therefore, by simulating workloads in a time scale of several seconds, we can ensure that the portion of time in the profile spent on the transient behaviors from one phase to another is minimized, just like one may see in a temperature profile for server workloads. Although the total simulated time is short (i.e., about one second) compared to a real server workload, Figure 7.7 indicates that the time interval for each phase is long enough to reach the steady-state operating temperature of the individual program. The temperature variations within each program also mimic the workload variations in both the cool and hot phases of a real server workload. Therefore, the time units shown in Figure 7.7 could be interpreted as scaled down from a much longer time interval (e.g. several hours). One disadvantage of our synthetic workload is that power peaks due to individual requests in the cool phase are not modeled. However, the effect of those intermittent power peaks on reliability banking is not significant, because of the filtering effect of the thermal package on temperature.

In our synthetic workloads, the cool phase is followed by the hot phase, and lifetime will be banked first and then withdrawn. In other workloads where the hot phase is followed by the cool phase, DTM can be applied in the hot phase if there is no previous lifetime banking, and lifetime will be banked in the following cool phase and prepared for withdrawal in the future hot phase. Thus, our lifetime banking based approach is effective in spite of the detail of workloads (i.e. the order of cool and hot phases). We define the *duty cycle* of the cool phase as the portion of time the cool phase occupies in the whole length of simulation. For example, in Figure 7.7, the duty cycle of the cool phase is equal to 0.5. In our experiments, we also construct workloads with different duty cycles of the cool phase (e.g., 0.6 and 0.75), and in all of these workloads, individual programs reach their own steady-state temperatures.

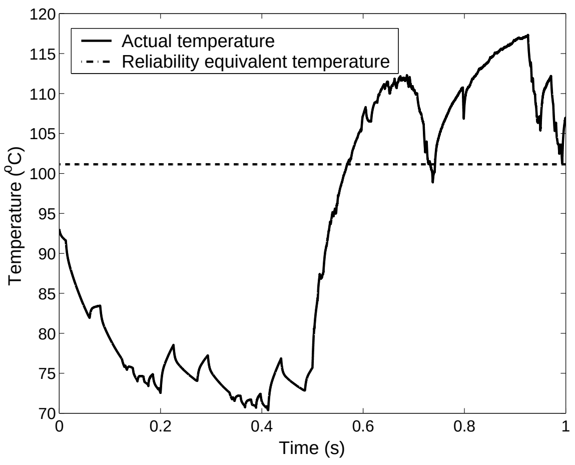


Figure 7.7: A constructed workload used to mimic the thermal behavior of server workloads. [64]

The application of the S_DRM technique to server workloads is straightforward, just like in the context-switched multi-program workload studied previously. However, our simulation results reveal that S_DRM is not the best choice for server workloads. In the workloads of general-purpose computing, since each phase is very short, the lifetime balance deposited in the previous cool phases can support the subsequent over-consumption of lifetime for an interval comparable to that of the hot phase. S_DRM can minimize the impacts on the phase within these workloads. However, in the server workloads, the interval of the hot phase is much longer, and temperature rises steadily towards the hot phase steady-state temperature. At the same time, due to the exponential dependence of lifetime consumption rate on temperature, the lifetime balance is consumed more and more rapidly, despite a previous long cool phase. Figure 7.8 demonstrates such a process in the time interval $[0.6s, 0.68s]$. After 0.68s, the lifetime balance becomes *zero*. S_DRM performs during the rest of the hot phase just as it behaves in the single program workload. Therefore, only a small portion of the execution in the hot phase benefits from the lifetime banking by the cool phase.

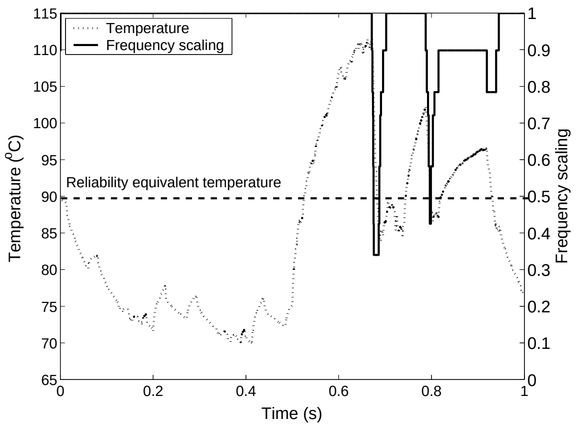


Figure 7.8: S_DRM (simple dynamic reliability management) on the synthetic workload shown in Figure 7.7.

Due to the above reason, one should find a more strategic way to spend the lifetime balance in order to maximize the performance in the hot phase. Since in steady state, temperature can be modeled as a function of the operating frequency, one can find the relationship between lifetime consumption rate and operating frequency. Let $f(t)$ denotes the operating frequency curve in the hot phase, and $r(f(t))$ be the corresponding lifetime consumption rate. The problem to find the maximum performance operating scheduling while satisfying the reliability constraint can be formulated as a constrained optimization problem as follows.

$$Max(E[f(t)]), \text{subject to} E[r(f(t)] = R, \; t \in \text{hot phase}$$

where $E[]$ is the expected-value function, and R is a constant in the hot phase that is determined by the lifetime balance deposited during the cool phase as well as the nominal lifetime consumption rate. We assume that, in the hot phase, system performance is proportional to the clock speed.

Figure 7.9 plots clock frequency as a function of the lifetime consumption rate. It is obvious that the relationship between clock speed and lifetime consumption rate forms a

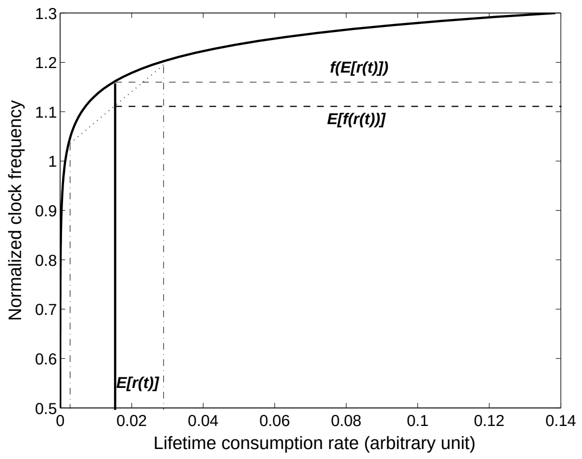


Figure 7.9: Relationship between clock frequency and lifetime consumption rate.

convex curve. According to Jensen's inequality, it follows that (as shown in Figure 7.9) $f(E[r(t)]) \geq E[f(r(t))]$, which implies that, in order to obtain the best performance, one should operate with a constant consumption rate. In other words, one should distribute the lifetime balance evenly across the hot phase. In order to calculate the desired consumption rate in the hot phase, one has to know the duration of the hot phase. Currently we assume that this information can be obtained through profiling technique thanks to the high regularity of the workload distribution for servers.

With the optimal operating condition in mind, we introduce our (P_DRM, profile-based dynamic reliability management) technique, which is a natural extension of our S_DRM with the awareness of the optimal operating points in the hot phase. When the server is running in the cool phase, P_DRM works the same way as S_DRM with lifetime balance banked. When the server enters the hot phase, P_DRM calculates a new nominal lifetime consumption rate based on the lifetime balance and the duration of the hot phase (obtained through profiling). Then P_DRM acts just like S_DRM, with the new calculated nominal consumption rate, which can further exploit some banking opportunities due to temperature variations within the hot phase.

The profiling only provides a *prediction* that allows the CPU to jump to the best operating point during a hot phase. In some cases we might not be able to obtain accurate workload profiles. However, with our P_DRM technique, the inaccuracy of workload profiles only affects the the performance optimality, and does not result in violations to the lifetime budget. That is because our technique always tracks the actual reliability consumption rate and compares it with the nominal lifetime consumption.

7.6.3 Simulation results for server workloads

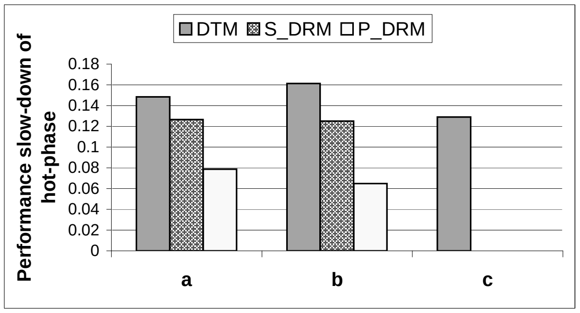


Figure 7.10: Performance comparison of different runtime management techniques on the synthetic workload shown in Figure 7.7 with different duty cycles of the cool phase: (a) 0.5, (b) 0.6 and (c) 0.75. [64]

We simulate the synthetic workload shown in Figure 7.7, which mimics the thermal behaviors of the real server workload, with different runtime management techniques. We change the program switching time so that we can test on 3 workloads with different duty cycles of the cool phase. We compare the performance slow-down in the hot phase and the results are presented in Figure 7.10. Both DRM techniques outperform DTM, and P_DRM performs the best. The performance of S_DRM is slightly better than that of DTM and much worse than P_DRM due to the reasons discussed in above. On the other hand, P_DRM can fully exploit the banking benefits of the cool phase. For example, when the cool phase occupies 60% of the total time (i.e. as indicated by (b) in Figure 7.10), P_DRM can reduce the performance penalty from 16%(DTM) to only 6% (or equivalently, the execution speed of the hot phase is increased by P_DRM by about 9.5% over DTM). Interestingly, for the case when the cool phase occupies 75% of the total time (i.e., (c) in Figure 7.10), no performance slow-down is incurred for both DRM techniques, because the reliability equivalent temperature for that workload is less than the reliability nominal temperature. Thus, in that case, the lifetime balance banked in the cool phase is enough to support the full speed execution in the hot phase, while DTM clamps the hot phase temperature to the reliability temperature, resulting in about a 13% performance penalty in the hot phase.

7.6.4 An analytical model for P_DRM for server workloads

In order to fully understand the potential benefits of p_DRM on server workloads, we present a first order analytical model, providing some insights of our proposed runtime techniques. In this model, we approximate server workloads using square waveforms as shown in Figure 7.11. The solid blue line represents the temperature/performance profile with DTM. The temperature profile with P_DRM in the cool phase overlaps with that of DTM. And P_DRM allows operating points above the reliability temperature in the hot phase, as presented by the dotted green line in the figure. We want to find out what is the allowable performance difference between the dotted green line and the solid blue line (i.e., the performance gain of P_DRM over DTM), subject to a fixed lifetime budget. Here we make an assumption that the processor can operate at a clock frequency higher than that clamped by the thermal threshold. There are two aspects to this. First, temperature excursions will require a reduction in frequency, thus reducing performance somewhat, but should still outperform a strict, temperature-limited form of DTM because the temperature dependence of frequency is mild [94]. Second, many ICs are actually under-clocked due to thermal limitations. In both cases, there exist possibilities that we can over-drive the processor in the hot phase to meet the QoS requirements without sacrificing reliability lifetime.

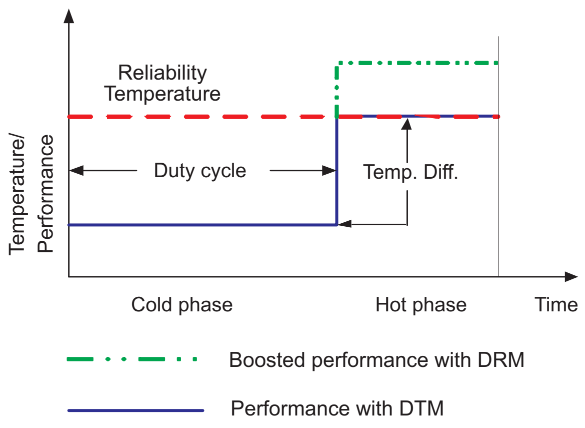


Figure 7.11: Modeling thermal behaviors of server workloads using square waveforms. [64]

As one can see from Figure 7.11, two factors might affect the potential performance boost by P_DRM over DTM: 1. the difference between the steady-state temperatures in both the hot phase and the cool phase, and 2. the duty cycle of the cool phase. We set the reliability temperature $T_n = 105^oC$, associated with the clock frequency $f_n = 3.0GHz$. This setting means that in the hot phase, the maximum performance achieved by DTM is to operate at $3.0GHz$. If we can assume that the dynamic power consumption of the processor is proportional to the cubic of clock frequency, the steady state temperature can be denoted by $T(f) = K_f f^3 + T_0$, where K_f is a constant and T_0 represents the ambient temperature of the thermal package. Accounting for the contribution of static power consumption to temperature, we set a higher ambient temperature $T_0 = 55^oC$, and obtain $K_f = 1.85K/GHz^3$. Let ΔT denote the temperature difference between the hot phase and the cool phase, f_2 the allowable operating clock frequency in the hot phase by P_DRM, and α the duty cycle of the cool phase. The following equation should be satisfied to retain the same lifetime budget with P_DRM:

$$[r_n(T_n) - r_1(T_n - \Delta T)]\alpha = [r_2(f_2, T(f_2)) - r_n(T_n)](1 - \alpha) \tag{7.2}$$

where r_n is the nominal reliability consumption rate at temperature T_n, r_1 is the consumption rate in the cool phase, and r_2 is the consumption rate in the hot phase with clock frequency f_2 and temperature $T(f_2)$. The left hand side of the above equation represents the reliability balance banked during the cool phase and the right hand side represents the banking deposits to be consumed in the hot phase. Although the temperature dependence of static power is not taken into account in this model, we feel that it captures the key relationships between performance, operating temperature and reliability consumption rate, and is thus sufficient for our purposes.

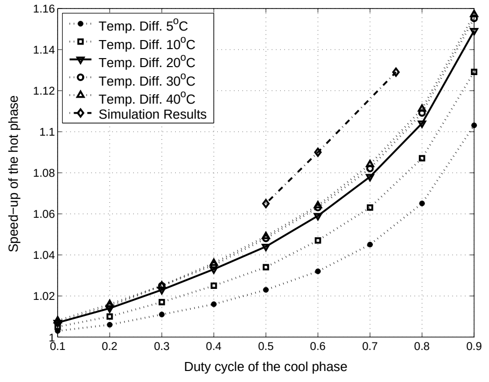


Figure 7.12: Performance speed-up due to lifetime banking on different workload characteristics. [64]

Using the above analytical model (i.e. Equation (7.2), we can calculate the performance speed-up by P_DRM (i.e. $\frac{f_2}{f_n}$ in the hot phase) as a function of ΔT and the duty cycle of the cool phase. The results are presented in Figure 7.12, which shows that the performance speed-up is highly dependent on the duty cycle of the cool phase. When the duty cycle of the cool phase is fixed, the increase of temperature difference will also increase the speed-up. However, after some value (e.g. about 20^oC), the temperature difference has a minor effect on the speed-up, due to the exponential dependence of the reliability consumption rate on temperature. Because the extra reliability balance brought by further lowering the temperature in the cool phase is negligible when compared to the very high consumption rate in the hot phase. This figure suggests that the “sweet spot” for performance speed-up with P_DRM lies in the case when the duty cycle of the cool phase is more than 50% and the temperature difference is more than 20^oC, and we can expect more than 5% of performance speed-up. Fortunately, as shown before, many server workloads satisfy these requirements.

The simulation results of DTM and P_DRM from Figure 7.7 are re-plotted in Figure 7.12. The workloads for these data are similar to that shown in Figure 7.7, with the cool phase duty cycle equal to 0.5, 0.6 and 0.75 respectively. The reliability temperature is set to 90^oC, while the temperature of the cool phase in these workloads is about 70^oC. These simulation results show a similar trend to that predicted by our simple analytical model, though our analytical model is not calibrated against any specific simulation data. Therefore, these simulation results confirm the applicability of our analytical model. Compared with the simulation results, it seems that the analytical model underestimates the performance speed-up by P_DRM. Two major reasons might help explain the discrepancy. First, in our analytical model, we use a cubic relationship between power and operating frequency, which exaggerates the effect of clock frequency on the temperature, leading to a more conservative estimate of the performance speed-up. Second, in the simulations, we include the idle penalties for frequency/voltage transitions due to dynamic frequency/voltage scaling, while in the analytical model, we do not assume any extra performance penalty for DTM.

7.7 Summary

In this chapter, we detailed the use of the temperature variability and lifetime resource models to develop novel DRM techniques that reduce the performance penalties associated with existing DTM techniques while maintaining the required expected IC reliability lifetime. When the operating temperature is below a nominal temperature (i.e., the threshold temperature used in DTM techniques), lifetime is being consumed at a slower than nominal rate, effectively banking lifetime for future consumption. A positive lifetime balance allows the nominal temperature to be exceeded for some time (thus consuming lifetime at a faster than nominal rate) instead of automatically engaging DTM and unnecessarily suffering the associated performance penalties.

For general-purpose computing, simulation results revealed that S_DRM provides performance improvements over traditional threshold-based DTM without sacrificing expected lifetime, or allows the usage of cheaper thermal package without sacrificing performance. For server workloads, simulations on synthetic workloads demonstrate the possibility to increase server QoS by using P_DRM when service requests are aggregated. A conservative analytical model further identifies the "sweet spots" of server workloads that benefit from our P_DRM. Although the DRM experiments presented in this paper do not explicitly study the scenarios with long periods of inactivity, which are commonly seen in non-server, user-driven workloads, our lifetime banking techniques can be applied in a straightforward way, because our dynamic reliability model (Equation (7.1)) also holds true in these situations. Consequently, much better performance gains would be expected compared to those obtained in the server style workloads presented here, because more lifetime banking opportunities are available during those inactivity periods.

Chapter 8

Conclusion and Future Directions

With the advance of technology scaling, power and thermal issues have been among the limiting factors facing IC design, due to both the emergence of mobile devices and increasing complexity and clock frequency on a chip. Higher temperatures not only degrade system performance, raise packaging costs, and increase leakage power, but they also reduce system reliability via temperature enhanced failure mechanisms. In order to ensure the system operates normally in all corner cases, designers have to use worst-case assumptions when designing a complex system. Due to increasing PVT (process, voltage, temperature) variability in systems on future technologies, worst-case assumptions result in excessive design margins by imposing extreme design constraints, which leads to cost-inefficient designs. In order to solve the problem, post-design runtime management techniques are needed for future systems. The advantages of adaptive runtime management are two-fold. On one hand, runtime management can adapt the system to the workload variations, reaching higher efficiency by reclamation of design margins. On the other hand, by monitoring the operating conditions continuously, runtime management can avoid those extreme cases which rarely happen, thus relieving the the worst-case design constraints.

8.1 Conclusion

In this dissertation, we study power and thermal impacts of workload variations in various applications, and investigate runtime management techniques to exploit these variations to reduce the increasing power and temperature constraints. In the area of power-aware design, we apply efficient low power techniques in runtime by exploiting workload execution time variations. In order to attenuate the thermal constraints, We explore the effect of temperature variations on circuit reliability, develop a reliability model under dynamic thermal stress, and investigate architectural techniques to maximize circuit performance without violating IC lifetime. We argue that by combining our techniques with existing design time based optimization techniques, we should be able to continuously enjoy the advantages brought by technology scaling. Specifically, we obtain the following results.

- An efficient method to find the optimal intra-task procrastinating voltage/frequency scaling for single tasks in practical real-time systems using statistical workload information [66]. Our method is analytic in nature and proved to be optimal. Simulation results verify our theoretical analysis and show significant energy savings over previous methods. In addition, in contrast to the previous techniques in which all available frequencies are used in a schedule, we find that, by carefully selecting a subset of a small number of frequencies, one can still design a reasonably good schedule while avoiding unnecessary transition overheads.

- A formal feedback-control algorithm for dynamic voltage/frequency scaling (DVS) in a portable multimedia system to save power while maintaining a desired end-to-end decoding delay [59,60]. Our algorithm is similar in complexity to the previously-proposed change-point detection algorithm [91] but does a better job of maintaining stable throughput and is not dependent on the assumption of an exponential distribution of the frame decoding rate. For approximately the same energy savings as reported by [91], our controller is able to keep the average frame delay within 10% of the target more than 90% of the time, whereas the change-point detection algorithm kept the average frame delay with 10% of the target only 70% or less of the time executing the same workload.

- Even though end-to-end delay is an important QoS metric for mobile multimedia players which receive incoming streaming data through the network, playback deadline missing rate is a more practical metric for high quality multimedia playback. By regarding a multimedia playback system as a soft real-time system, we model the decoding process as a discrete-time system excited by random input sequences representing the incoming stream. Using this novel stochastic process model, we apply formal methods to analyze the decoding system and design a feedback-based on-line dynamic voltage/frequency scaling (DVS) algorithm to effectively reduce the energy consumption during multimedia playback. We implemented our technique in a laptop computer equipped with a DVS enabled processor, and results reveal the proposed algorithm is indeed a good trade-off among energy, playback quality, hardware cost and playback latency [65].

- Thermal effects and their induced reliability issues are becoming a limiting factor for performance, even if aggressive low power design techniques are available. We present a physics-based dynamic electromigration model for estimating interconnect lifetime subject to temporal and spatial thermal gradients. This model returns reliability equivalent temperature and current density that can be used in traditional reliability analysis tools. Our models reveal that blindly using the maximum or average temperature to evaluate EM lifetime is inappropriate. Our results not only increase the accuracy of reliability estimates but enable designers to more aggressively explore the design space and to reclaim the design margin imposed by less accurate, more pessimistic models [61].

- Our dynamic reliability model reveals that reliability can be modeled as a resource to be consumed at a temperature dependent rate. This novel view enables the use of the temperature variability and lifetime resource models to develop banking based dynamic reliability management (DRM) techniques that reduce the performance penalties. When the operating temperature is below a nominal temperature (i.e., the threshold temperature used in DTM techniques), lifetime is being consumed at a slower than nominal rate, effectively banking lifetime for future consumption. A positive lifetime balance allows the nominal temperature to be exceeded for some time (thus consuming lifetime at a faster than nominal rate) instead of automatically engaging DTM and unnecessarily suffering the associated performance penalties. We develop DRM techniques for both general-purpose and server workloads. Simulation results reveal that these techniques provide performance (QoS) improvements over traditional threshold-based DTM without sacrificing expected lifetime, or allows the usage of cheaper thermal package without sacrificing performance [63, 64].

8.2 Future Directions

Although uni-processor system is used as the test vehicle throughout this dissertation, the presented techniques can be readily applied to both distributed (multiple) processor systems and other off-chip components such as memory and I/O devices. There are several interesting direction in which the work described in this dissertation can be extended.

Procrastinating voltage scaling for multi-task real-time systems. It has been shown that procrastinating voltage scaling for multi-task real-time systems is superior to other existing techniques in energy saving [117]. However, in that work, ideal voltage/frequency scaling (i.e. continuous scaling) is assumed. The techniques presented in Chapter 3 can be applied in this scenario to derive practical procrastinating voltage scaling for multi-task real-time systems.

Improvement in the dynamic interconnect reliability model. One limitation in the application of our dynamic interconnect EM model is that our analysis is currently based on two-terminal interconnects, such as those seen in global signal interconnects and power/ground distribution networks. Recently Alam *et al.* [8] proposed lifetime predictions for multi-terminal interconnects. It would be useful to extend our current model to account for multi-terminal interconnects.

Improvement in chip level reliability model. Recent research on material and device reliability has shown that temperature- and voltage-dependent dynamic processes also govern other failure mechanisms, such as stress-migration [97], negative-bias temperature instability (NBTI) [116] and gate oxide breakdown [111]. It is possible to derive consumption-rate-based dynamic reliability models for these failure mechanisms just as for electromigration in Chapter 7. A simple yet conservative way to incorporate multiple failure mechanisms in the reliability-banking framework is to apply lifetime banking for each individual failure mechanism and obtain each mechanism's allowable operating point with the techniques presented in this article. We would then choose the safest one of the allowable operating points (that is, the lowest operating frequency) for circuit operation. Thus, no reliability budget violation will occur for individual failure mechanisms; however, performance gain will be minimal. A more complicated approach could trade off reliability budgets among different failure mechanisms without compromising system reliability.

The banking techniques presented in Chapter 5 focus on the worst-case component (the hottest interconnect metal in the chip). Because our techniques guarantee that the worst-case component will satisfy its reliability constraint, we can safely claim that they don't violate system reliability. Nevertheless, this approach is conservative because temperature distribution and current density are uneven across the chip, and we can model the chip as a system consisting of serial and parallel components. A more sophisticated approach would trade off reliability among different components without violating system reliability. Such an approach would require a complex reliability model such as the one presented by Srinivasan *et al* [98].

Power and thermal management in multi-core and distributed systems. Due to severe power/thermal problem, it becomes inevitable as shown in industry trend to shift from the pursuit of higher clock frequency to chip multi-processing (CMP). In CMP architectures, each core (i.e. processing element) might not produce much heat individually. But the congregate thermal effect still imposes a great challenge for thermal management. Each core needs to operate at a different clock frequency to alleviate the total thermal effect without sacrifice of performance. It would be an interesting optimization problem to co-operate multiple cores in a power- and temperature-aware fashion. In the other hand, Power and temperature in large scale computing infrastructures such as Internet data centers have become the key concerns. As CMP multi-core systems can be thought as miniatures of such large scale distributed systems, similar optimization problems are to be solved in these large scale systems [12].

As much of the existing work on power- and temperature-aware design focuses on the processor unit, the physical effects of peripheral components such as memory, disks and I/Os become more prominent. In order to achieve system-wide optimizations, dynamic techniques to coordinately manage individual system level component are urgently needed.

Appendix A

Optimal Speed Schedule for Multi-task with Memory Stalls

Assuming that there are two tasks which must be finished before deadline D, the total memory stall time when these two tasks are executed is denoted by Dr_m. In other words, a fraction of r_m of the total task execution time is devoted to memory access. Let $Dr_m r_1$ denote the memory stall time of task 1, the memory stall time of task 2 can be expressed as $Dr_m(1-r_1)$. The ratios of CPU computation time (at full speed) to the memory stall time in the tasks are p_1 and p_2 for task 1 and task 2 respectively. Task 1 is executed with CPU speed f_1 and task 2 with f_2. Therefore the execution time for task 1 can be expressed as $\frac{Dr_m r_1 p_1}{f_1} + Dr_m r_1$ and similarly $\frac{Dr_m(1-r_1)p_2}{f_2} + Dr_m(1-r_1)$ for task 2. The relation of these parameters can be illustrated in Figure A.1.

Due to real-time constraint, it must be satisfied that

$$\frac{Dr_m r_1 p_1}{f_1} + \frac{Dr_m(1-r_1)p_2}{f_2} = D(1-r_m) \tag{A.1}$$

On the other hand, the power consumption for CPU computation and memory access can

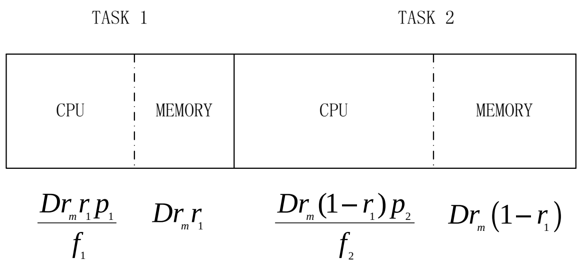


Figure A.1: Time distribution among task execution. Note that, in general, the memory stall time is interleaved with CPU computation time during task execution.

be modeled as cubic functions of CPU clock speed:

$$P = af^3 \quad \text{,for CPU computation}$$
$$P = bf^3 \quad \text{,for memory stall}$$

Therefore, the total energy consumed by the two tasks can be calculated as:

$$E(f_1, f_2) = \left[af_1^2 Dr_m r_1 p_1 + bf_1^3 Dr_m r_1\right] + \left[af_2^2 Dr_m(1-r_1)p_2 + bf_2^3 Dr_m(1-r_1)\right] \quad \text{(A.2)}$$

Given the timing constraint Equation (A.1), we would like to find the optimal execution speed f_1 and f_2 such that total energy consumption is minimized. This can be solved using Lagrange multiplier method and it turns out that the optimal execution speeds satisfy:

$$[2+3\frac{b}{a}\frac{f_2}{p_2}]f_2^3 = [2+3\frac{b}{a}\frac{f_1}{p_1}]f_1^3 \quad \text{(A.3)}$$

Thus, the energy-optimal speed schedule can be found by solving both Equation (A.1) and (A.3).

Notice that when $p_1 = p_2$, $f_1 = f_2$. A uniform speed to execute the two tasks yields minimal energy consumption in this case, which is the same as the case when the two tasks are purely CPU-bounded.

When $p_1 \neq p_2$, we are interested to investigate how the energy consumption using uniform speed deviates from the optimal solution. Let $f_{uniform}$ denotes the uniform speed to execute both tasks. The real-time constraint requires that

$$\frac{Dr_m r_1 p_1 + Dr_m(1-r_1)p_2}{f_{uniform}} = D(1-r_m) \tag{A.4}$$

The energy consumption using a uniform speed is

$$E(f_{uniform}) = af_{uniform}^3 D(1-r_m) + bf_{uniform}^3 Dr_m$$

Finally, it can be shown that the ratio of optimal energy consumption to that of using uniform speed can be expressed as:

$$\frac{E(f_1,f_2)}{E(f_{uniform})} = \frac{\frac{a}{b}\left[\frac{f_2^3(f_{uniform}-f_1)+f_1^3(f_2-f_{uniform})}{f_2-f_1}\right](1-r_m) + f_1^3 r_m r_1 + f_2^3 r_m(1-r_1)}{\frac{a}{b}f_{uniform}^3(1-r_m) + f_{uniform}^3 r_m}$$

where f_1, f_2 and $f_{uniform}$ can be solved by Equation (A.1), (A.3) and (A.4). Therefore, $\frac{E_{optimal}}{E_{uniformspeed}}$ is a function of r_m, r_1, p_1, p_2 and $\frac{a}{b}$, and independent of D.

Figure A.2 compares the optimal energy consumption with that using uniform speed for both tasks, with different values of r_m, r_1, p_1 while assuming $a = b$. Several interesting observations can be made from this figure.

1. When r_m is relatively small (e.g. $r_m < 0.1$), the energy difference is within 5% in spite of the values of r_1, p_1 and p_2, as indicated by the plots on the top half in Figure A.2. Recalling that r_m represents the portion of the total allowable execution time occupied by total memory stalls (sum of two tasks) during task execution, this suggests that, when the total memory stall time is small compared to the available execution time, using uniform speed schedule is still desirable.

2. In spite of the value of r_m, when p_1 and p_2 are both small or large, the energy con-

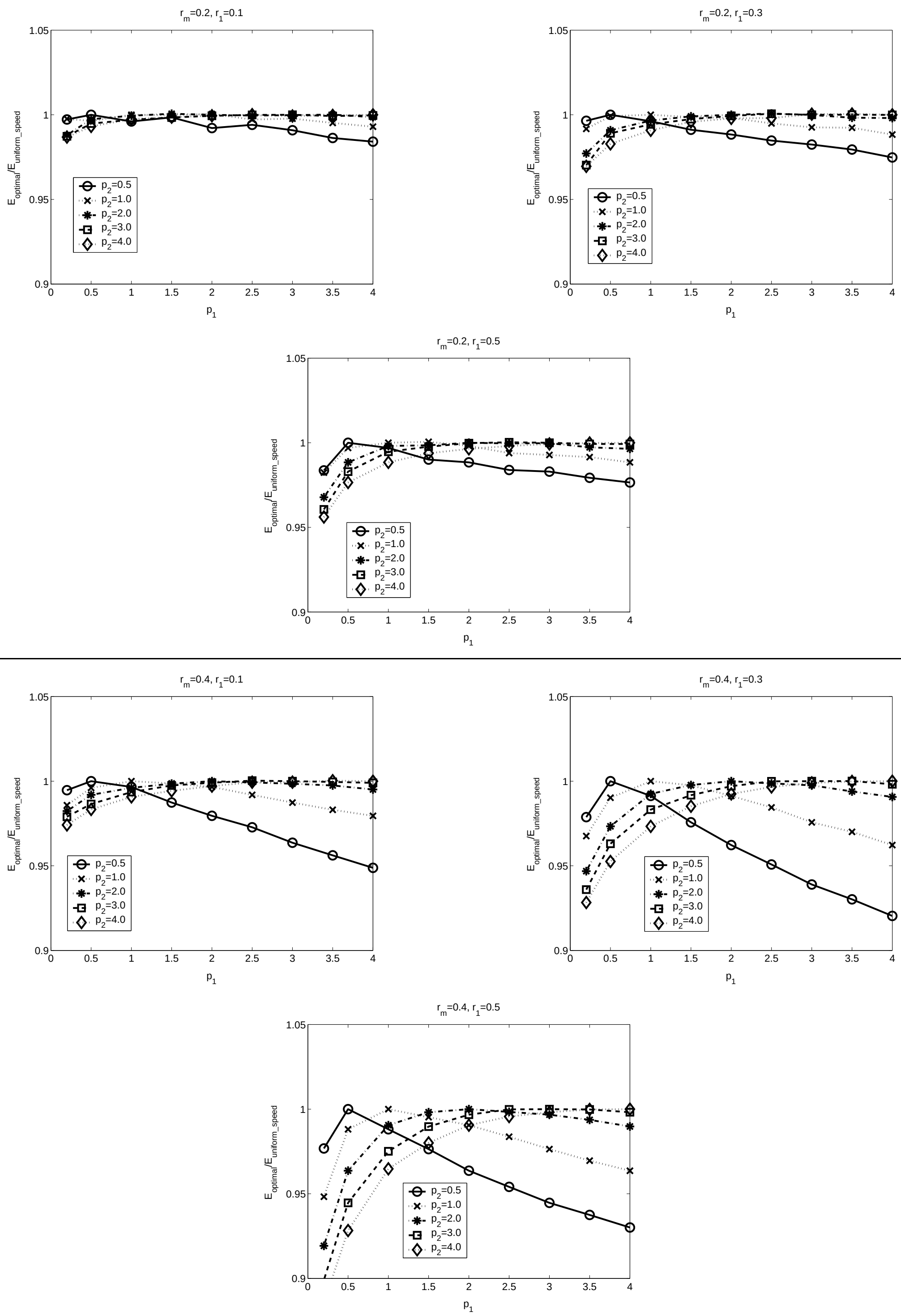


Figure A.2: Comparison of optimal energy with that using uniform speed at different values of r_m, r_1, p_1 and p_2 ($a = b$).

sumption using uniform speed schedule is very close to that of the optimal. Recalling that p_i ($i = \{1,2\}$) represents the ratio of CPU computation time to the memory stall time when task i is executed at full speed, larger p_i implies the task is more CPU-bounded, while smaller implies more memory-bounded. The observation suggests that when the two tasks have similar performance bottlenecks, using uniform speed schedule is still one of best solutions.

3. Only when r_m is relatively large (e.g. $r_m > 0.1$) and the two tasks have quite distinct performance bottlenecks (e.g. $p_1 = 0.2$, $p_2 = 4.0$ for memory-bounded vs. CPU-bounded), is uniform speed schedule inferior to the optimal schedule.

The results presented in Figure A.2 assume that $a = b$. In real systems, one might expect that $a > b$, since CPU operations usually consume more power than memory operations. One can verify that this condition actually increases the suitable design space for uniform speed schedule.

With the above observations at hand, we can evaluate whether uniform speed execution is still the best choice for real MPEG playback applications with the timing characteristics similar to those shown in Table 5.1. There are two cases:

- Case I: There are two distinct tasks for each frame during movie playback: decode task and display task. As revealed from Table 5.1 and Figure 5.3(b), decode task is more CPU-bounded while display task is strongly memory-bounded. According to Observation 3, it is better to execute these two tasks at different speed. That is exactly what we did in our experiments–we display frames using a fixed low CPU speed.

- Case II: Decode multiple frames using one single speed whenever possible. Table 5.1 shows that the decoding task is CPU-bounded disregard the frame type. Therefore, according to Observation 2, decoding multiple frames with uniform speed is expected to achieve near-to-optimal energy saving. That’s one of the key ideas behind our feedback DVS system design to emphasize speed uniformity as illustrated in Section 5.5 of Chapter 5.

Appendix B

Electromigration with Free Stress at Both Ends

Consider an EM process with zero stress at both ends of the metal line and with non-uniform temperature distribution across the metal line. In the steady state, the mechanical stress along the line reaches its steady distribution, or $\frac{\partial\sigma}{\partial t} = 0$. And the following equation is hold (from Equation (6.2)):

$$\nabla J = \frac{\partial}{\partial x}\left(\beta\left(T(x)\right)\left(\frac{\partial\sigma(x)}{\partial x} - \alpha(T(x))\right)\right) = 0$$

with the boundary conditions: $\sigma(0,t) = \sigma(l,t) = 0$, where $T(x)$ is the temperature profile.

Therefore, $\beta\left(T(x)\right)\left(\frac{\partial\sigma(x)}{\partial x} - \alpha(T(x))\right) = J$, where J is a constant and represents the steady state atom flux. Thus, in the steady state of EM process, there exists a constant atomic flux from one end (cathode) of the metal line to the other (anode). It follows that $\left(\frac{\partial\sigma(x)}{\partial x} - \alpha(T(x))\right) = \frac{J}{\beta(T(x))}$. By integrating both sides of this equation along the metal line, and noticing that $\sigma(0) = \sigma(l) = 0$, we obtain $-\int\limits_0^l \alpha(T(x)dx = J\int\limits_0^l \frac{1}{\beta(T(x))}dx$. And finally the steady state atomic flux can be expressed as:

$$J = -\frac{\int_0^l \alpha(T(x))dx}{\int_0^l \frac{1}{\beta(T(x))}dx}$$

Bibliography

[1] FFmpeg project. `http://ffmpeg.mplayerhq.hu/`.

[2] Linux kernel CPUfreq subsystem. `http://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufreq.html`.

[3] Movie trailers. `http://trailer.nerodigital.com/enu/index.html`.

[4] Movie trailers. `http://www.divx.com/movies`.

[5] Movie trailers. `http://www.apple.com/quicktime/guide/hd/`.

[6] The 2005 international technology roadmap for semiconductors (ITRS), 2005.

[7] A. H. Ajami, K. Banerjee, and M. Pedram. Modeling and analysis of non-uniform substrate temperature effects on global ulsi interconnects. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):849–861, 2005.

[8] S. M. Alam, G. C. Lip, C. V. Thompson, and D. E. Troxel. Circuit level reliability analysis of Cu interconnects. In *Proc. of the 5th International Symposium on Quality Electronic Design (ISQED'04)*, pages 238–243, 2004.

[9] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi. Overhead-conscious voltage selection for dynamic and leakage power reduction of time-

constraint systems. In *Proc. of Design, Automation and Test Europe Conference (DATE2004)*, 2004.

[10] K. Banerjee and A. Mehrotra. Global (interconnect) warnning. *IEEE Circuits and Device Magazine*, pages 16–32, September 2001.

[11] L. Benini and G. D. Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer, 1998.

[12] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*, 37(11):68 – 74, November 2004.

[13] D. T. Blaauw, C. Oh, V. Zolotov, and A. Dasgupta. Static electromigration analysis for on-chip signal interconnects. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):39–48, January 2003.

[14] J. R. Black. Mass transport of aluminum by momentum exchange with conducting electrons. In *IEEE Int. Rel. Phys. Symp.*, pages 148–159, 1967.

[15] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, and R. Rajamony. The case for power management in web servers. In R. Graybill and R. Melhem, editors, *Power Aware Computing*. Kluwer Academic Publications, 2002.

[16] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. of the 7th International Symposium on High-Performance Computer Architecture (HPCA-7)*, January 2001.

[17] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architecture-level power analysis and optimizations. In *Proc. 27th Annual International Symp. on Computer Architecture (ISCA)*, pages 83–94, June 2000.

[18] D. Burger and T. M. Austin. The simplescalar toolset, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.

[19] I. Castillo. *Introduction to Direct Hardware Access on Pocket PC*. `http://www.columns.pocketnow.com`, September 2001.

[20] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A distributed real-time mpeg video audio player. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 142 – 153, 1995.

[21] Y.-K. Cheng and S.-M. Kang. A temperature-aware simulation environment for reliable ULSI chip design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1211–20, October 2000.

[22] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat. Analytical thermal model for multilevel VLSI interconnects incorporating via effect. *IEEE Electron Device Letters*, 23(1):31–33, January 2002.

[23] K. Choi, K. Dantu, W. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a mpeg decoder. In *Proceedings of International Conference on Computer Aided Design*, pages 732–37, November 2002.

[24] K. Choi, R. Soma, and M. Pedram. Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding. In *Proceedings of 41st Design Automation Conference*, pages 544 – 549, June 2004.

[25] Z.-S. Choi, C. L. Gan, F. Wei, C. V. Thompson, J. H. Lee, K. L. Pey, and W. K. Choi. Fatal void size comparisons in via-below and via-above Cu dual-damascene interconnects. In *Materials, Technology and Reliability of Advanced Interconnects – 2004*, volume 812 of *Mater. Res. Soc. Symp. Proc.*, pages 373–378, 2004.

[26] E. Chung, L. Benini, and G. D. Micheli. Contents provider-assisted dynamic voltage scaling for low energy multimedia applications. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 42–47, August 2002.

[27] J. Clement. Reliability analysis for encapsulated interconnect lines under DC and pulsed DC current using a continuum electromigration transport model. *J. Appl. Phys.*, 82(12):5991, December 1997.

[28] J. Clement. Electromigration modeling for integrated circuit interconnect reliability analysis. *IEEE Trans. on Device and Materials Reliability*, 1(1):33–42, March 2001.

[29] J. A. Davis and J. D. Meindl, editors. *Interconnect Technology and Design for Gigascale Integration*. Kluwer, 2003.

[30] D.Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, pages 171–182, January 2001.

[31] N. Dragone, A. Aggarwal, and L. R. Carley. An adaptive on-chip voltage regulation technique for low-power applications. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 20–24, July 2000.

[32] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

[33] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, third edition, 1994.

[34] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, third edition, 1998.

[35] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and dvs processors. In *Proceedings of International Symposium on Low Power Electronics and Design 2001 (ISLPED'01)*, August 2001.

[36] W. Guo, Z. Li, H. Zhu, W. Zhang, Y. Ji, Y. Sun, and G. Shen. Temperature gradient impact on electromigration failure in vlsi metalization. In *Proceedings of 14th IEEE SEMI-THERM Symposium*, 1998.

[37] M. Hauschildt, M. Gall, S. Thraasher, P. Justison, L. Michaelson, H. Kawasaki, and P. S. Ho. Statistical analysis of electromigration lifetimes and void evolution for cu interconnects. In *Materials, Technology and Reliability of Advanced Interconnects – 2004*, volume 812 of *Mater. Res. Soc. Symp. Proc.*, pages 379–384, 2004.

[38] L. He, W. Liao, and M. R. Stan. System level leakage reduction considering the interdependence of temperature and leakage. In *Proceedings of 41st Design Automation Conference (DAC)*, pages 12–17, June 2004.

[39] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A control theoretic analysis of RED. In *Proceedings of IEEE INFOCOM*, April 2001.

[40] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/avc baseline profile decoder complexity analysis. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 13(7):704–716, July 2003.

[41] C.-H. Hsu and W.-C. Feng. Reducing overheating-induced failures via performance-aware cpu power management. In *The Sixth LCI International Conference on Linux Clusters: The HPC Revolution 2005*, April 2005.

[42] C.-H. Hsu and U. Kremer. The design, implementation, and evaluation of a compiler algorithm for cpu energy reduction. In *Proc. of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 38–48, 2003.

[43] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proceedings of 41st Design Automation Conference (DAC)*, pages 878–883, June 2004.

[44] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, December 2001.

[45] W. R. Hunter. Self-consistent solutions for allowed interconnect current density-part I: Implications for technology evolution. *IEEE Trans. on Electron Devices*, 44(2):304–309, February 1997.

[46] W. R. Hunter. Self-consistent solutions for allowed interconnect current density-part II: Application to design guidelines. *IEEE Trans. on Electron Devices*, 44(2):310–316, February 1997.

[47] C. Im and S. Ha. Dynamic voltage scheduling with buffers in low- power multimedia applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(4):686–705, November 2004.

[48] R. Jejurikar and R. Gupta. Procrastination scheduling in fixed priority real-time systems. In *Proc. of the 2004 ACM conference on Languages, compilers, and tools for embedded systems (LCTES'04)*, June 2004.

[49] N. K. Jha. Low power system scheduling and synthesis. In *IEEE Int. Conf. on Computer-Aided Design*, November 2001.

[50] K. Jonggook, V. C. Tyree, and C. R. Crowell. Temperature gradient effects in electromigration using an extended transition probability model and temperature gradient free tests. In *Proceedings of the IEEE Integrated Reliability Workshop (IRW)*, pages 24–40, 1999.

[51] M. A. Korhonen and P. Bøgesen. Stress evolution due to electromigration in confined metal lines. *J. Appl. Phys.*, 73(8):3790, April 1993.

[52] J. R. Lorch and A. J. Smith. PACE: A new approach to dynamic voltage scaling. *IEEE Tran. on Computers*, 53(7):856–869, July 2004.

[53] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, June 2001.

[54] C. Lu, G. A. Alvarez, and J. Wilkes. Aqueduct: Online data migration with performance guarantees. In *Proceedings of the USENIX Conference on File and Storage Technologies*, pages 219–230, January 2002.

[55] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Design and evaluation of a feedback control EDF scheduling algorithm. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, pages 56–67, December 1999.

[56] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems Journal*, 23:85–126, 2002.

[57] Y. Lu, L. Benini, and G. D. Micheli. Dynamic frequency scaling with buffer insertion for mixed workloads. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 21(11):1284–1305, November 2002.

[58] Y. Lu, A. Saxena, and T. F. Abdelzaher. Differentiated caching services: a control-theoretical approach. In *Proceedings of the International Conference on Distributed Computing Systems*, April 2001.

[59] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In *Proceedings of the 2002 International Conferences on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 156–163, October 2002.

[60] Z. Lu, J. Hein, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling. In *the 2002 Workshop on Self-healing, Adaptive, and Self-managed Systems (SHAMAN'02)*, June 2002. in conjunction with ICS 2002.

[61] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron. Interconnect lifetime prediction under dynamic stress for reliability-aware design. In *Proc. of International Conference on Computer Aided Design*, pages 327–334, November 2004.

[62] Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *Proc. of International Conference on Computer Design*, pages 489–96, October 2003.

[63] Z. Lu, J. Lach, M. Stan, and K. Skadron. Banking chip lifetime: Opportunities and implementation. In *Workshop on High Performance Computing Reliability Issues*, February 2005. in conjunction with HPCA 2005.

[64] Z. Lu, J. Lach, M. Stan, and K. Skadron. Improved thermal management with reliability banking. *IEEE Micro.*, 25(6):40–49, November/December 2005. 2005 special issue on Reliability-Aware Microarchitectures.

[65] Z. Lu, J. Lach, M. Stan, and K. Skadron. Design and implementation of an energy efficient multimedia playback system. In *Proc. of the 40th Asilomar Conference on Signals, Systems and Computers*, October 2006.

[66] Z. Lu, Y. Zhang, M. R. Stan, J. Lach, and K. Skadron. Procrastinating voltage scheduling with discrete frequency sets. In *Proc. of Design, Automation and Test Europe Conference (DATE)*, March 2006.

[67] M. Mattavelli and S. Brunetton. Implementing real-time video decoding on multimedia processors by complexity prediction techniques. *IEEE Transactions on Consumer Electronics*, 44(3):760–767, August 1998.

[68] K. Mayer-patel, B. C. Smith, and L. A. Rowe. The Berkeley software MPEG-1 video decoder. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(1):110–125, February 2005.

[69] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks, and S. Naffziger. Power and temperature control on a 90-nm Itanium family processor. *IEEE Journal of Solid-State Circuits*, 41(1):229–237, January 2006.

[70] R. C. McOwen. *Partial differential equations: methods and applications*. Prentice-Hall, 1995.

[71] W. K. Meyer, R. Solanki, and D. Evans. Near-threshold electromigration of damascene copper on tin barrier. In *Materials, Technology and Reliability of Advanced Interconnects – 2003*, volume 766 of *Mater. Res. Soc. Symp. Proc.*, pages 145–150, 2003.

[72] B. Mochocki, X. Hu, and G. Quan. A unified approach to variable voltage scheduling for nonideal dvs processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(9):1370– 1377, September 2004.

[73] T. Mudge. Power: A first class design constraint for future architectures. *IEEE Computer*, 34(4), April 2001.

[74] H. V. Nguyen, C. Salm, B. Krabbenborg, K. Weide-Zaage, J. Bisschop, A. J. Mouthaan, and F. G. Kuper. Effect of thermal gradients on the electromigration lifetime in power electronics. In *Proc. of the 42nd Annual International Reliability Physics Symposium (IRPS)*, pages 619–620, 2004.

[75] E. T. Ogawa, K.-D. Lee, V. A. Blaschke, and P. S. Ho. Electromigration reliability issues in dual-damascene Cu interconnections. *IEEE Trans. on Reliability*, 51(4):403–419, December 2002.

[76] Y.-J. Park, V. K. Andleigh, and C. V. Thompson. Simulations of stress evolution and the current density scaling of electromigration-induced failure times in pure and alloyed interconnects. *J. Appl. Phys.*, 85(7):3546–3555, April 1999.

[77] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proc. of 18th Symposium on Operating Systems Principles*, October 2001.

[78] D. Ponomarev, G. Kucuk, and K. Ghose. Dynamic allocation of datapath resources for low power. In *Proc. Workshop on Complexity-Effective Design (WCED-01)*, June 2001.

[79] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proc. of the Conf. on Mobile Computing and Networking*, July 2001.

[80] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni. Pushing asic performance in a power envelope. In *Proceedings of the 40th conference on Design automation*, pages 788 – 793, June 2003.

[81] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective.* Prentice Hall, 2 edition, December 2002.

[82] A. Ramakrishnan and M. G. Pecht. A life consumption monitoring methodology for electronic systems. *IEEE Trans. on Components and Packaging Technologies*, 26(3):625–634, September 2003.

[83] R. Rao and S. Vrudhula. Energy optimal speed control of devices with discrete speed sets. In *Proc. of the 42nd annual conference on Design automation*, June 2005.

[84] S. Rochel, G. Steele, J. R. Lloyd, S. Z. Hussain, and D. Overhauser. Full-chip reliability analysis. In *Proc. Int. Reliability Physics Symposium*, pages 356–362, 1998.

[85] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 1987.

[86] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, April 1990.

[87] H. Sanchez et al. Thermal management system for high-performance powerPC microprocessors. In *Proceedings of COMPCON'97*, February 1997.

[88] M. E. Sarychev, Y. V. Zhitnikov, L. Borucki, C.-L. Liu, and T. M. Markhviladze. General model for mechanical stress evolution during electromigration. *J. Appl. Phys.*, 86(6):3068–75, September 1999.

[89] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings of 8th International Symposium on High-Performance Computer Architecture (HPCA'02)*, February 2002.

[90] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware QoS management in web servers. In *Proc. of Real-Time Systems Symposium (RTSS)*, pages 63– 72, December 2003.

[91] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli. Dynamic voltage scaling for portable systems. In *Proc. Design Automation Conf.*, June 2001.

[92] A. Sinha and A. P. Chandrakasan. Energy efficient real-time scheduling. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, November 2001.

[93] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, pages 17–28, February 2002.

[94] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. Stan, and W. Huang. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. on Architecture and Code Optimization*, 1(1):94–125, March 2004.

[95] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankanarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. of the 30th International Symposium on Computer Architecture*, pages 2–13, June 2003.

[96] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proc. of the 17th Annual ACM International Conference on Supercomputing*, pages 109–120, June 2003.

[97] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proc. of the 31st International Symposium on Computer Architecture*, pages 276–287, June 2004.

[98] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. of the 32st International Symposium on Computer Architecture*, June 2005.

[99] J. A. Stankovic, C. Lu, S. H. Son, and G. Tao. The case for feedback control real-time scheduling. In *Proceedings of the IEEE Euromicro Conference on Real-Time*, June 1998.

[100] D. C. Steere, A. Goel, J. Gruenburg, D. McNamee, C. Pu, and J. Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *Proceedings of the Third Symposium on Operating System Design and Implementation*, pages 145–158, February 1999.

[101] V. Swaminathan and K. Chakrabarty. Network flow techniques for dynamic voltage scaling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23:1385–1398, October 2004.

[102] Y. Tan, P. Malani, Q. Qiu, and Q. Wu. Workload prediction and dynamic voltage scaling for mpeg decoding. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 911–916, January 2006.

[103] L. M. Ting, J. S. May, W. R. Hunter, and J. W. McPherson. AC electromigration characterization and modeling of multilayered interconnects. In *Proc. Int. Reliability Physics Symposium*, pages 311–316, March 1993.

[104] A. Varma, B. Ganesh, M. Sen, S. R. Choudhary, L. Srinivasan, and B. Jacob. A control-theoretic approach to dynamic voltage scaling. In *Proceedings of International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES)*, pages 255–266, October 2003.

[105] S. Velusamy, K. Sankaranarayanan, D. Parikh, T. Abdelzahe, and K. Skadron. Adaptive cache decay using formal feedback control. In *Proceedings of the 2002 Workshop on Memory Performance Issues*, May 2002.

[106] R. A. Wachnik, R. G. Filippi, T. M. Shaw, and P. C. Lin. Practical benefits of the electromigration short-length effect, including a new design rule methodology and an electromigration resistant power grid with enhanced wireability. In *2000 Symposium on VLSI Technology Digest*, pages 220–221, 2000.

[107] T.-Y. Wang and C. C.-P. Chen. 3-D thermal-ADI: A linear-time chip level transient thermal simulator. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1434–45, December 2002.

[108] T.-Y. Wang, J.-L. Tsai, and C. C.-P. Chen. Thermal and power integrity based power/ground networks optimization. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, volume 2, pages 830–835, February 2004.

[109] Wikipedia. Pentium 4. `http://en.wikipedia.org/wiki/Pentium_4`.

[110] L. S. Y. Wong, S. Hossain, and A. Walker. Leakage current cancellation technique for low power switched-capacitor circuits. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 310–315, August 2001.

[111] E. Wu, J. Sune, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, and D. Harmon. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. *Solid-State Electronics*, 46(11):1787–1798, November 2002.

[112] F. Xie, M. Martonosi, and S. Malik. Intra-program dynamic voltage scaling: Bounding opportunities with analytical modeling. *ACM Transactions on Architecture and Code Optimization*, 1, September 2004.

[113] R. Xu, C. Xi, R. Melhem, and D. Moss. Practical PACE for embedded systems. In *Proc. of the 4th ACM International Conference on Embedded Software (EMSOFT '04)*, 2004.

[114] H. Ye, C. Basaran, and D. C. Hopkins. Numerical simulation of stress evolution during electromigration in IC interconnect lines. *IEEE Trans. on Components and Packaging Technologies*, 26(3):673–681, September 2003.

[115] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, October 2003.

[116] S. Zafar, B. Lee, J. Stathis, A. Callegar, and T. Ning. A model for negative bias temperature instability (NBTI) in oxide and high-k pfets. In *2004 Symposium on VLSI Technology and Circuits*, June 2004.

[117] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal procrastinating voltage scheduling for hard real-time systems. In *Proc. of the 42nd annual conference on Design automation*, June 2005.

[118] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical Report CS-2003-05, U.Va. Department of Computer Science, March 2003.

[119] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, pages 84–93, May 2004.