UC Riverside
UC Riverside Electronic Theses and Dissertations

Title
System-Level Electromigration-Induced Dynamic Reliability Management

Permalink
https://escholarship.org/uc/item/9tc8w0b9

Author
Kim, Taeyoung

Publication Date
2017

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

System-Level Electromigration-Induced Dynamic Reliability Management

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Taeyoung Kim

June 2017

Dissertation Committee:

Dr. Sheldon X.-D. Tan, Chairperson
Dr. Nael Abu-Ghazaleh
Dr. Zizhong Chen
Dr. Daniel Wong

Copyright by
Taeyoung Kim
2017

The Dissertation of Taeyoung Kim is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

This thesis could not have been completed without the great support that I have received from so many people over the years. I wish to offer my most heartful thanks to the following people.

I would like to thank my advisor, Dr. Sheldon X.-D. Tan for guiding and supporting me over years. He has set an example of excellence as researcher, mentor, instructor and role model. His kindness, insight and suggestions always lead me to the right way.

I would like to thank my thesis committee members, Dr. Nael Abu-Ghazaleh, Dr. Zizhong Chen, Dr. Daniel Wong for their direction, dedication and invaluable advice. In addition to committee members, I would like to thank two other faculties, Dr. Hyoseung Kim and Dr. Qi Zhu for the discussion, ideas and feedback on Chapter 7 and Chapter 3 in this thesis.

I would like to thank all the members in our VSCLAB (VLSI System & Computation Laboratory) at University of California at Riverside (UCR). I thank especially Xin, Kai, Hao-Bao, Yan, David, Yue, Hengyang, Zhongdong, Chase, Zeyu, Han, Lebo, and Shaoyi for the collaborative research works, discussion and help, which lead to the presented works in this thesis. I appreciate the friendship of my fellow students in UCR.

Last but not least, I would like to thank my family, Yunji, Jihu (Theodore), and Sunghu (Edward), and my parents for the love, support, and constant encouragement during the years of my study. I undoubtedly could not have done this without them.

To Yunji, Jihu (Theodore), Sunghu (Edward) and my parents for all the support.

ABSTRACT OF THE DISSERTATION

System-Level Electromigration-Induced Dynamic Reliability Management

by

Taeyoung Kim

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2017
Dr. Sheldon X.-D. Tan, Chairperson

Technology scaling has led to further processor integration, and future manycore chips will have more cores integrated. However, due to the diminishing of Dennards scaling, the power density of chips starts to increase for current and future technology nodes. Because of this, only a certain percentage of a manycore processor can be powered on because of power and temperature limitations. These trends have resulted in so-called dark silicon manycore processors. Additionally, reliability is becoming a limiting constraint in high-performance nanometer VLSI chip designs due to the high failure rates in deep submicron and nanoscale devices. It is expected that future chips will show signs of reliability-induced aging much faster than the previous generations. Among of many reliability effects, electromigration (EM)-induced reliability has become a major design constraint due to the aggressive transistor and interconnect scaling and increasing power density.

This thesis focuses on developing new system level EM-induced dynamic reliability managements on many different systems. Specifically, first, I develop system level management for real-time embedded systems. I investigate a new lifetime optimization technique

for real-time embedded processors considering the electromigration-induced reliability. The new approach is based on a recently proposed physics-based electromigration (EM) model for more accurate EM assessment of a power grid network at the chip level. Second, I develop a new energy and lifetime optimization techniques for emerging dark silicon manycore microprocessors considering both hard long-term reliability effects (hard errors) and transient soft errors. To optimize EM-induced lifetime, I apply the adaptive Q-learning based method, which is suitable for dynamic runtime operation as it can provide cost-effective yet good solutions. Third, I develop a new dynamic reliability management (DRM) techniques at the system level for emerging low power dark silicon manycore microprocessors operating in near-threshold region. I mainly consider the electromigration (EM) recovery effects. To leverage the EM recovery effects, which was ignored in the past, at the system-level, I develop a new equivalent DC current model to consider recovery effects for general time-varying current waveforms so that existing compact EM model can be applied. Fourth, I develop a new approach for cross-layer electromigration (EM) induced reliability modeling and optimization at physics, system and data center levels. To speed up the online optimization for energy in a data center, I investigate a new combined data center power and reliability compact model using a learning based approach in which a feed-forward neural network (FNN) is trained to predict energy and long term reliability for each processor under data center scheduling and workloads. Lastly, I develop long-term reliability management for GPU architectures using spatial multitasking, which allows GPU computing resources to be partitioned among multiple applications. I find that the existing reliability-agnostic thread block scheduler for spatial multitasking is effective in achieving high GPU utilization,

but poor in reliability. I develop and implement a long-term reliability-aware thread block scheduler in GPGPU-sim, and compare it against the existing reliability-agnostic scheduler.

Contents

List of Figures xii

List of Tables xv

1 Introduction 1
- 1.1 Motivation . . . 1
  - 1.1.1 Reliability issue for real-time embedded systems . . . 3
  - 1.1.2 Reliability issue for dark silicon processors . . . 4
  - 1.1.3 Reliability issue for near-threshold dark silicon processors . . . 7
  - 1.1.4 Reliability issue for data center systems . . . 8
  - 1.1.5 Reliability issue for general-purpose graphics computing units . . . 10
- 1.2 Dissertation contributions . . . 11
- 1.3 Organization . . . 16

2 Electromigration reliability model 17
- 2.1 Review of the physics-based EM modeling . . . 17
  - 2.1.1 Void dynamics: nucleation and growth phases . . . 20
  - 2.1.2 EM assessment at power grid level . . . 24
- 2.2 Summary . . . 26

3 Reliability-aware lifetime optimization for real-time embedded systems 27
- 3.1 System-level EM-induced reliability model . . . 28
- 3.2 Real-time embedded system models . . . 29
- 3.3 Proposed new lifetime optimization method . . . 31
  - 3.3.1 The new lifetime optimization flow . . . 32
  - 3.3.2 Formulation one: continuous constrained nonlinear optimization . . . 36
  - 3.3.3 Formulation two: mixed-integer linear programming . . . 37
- 3.4 Numerical results and discussions . . . 44
  - 3.4.1 Experimental setup . . . 50
  - 3.4.2 Evaluation of proposed lifetime optimization . . . 51
  - 3.4.3 Core utilization effects and trade-off on energy and lifetime . . . 52
  - 3.4.4 Solution quality study and analysis . . . 53

3.4.5 Trade-off between performance and lifetime 54
3.5 Summary 55

4 Learning-based reliability management and energy optimization for marycore dark silicon processors 57
4.1 Review of system-level EM and soft error reliability models 58
4.1.1 System-level EM reliability model 58
4.1.2 Soft error reliability model considering DVFS impacts 59
4.1.3 Impact of process technology on soft error reliability model 61
4.2 New dynamic lifetime and energy optimization methods for dark silicon . . 62
4.2.1 Q-learning based formulation and solution for lifetime and energy optimizations 63
4.2.2 MILP based formulation and solution for lifetime optimization . . . 69
4.2.3 Implementation of the dark silicon evaluation platform 70
4.2.4 Time complexity analysis 74
4.2.5 Practical application of the proposed DRM with reliability models . 74
4.3 Numerical results and discussions 76
4.3.1 Experimental setup 76
4.3.2 Evaluation of the proposed Q-Learning lifetime optimization method 77
4.3.3 Accuracy and convergence rate of proposed Q-learning DRM method 78
4.3.4 Hard and soft errors in dark silicon manycore processor 83
4.3.5 Evaluation of proposed Q-Learning based energy optimization method 85
4.4 Summary 94

5 Recovery-aware dynamic reliability management for near-threshold dark silicon processors 96
5.1 Recovery-aware Electromigration modeling at system levels 97
5.1.1 New equivalent DC current based modeling for EM recovery effects . 97
5.1.2 EM modeling for varying temperature effects 103
5.2 New learning-based reliability management for near threshold dark silicon for EM recovery effects 106
5.2.1 Near threshold dark silicon 106
5.2.2 Framework of dark silicon in near-threshold computing region 107
5.2.3 SARSA-based learning algorithm for DRM considering long-term recovery 110
5.3 Numerical results and discussions 115
5.3.1 Evaluation of the lifetime impacts considering EM recovery effects . 115
5.3.2 Evaluation of the DRM for near-threshold dark silicon processors . . 116
5.3.3 DRM considering recovery effects 117
5.4 Summary 117

6 Cross-layer modeling and optimization for electromigration-induced reliability 119
6.1 EM-induced reliability model for a manycore processor in datacenter 120
6.2 EM-induced reliability-aware datacenter model 121

6.3 New reliability-constrained energy optimizatoin for datacenter 123
6.3.1 Neural networks for datacenter energy and reliability models 123
6.3.2 Q learning optimization for datacenter 126
6.3.3 Proposed new datacenter framework for energy and reliability 129
6.4 Numerical results and discussions 130
6.4.1 Experimental setup 130
6.4.2 Evaluations of proposed new modeling and optimization 131
6.5 Summary 133

7 Long-term reliability management for multitasking GPGPUs 135
7.1 System-level reliability resource consumption model 135
7.2 GPGPU Architecture and Stream Multiprocessor Scheduling 140
7.3 Simulation Framework for EM Assessment on GPGPU 141
7.4 Resource consumption rate-aware thread block scheduler for long-term reliability 145
7.5 Numerical results and discussions 149
7.5.1 Fixed Scheduling Performance 150
7.5.2 Monitoring-Aware Scheduling Evaluation 151
7.5.3 Sensitivity to SM partitioning 151
7.5.4 Threshold Exploration 152
7.6 Summary 154

8 Conclusion 155
8.1 Summary of research contributions 156
8.1.1 Reliability-aware lifetime optimization for real-time embedded systems 156
8.1.2 Learning-based reliability management and energy optimization for mary-core dark silicon processors 157
8.1.3 EM Recovery-aware dynamic reliability management for near-threshold dark silicon processors 158
8.1.4 Cross-layer modeling and optimization for EM-induced reliability in data center 159
8.1.5 Long-term reliability management for multitasking GPGPUs 159

Bibliography 161

List of Figures

1.1 Evolution of current densities: J_{max}, the maximum equivalent DC current density and J_{EM}, the current density for targeted lifetime [1] 2
1.2 (a) Total datacenter cost by primary causes of unplanned outage (Thousand dollars) (b) Power consumption breakdown for one server 9

2.1 Interconnect tree confined by diffusion barriers/liners [2]. 19
2.2 (a) EM-stress distribution change over time in a simple metal wire in nucleation phase. (b) EM-stress evaluation on cathode versus time using physics-based model [3]. 22
2.3 EM-stress distribution change over time in simple metal wire for void growth [3]. 23
2.4 Voltage of the first failed node in different simulation time [4]. 25

3.1 Single-rate and multi-rate task scheduling models 31
3.2 Multi-rate preemption 35
3.3 Core utilization effect - energy savings 44
3.4 Core utilization effect - lifetime improvement 46
3.5 The comparisons of simulated annealing and mixed-integer linear programming methods for the lifetime optimization for single-rate with 6, 12, and 24 tasks per one task set under different core utilizations (0.3 to 0.8 in x-axis) 50
3.6 The comparisons of simulated annealing and mixed-integer linear programming methods for the lifetime optimization for multi-rate with 6, 12, and 24 tasks per one task set under different core utilizations (0.3 to 0.8 in x-axis) 50
3.7 Trade-off between lifetime and performance (each triangle is different set of core utilization) 55

4.1 Q-Learning model with reliability-aware dark silicon framework 65
4.2 (a) SPLASH2 benchmark 64 multithreaded tasks power traces with 44 cores off(b) Thermal (color:degree) and EM lifetime (number:yrs) analysis on 64 cores 73
4.3 Lifetime improvements given power budget and performance deadline on 64-core dark silicon chip (a) PARSEC small task set (b) SPLASH-2 large task set 79

4.4 Q-learning result for performance deadline from (a,b) PARSEC tasks and light (c,d) SPLASH-2 tasks in 64-core dark silicon chip 80
4.5 Q-learning result for power constraints from heavy 4 PARSEC tasks and light 64 SPLASH-2 tasks in 64-core dark silicon chip 81
4.6 Post-validation with MILP for Q-learning accuracy (a) 4 heavy task PARSEC workloads (b) 64 light tasks SPLASH-2 workloads 82
4.7 Comparison between EM-induced lifetime and system-level soft error rate at different powers (by different DVFS configurations) on (a) PARSEC small tasks and (b) SPLASH-2 tasks 84
4.8 Impact of different process technologies on system-level soft error rate, from left bar, case A) Global DVFS, case B) our proposed DRM with only EM constraint, our proposed DRMs with both EM and SER constraints of case C) 45nm, case D) 32nm, and case E) 22nm 85
4.9 Energy optimization results (Global DVFS, proposed with EM, and with/without tight and loose soft error constraint from small task set on PARSEC benchmarks (different performance deadlines in (a) and (b)) 87
4.10 Energy optimization results (Global DVFS, proposed with EM, and with/without tight and loose soft error constraint from large task set on SPLASH-2 benchmarks (different power budgets in (a) and (b)) 88
4.11 Constraint violation cases 90
4.12 Convergence rate of proposed DRM method with EM-induced lifetime constraint in 64-core Dark Silicon (SPLASH-2 Tasks) 91
4.13 (a) The scalability analysis for our proposed DRM (case 1: two p-states with dark silicon, case 2: three p-states with dark silicon (b) Total average iteration number for both two cases 93

5.1 Stress evolution caused by periodical current density 98
5.2 Stress evolutions caused by actual currents and traditional effective DC current 99
5.3 (a) Original input driving current density. (b) Calculated EM DC equivalent current density with t_{nuc} 102
5.4 (a) Original input driving current density. (b) Calculated EM DC equivalent current density with two methods 104
5.5 Comparing the nucleation time of two equivalent methods and original stress 105
5.6 The DRM and NTC Framework 108
5.7 (a) two cases of power traces from proposed framework and (b) and the resulting MTTF without/with recovery effects 113
5.8 Performance, energy and EM-induced lifetime from proposed DRM considering recovery effects for three cases (1) Recovery effects with $T_{recovery} = 50s$ (first column) (2) Recovery effects with $T_{recovery} = 1000s$ (the second column) (3) Only DRM without recovery effects (the third column) 114

6.1 Feed-forward neural network structure and data configuration 125
6.2 The evaluation platform for datacenter and energy and reliability management algorithms 129

6.3 Validating violations with constraint limits (a) Average socket MTTF (b) Average cluster power (c) Tail latency 134

7.1 Consumption Rate of Rodina Benchmarks 138
7.2 GPU Architecture 139
7.3 Off-chip PDN (a), On-chip PDN (b), and details of PDN of each SM (c) . . 142
7.4 Simulation framework for long-term reliability assessment on GPGPU . . . 144
7.5 Consumption Rate of Rodina Benchmarks 149
7.6 Normalized Consumption Rate with Fixed Scheduling Policy 151
7.7 Sensitivity to rotation threshold for migration-aware scheduling 152

List of Tables

1.1 High performance computing (HPC) reliability and power issues [5, 6]) . . . 2

3.1 A preliminary measurement for preemptive effect on lifetime 35
3.2 Optimization method evaluation for low core utilization single-rate task (SA and MILP) 45
3.3 Optimization method evaluation for high core utilization single-rate task (SA and MILP) 47
3.4 Optimization method evaluation for low core utilization multi-rate task (SA and MILP) 48
3.5 Optimization method evaluation for high core utilization multi-rate task (SA and MILP) 49
3.6 Elapsed CPU Time to solve the proposed Simulated Annealing and MILP problems (second per one taskset) 53

4.1 Raw SEU Rate Per Microprocessor on different technologies [7] 62
4.2 An example of control states for a 3-core processor 64
4.3 Elapsed CPU Time to solve the proposed Q-learning and MILP problems . 92
4.4 Large-scale experiments with five p-state on 128-core and 256-core 94

5.1 Results for dynamic reliability management for 64-core near-threshold dark silicon 115

6.1 Accuracy analysis (RMSE) of the feed-forward neural network (FNN) model 132
6.2 Energy optimization for datacenter 132

7.1 Parameters used in EM analysis 144
7.2 Benchmarks separated into Consumption Groups 150
7.3 Sensitivity to SM partition for Backprop-Myocyte 153

Chapter 1

Introduction

1.1 Motivation

Long-term reliability is becoming a limiting constraint for high performance and embedded real-time system designs due to the high failure rates in deep submicron and nanoscale devices. The increase in failure rates is caused by high integration levels and higher power densities, which leads to excessive on-chip temperatures. The introduction of new materials, processes and devices, coupled with voltage scaling limitations and increasing power density will impose many new reliability challenges. The Exascale Roadmap from United States Department of Energy (U.S. DOE) reported potential power increasing issues in the future high performance computing (HPC) as seen in Table 1.1 [5]. Moreover, failure rates of large-scale HPC will dramatically increase, thus, it is expected to be order of magnitude of hours in the future in Table 1.1 [6]. The semiconductor industry faces the challenges to maintaining reliability such as the continued increase in die size and number of transistors and the constant scaling of transistors for performance [8]. Increasing transistor

Table 1.1: High performance computing (HPC) reliability and power issues [5, 6])

Year	**2009**	**2012**	**2016**	**2020**
Mean Time to Interrupt	1-4 days	5-19 hours	50-230 min	22-120 min
Power	$6MW$	$\sim 10MW$	$\sim 10MW$	$\sim 20MW$

density and thus power density is causing higher temperatures on chip, resulting in failure acceleration. Scaling to smaller transistors increases failure rates by shrinking the thickness of dielectrics. This has led the International Technology Roadmap for Semiconductor (ITRS) to predict the onset of significant reliability problems in the future, and at a pace that has not been seen in the past [9].

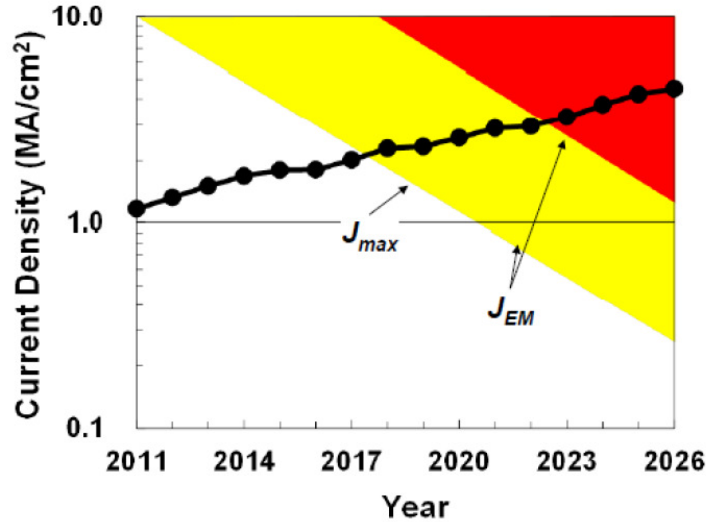


Figure 1.1: Evolution of current densities: J_{max}, the maximum equivalent DC current density and J_{EM}, the current density for targeted lifetime [1]

Moreover, an interconnect reliability is becoming a more serious issue of the semiconductor industry in the future. Fig. 1.1 describes the comparison of the evolution of the

maximum equivalent DC current density (from device requirement), J_{max}, and the current density for targeted lifetime, J_{EM}. J_{max} increases with scaling due to the reduction in interconnect cross-section and increasing operating frequency. Due to the continued scaling of interconnects, J_{max} will exceed the J_{EM} limits, (target limit-yellow and critical limit-red), as seen in Fig. 1.1

1.1.1 Reliability issue for real-time embedded systems

For safety-critical real-time embedded systems (such as satellite and surveillance systems) where reliability is as important as energy efficiency, reliability-aware energy management becomes a necessity. Some initial efforts have been carried out for system level reliability analysis for SoCs (system-on-a-chip). RAMP [10] is the first architecture level tool for modeling the long-term processor reliability of microprocessors at the design stage. The follow-up work by the same authors proposed a dynamic reliability management (DRM) concept by dynamic voltage and frequency scaling (DVFS) [11]. It showed that it was not sufficient to just manage the temperature or power from the reliability perspective. For real-time embedded systems, many existing works focus on minimizing energy consumption while meeting all the deadlines for various real-time task models. Existing works include power management schemes, which exploits the available static and/or dynamic slack in the systems [12, 13, 14, 15, 16, 17]. For long-term reliability effects, reducing power will implicitly improve the reliability of a processor. However, the two objectives, increasing lifetime and reducing power, are still not the same. Some reliability-aware power management works have been proposed recently [18, 19] by using low power techniques such as DVFS. However, most of those existing works focus on the transient errors instead of long-term

wearout failures. Recently a reliability-aware task allocation and scheduling method for multi-core embedded processors were proposed [20]. This work considers long-term failure mechanisms using general reliability models. However, such general models will not be accurate for specific failure mechanisms. Also, task allocation and scheduling are not best methods to manage the long-term wearout failures as they will not significantly change the temperatures of the chip as our study shows. Low power techniques like DVFS are more desired.

A method in [21] shows that the power/performance and reliability are intrinsically conflicting metrics and have strong interactions in SoC designs, and proposes a joint policy optimization method. Another dynamic reliability management method was proposed in [22], in which a simple PID based run-time control was applied to optimize the performance subject to the long-term reliability constraints. Recently, DVFS techniques considering negative bias temperature instability (NBTI), and time dependent dielectric breakdown (TDDB) effects were proposed for microprocessors [23, 24, 25]. A supply voltage scheduling technique was proposed for optimizing energy subject to NBTI constraints [26].

1.1.2 Reliability issue for dark silicon processors

For the last several decades, technology scaling has led to the continuous integration of devices, and microprocessors will have more cores integrated in the future. However, due to the failure of Dennard's scaling [27], chip power density is increasing on technology nodes since transistor and voltage scaling is no longer linear. The consequence is the emergence of so-called dark silicon manycore microprocessors, which mean only a percentage of

cores can be powered on the chip due to the power and temperature limitations. Recently, architecture researchers predicted that future many-core (100-1000 cores) silicon dies can only be powered up partially (so-called dark silicon) as power constraints will not allow all the cores to be active at the same time. Such manycore systems pose new challenges and opportunities for power/thermal and reliability management of those chips [28].

Existing works for dark silicon research mainly have been focused on the core organization, optimal number of cores, task allocation, migration, and scheduling [28, 29, 30, 31]. Moreover, those existing works focus on performance latency, bandwidth and energy efficiency for dark silicon chips. Recently, the reliability management methods for dark silicon manycore scaling have been studied [32, 33]. However, all of these works considered general reliability models, which will not be accurate for specific failure mechanisms. Recently, a new EM model has been used for energy optimization as a DRM but it only considered the EM model [34]. For dynamic power and thermal management, learning based methods have recently become popular. Many proposed methods applied Q-learning based method, which is a reinforcement machine learning method for the adaptive control [35, 36, 37, 38].

Energy-efficient or green computing is important for sustainability and environmental responsibility. This is also true for dark silicon many-core microprocessors as they may power many IT equipment and data centers in the near future. Power, performance and temperature limitations are traditional dominant factors for energy efficient high-performance and mobile computing. As technology advances, reliability starts to become another limiting factor in high-performance nanometer microprocessors due to the high failure rates in deep submicron and nanoscale devices. It is expected that future chips

will show signs of reliability-induced aging much sooner than the previous generations based on the prediction of ITRS 2014 [39].

Among many reliability effects, we consider electromigration (EM) and soft error-induced reliability effects as they have become major concerns for designers due to aggressive transistor scaling and increasing power density. EM effect is the dominant interconnect failure mechanism in the 22nm and below technology due to the shrinking wire width and thermal elevation due to FinFET devices [40], which will have immediate impacts on the metals above the FinFET devices. We want to stress that there exist many other long-term reliability effects such as NBTI (negative-bias temperature instability), hot carrier injection (HCI), TDDB (time dependent dielectric breakdown) for devices and stress migration and thermal migration for interconnects. However, in this work, we only consider EM reliability for the demonstration of the proposed reliability management techniques. The proposed techniques are orthogonal to other long-term reliability managements as those long-term reliability effects generally behave similarly or in a similar trend under their stressing conditions in terms of voltage, current and temperature [41].

On the other hand, soft-error related reliability has quite different impacts on VLSI chips (from the long-term reliability). This is especially true for chips operating in the very low voltage or even near threshold voltage regions. For practical chips, we have to consider both reliability effects at the same time. Although there are many soft-error mitigation techniques ranging from redundancy based design to software-based methods, it is important to study their impacts in the context of long-term reliability optimization techniques such as DVFS and on/off switching of cores from dark silicons.

As a result, in this work, we consider the energy or reliability optimization subject to the two kinds of reliability (long term and soft errors) and the power, performances, and temperature constraints. We look at the two system level control knobs: core status knobs to enable or disable a core due to dark silicon requirement, and dynamic voltage frequency scaling (DVFS) knobs for traditional power and thermal management. Hence, in this work, we try to solve the resulting optimal control problems to seek the best policies for DVFS and core status in the context of the two kinds of reliability constraints.

1.1.3 Reliability issue for near-threshold dark silicon processors

To further reduce powers for many applications, ultra-low power designs become necessary. Recent research has led to sub-threshold region where CMOS circuits are found to be capable of operating with a supply voltage of less than 200mV. The theoretical lower limit of V_{dd} has been determined to be 36 mV [42]. But at such low voltages, a leakage power dissipation increases drastically making the reduction in dynamic power insignificant. Also the circuit delay increases rapidly as the supply voltage is scaled down, resulting in decreased operation frequency or performance of the circuits.

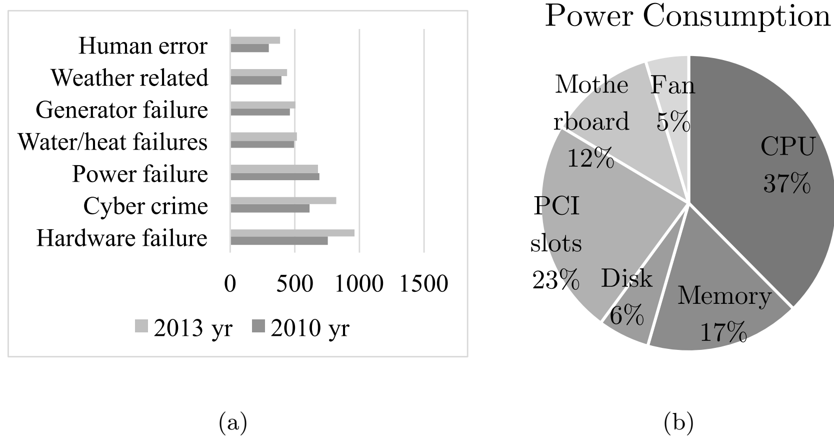
For dark silicon manycore processors operating in near threshold voltage, reliability becomes quite significant for the long-term reliability such electromigration. To address the increasing reliability issues, a system-level and run-time approach becomes more appealing. There are some existing works on dynamic reliability managements for dark silicon in the past [34, 43]. These works have been proposed to leverage the dark silicon many-core processors in order to save energy while maintaining performance considering reliability.

Run-time management of the heterogeneous dark silicon processors and optimal policy of core status have been addressed. Dynamic voltage frequency scaling method have been employed as energy saving techniques in those works. However, dynamic reliability management for near-threshold dark silicon processors has not been studied.

1.1.4 Reliability issue for data center systems

Datacenter downtime has become a major concern as every minute equates to money lost. An unplanned outage can easily cost a datacenter $8,000 dollars per minute of downtime and can even reach costs of $16,000 per minute of downtime. The main root causes of unplanned failures are largely attributed to power system failure and human error. Hardware failures, such as server failures, only account for about 4%-5% of unplanned downtime. However, these types of failures are often much more difficult and costly to recover from. As a result, unplanned datacenter outages caused by server failures are responsible for the highest incurred costs, compared to downtimes attributed to other root causes, despite their low rate of occurrence as seen in Fig. 1.2(a) [44]. This presents much of the motivation behind the work in this chapter as we develop a framework for reducing this hardware failure subject to performance constraints.

Although the servers consist of multiple components, existing works for datacenter hardware failure research have been mainly focused on the large scale studies in a hard disk [45] and memory failures [46]. However, in a typical server, the processor accounts for the majority of the power consumption at nearly 40% compared to other component such as memory and peripherals [47] in Fig. 1.2(b). Furthermore, a recent study found that processors are the leading cause of single node hardware failure in high performance computing



(a) (b)

Figure 1.2: (a) Total datacenter cost by primary causes of unplanned outage (Thousand dollars) (b) Power consumption breakdown for one server

clusters [48]. This trend is expected to become increasingly common as processor reliability is becoming a limiting constraint in high-performance processor designs due to high failure rates in deep submicron and nanoscale devices. Technology scaling has led to the continuous integration of devices, and processors will have more cores integrated. This growing trend for large scale many-core devices was brought upon by the increase in transistor density and the subsequent breakdown of Dennard Scaling. The result of which is the loss of power distribution scaling with transistor sizes, leading to increased chip temperatures, and the movement from utilization of a single powerful machine to a large cluster of machines which can help distribute workloads. However, large cluster system generates reliability concerns as we no longer can consider the reliability of just a single device or chip. This is especially true as each node in the datacenter begins to utilize highly integrated processors with their own reliability concerns. It is increasingly obvious that single server, or even chip level, reliability needs to be a large factor in how we address the reliability of numerous devices employed on a larger scale. In order to address these concerns, the relationship

between datacenter and processor reliability should be examined. The reliability issue for datacenter presents the challenge of correlating processor and datacenter cluster reliability. We examine reliability effects of processors under practical datacenter workloads and model the effects that the operating parameters of the servers have on the processor reliability.

1.1.5 Reliability issue for general-purpose graphics computing units

The use of general-purpose graphics computing units (GPGPUs) for high-performance computing has recently gained much attention. Due to the massive parallelism of GPGPUs, they can deliver significant performance improvement over traditional CPU-based computing devices. As the complexity of high-performance computing systems continues to increase, the probability of failure in one of the machines is also expected to increase. This naturally brings up the necessity of research in reliability, especially with the focus on GPGPUs.

While long-term reliability of computing systems has emerged as a serious problem, to the best of our knowledge, no prior work has assessed its impact on GPGPUs. Only some initial efforts on soft errors have been carried out for GPGPUs [49, 50, 51, 52, 53], where most of the radiation-induced (soft error) failure are caused by the corruption of memory resources. Our main contribution in this work is the quantification of the EM-induced interconnect reliability of GPGPUs using spatial multitasking, with widely-used GPGPU benchmarks. Spatial multitasking is a technique commonly used in GPUs to improve GPU utilization [54]. Under spatial multitasking, thread blocks from different applications are allocated to their own exclusive sets of Streaming Multiprocessors (SMs), enabling simultaneous execution of multiple kernels.

In the literature, several approaches to the EM assessment of chip interconnect have been proposed. Among them, one approach [55] proposed that the on-chip interconnects should be divided into a set of interconnect trees and the void nucleation time and void growth kinetics should be calculated for each tree. While being physically correct, this approach misses the dynamical nature of the failure development; the nucleated and growing voids in a number of limbs/segments of different trees affect the current densities in other void-less segments by changing the resistances of the void-containing segments. The latter undermines a validity of any steady state out filtration of the "immortal" trees. A more precise approach for the full-chip EM assessment would be to calculate the cooperative void evolution dynamics, which derives the continuously changing cross on-chip power delivery network (PDN) and current density distribution that captures changes in the void population. An interconnect time-to-failure (TTF) should be calculated as an instant in time when a change in V_{DD}/ V_{SS} reaches a threshold, here say 12% in the GPU case [56], however, this number can be changed by the different design specification and off-chip inductive effects. This time-iterative approach requires repeated calculation of the current densities in each PDN segment at each time-step.

1.2 Dissertation contributions

This dissertation presents new system-level EM-induced dynamic reliability management for many different systems. The major contributions of this dissertation are summarized as follows:

- For real-time embedded systems, a new lifetime optimization technique considering the electromigration-induced reliability has been developed. The new approach is based on a recently proposed physics-based electromigration (EM) model for more accurate EM assessment of a power grid network at the chip level [57]. A dynamic voltage and frequency scaling has been applied (by selecting the performance states or p-states of the tasks to manage the power), which lead to different lifetimes of the processor running different tasks over their periods. Both single-rate and multi-rate embedded systems with preemption is considered. The goal of the optimization is to maximize the EM-induced reliability (lifetime) of the embedded processor subject to the timing constraints. The resulting problem is a constrained nonlinear optimization problem. To solve the resulting problem, Two problem formulations and corresponding solutions are explored. First, the optimization problem is formulated as the continuous constrained nonlinear optimization problem using task's MTTFs as the variables, which is further solved by simulated annealing method. To find the optimal solutions with regard to the proposed EM-induced reliability model and assumptions, a different set of variables, and linearize both objective functions and constraints are selected. The resulting problem is solved by the mixed-integer linear programming (MILP) method, which however higher computational costs for large-scale optimization problems.

- A new energy and lifetime optimization techniques has been developed for emerging dark silicon manycore microprocessors considering both hard long-term reliability effects (hard errors) and transient soft errors. I consider a recently proposed physics-based EM reliability model to predict the EM-induced long-term reliability. For sys-

tem level soft error modeling, I consider the DVFS-aware soft error rate (SER) model and the Sum Of the Failure Rates (SOFR) method are employed. To model dark silicon, I consider thermal design power (TDP) as the power constraint for dark silicon manycore microprocessors. I employ both dynamic voltage and frequency scaling (DVFS) and dark silicon core state using On/Off switching action as the two control knobs. I show that on-chip power consumption has different (even contradicting) impacts on soft and hard reliability effects. Our study also shows that soft-error should be mitigated by other techniques if aggressive low power and high long-term reliability are pursued. I focus on two optimization techniques for improving lifetime and reducing energy. To optimize EM-induced lifetime, I first apply the adaptive Q-learning based method, which is suitable for dynamic runtime operation as it can provide cost-effective yet good solutions. The second lifetime optimization approach is the mixed-integer linear programming (MILP) method, which typically yields better solutions but at higher computational costs. To optimize the energy of a dark silicon chip subject to the both hard and soft reliability effects and performance constraints, the Q-learning method has been applied as well. A large class of multithreaded applications is used as the benchmark to validate and compare the proposed dynamic reliability management methods.

- For recovery-aware system-level EM management, I develop a new dynamic reliability management (DRM) techniques at the system level for emerging low power dark silicon manycore microprocessors operating in near threshold region. I mainly consider the EM failures. To leverage the EM recovery effects, which was ignored in the past, at

the system level, I develop a new equivalent DC current model to consider recovery effects for general time-varying current waveforms so that existing compact EM model can be applied. The new equivalent DC current is calculated in two steps: firstly, the equivalent square waveform is calculated so that peak and terminal stresses are matched, secondly, the parameterized equivalent DC current is derived in terms of the parameters of the fitted periodic square waveforms from the first step. The significance of the new EM current model is that it allows EM recovery effects can be considered at the system level for the first time and thus allow EM-induced lifetime of chips to be better managed at the system level. The system level energy optimization problems considering recovery-aware EM-induced reliability subject to power and performance constraints was framed by seeking the best dark silicon cores' voltage and on/off status. The resulting problem was solved by the State-Action-Reward-State-Action (SARSA) reinforcement learning algorithm.

- For data center reliability management, I develop a novel cross-layer approach to optimize the energy of a datacenter subject to long-term reliability and performance constraints. I consider a recently proposed physics-based EM reliability model to predict the EM reliability of full-chip power grid networks for long-term failures. EM has been previously identified as a major contributor to processor reliability in datacenters due to challenges of thermal management [58]. I stress the proposed method is orthogonal to other long-term reliability issues such as NBTI (negative biased temperature instability, TDDB (time-dependent dielectric breakdown), hot carriers etc. I show how the new physics-based dynamic EM model at the physics

level can be abstracted at the system level and even at the datacenter level. Our datacenter system-level power model is based on the BigHouse simulator, which is recently proposed and uses a combination of queuing theory and stochastic modeling. To speed up online optimization for energy in a datacenter, I develop a new combined datacenter power and reliability model using a learning based approach in which a feed-forward neural network (FNN) is trained to predict energy and long term reliability for each processor under datacenter scheduling and workloads. To optimize the energy and reliability of a datacenter model, I apply the efficient and adaptive Q-learning based reinforcement learning method.

- For general-purpose graphics computing units (GPGPUs), I develop long-term reliability management for GPGPUs using spatial multitasking for executing general-purpose workloads. I develop a distributed power delivery network model at functional unit granularity. I utilize this PDN model for our EM analysis of GPU architectures using a recently proposed physics-based EM reliability model [59] and consider the EM-induced time-to-failure (TTF) at the GPU system level as a reliability resource. For GPU scheduling, I focus on spatial multitasking, which allows GPU computing resources, i.e., SMs, to be partitioned among multiple applications. I find that the existing reliability-agnostic thread-block scheduler for spatial multitasking is effective in achieving high GPU utilization, but ineffective for reliability. I develop a long-term reliability-aware thread-block scheduler in GPGPU systems and evaluate with widely-used GPGPU benchmarks. I find our proposed spatial multitasking scheduling shows a reliability improvement of up to 30%.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 provides the review of fundamentals of EM model. Chapter 3 describes new lifetime task optimization techniques for real-time embedded processors considering the electromigration-induced reliability. Chapter 4 develop a new energy and lifetime optimization technique for emerging dark silicon manycore microprocessors considering hard and soft errors. In Chapter 5, I develop a new recovery-aware dynamic reliability management (DRM) techniques for emerging near-threshold dark silicon manycore microprocessors considering electromigraion (EM) reliability. In Chapter 6, I develop a novel cross-layer approach to optimizing the energy of a datacenter subject to long-term reliability and performance constraints. Lastly, in Chapter 7, I develop long-term reliability management for GPGPUs using spatial multitasking for executing general-purpose workloads.

Chapter 2

Electromigration reliability model

This chapter gives a review of the fundamental physics-based electromigration (EM) model and explains void dynamics, such as nucleation and growth phases. Stress evolution of void nucleation and growth phases can be represented as physics-based model. An accurate estimation of the void nucleation and growth times can be obtained from the hydrostatic stress evolution kinetics. For the full chip analysis, EM-induced reliability analysis method for power grid network is introduced.

2.1 Review of the physics-based EM modeling

EM is the physical phenomenon of the migration of metal atoms along a direction of applied electrical field. Atoms (either lattice atoms or defects/impurities) migrate toward the anode end of metal wire along the trajectory of conducting electrons. During the migration process, hydrostatic stress will be generated inside the embedded metal wire due to momentum exchange between lattice atoms and conduction electrons and is a prime cause

of void and hillock formation at the opposite ends of the wire. Indeed, when metal wire is embedded into a rigid confinement, which is the case with interconnect metallization, the wire volume changes (induced by the atom depletion and accumulation due to migration) create tension at the cathode end and compression at the anode ends of the line. Over time, the lasting unidirectional electrical load increases these stresses, as well as the stress gradient along the metal line. In some cases, usually when a line is long, this stress can reach a critical level, resulting in a void nucleation at the cathode and/or hillock formation at the anode end of line. The EM effects are mainly modeled and heavily used by empirical Black's equation [60] and Blech limit [61]. However, those models are not physics-based and they do not fully consider the predictability for varying stressed conditions and complicated interconnect wire structures. Additionally, they do not address the inherent redundancy in the power grid networks, which are the most vulnerable wires in a chip.

Traditional physics-based EM assessment models and simulation flows are attributed to a single interconnect line confined by the liners serving as barriers for atomic diffusion. However, modern power/ground (P/G) networks consist of large segments representing a continuously connected, highly conductive metal (copper, Cu) lines within one layer of metallization, terminated by diffusion barriers. These segments, which are also known as interconnect trees [55], may have multiple voltage input/output and current source ports represented by interlayer vias and contacts. The major difference between iso-lines and individual limbs of interconnect trees, is an absence of blocking boundaries at one or both ends of the limbs. It prevents atoms from accumulation/depletion, and eliminates related stress buildup at the branch ends, and, hence, makes tree decomposing on individ-

ual segments and a consequent traditional immortality assessment and mean-time-to-failure (MTTF) calculation as a groundless [57].

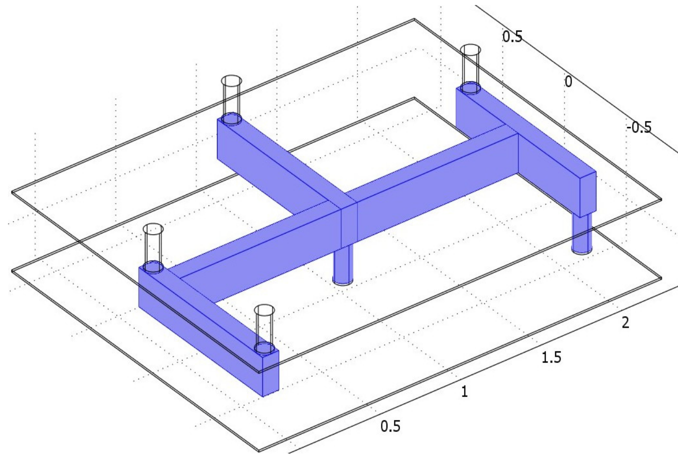


Figure 2.1: Interconnect tree confined by diffusion barriers/liners [2].

A physically viable approach to the EM assessment in chip interconnect was proposed in a number of publications, see for example [55], representing connected, high conductive metal lines within one layer of metallization, terminated by diffusion barriers as shown in Fig. 2.1 [2]. In their approach the on-chip interconnects should be divided into a set of interconnect trees and the void nucleation time and void growth kinetics should be calculated for each tree. While being physically correct (in the scope of used approximations), this approach has missed a dynamical nature of the failure development; the nucleated and growing voids in a number of limbs/segments of different trees affect the current densities in other void-less segments by changing the resistances of the void-containing segments. The later undermines a validity of any steady state out filtration of the "immortal" trees. A correct approach to the full-chip EM assessment requires calculating the cooperative void evolution dynamics resulting in the continuously changing cross P/G network current den-

sity distribution that in turn results in a change in the void population. An interconnect MTTF should be calculated as an instant in time when a change in $V_{DD}/\ V_{SS}$ reaches a threshold, say 10%. This time-iterative approach requires repeated calculation of the current densities in each P/G segment at each time-step. Here, we use 10% to simplify our work [4], however, this number can be changed by the different design specification and off-chip inductive effects.

2.1.1 Void dynamics: nucleation and growth phases

An accurate calculation of the void nucleation time inside an interconnect tree can be done based on resolving the hydrostatic stress evolution kinetics inside a multi-branch tree. The hydrostatic stress is an isotropic stress when an interconnect tree is under equal compressive or tensile stress in all directions. It can be done by solving the system of second order partial differential equations, [62], describing the hydrostatic stress (${\sigma^i}_{Hyd}$) evolution in each individual segment,

$$\frac{\partial {\sigma^i}_{Hyd}}{\partial t} = \frac{\partial}{\partial x}\left[\frac{{D^i}_a B^i \Omega}{k_B T^i}\left(\frac{eZ^i \rho j^i}{\Omega} + \frac{\partial {\sigma^i}_{Hyd}}{\partial x}\right)\right] \tag{2.1}$$

Here, D_a is the effective atomic diffusivity, eZ is the effective charge of migrating atoms, B is the effective bulk elasticity modulus, Ω is the atomic lattice volume, ρ is the copper resistivity, j is the current density, k_B is the Boltzmann constant, T is the temperature, x is the coordinate along the segment length, and t is time. Boundary conditions for these equations reflect the continuities of stresses at the limb joints, and the zero flux conditions at the terminating limb ends. Analytic solution of (2.1) allows one to extract the

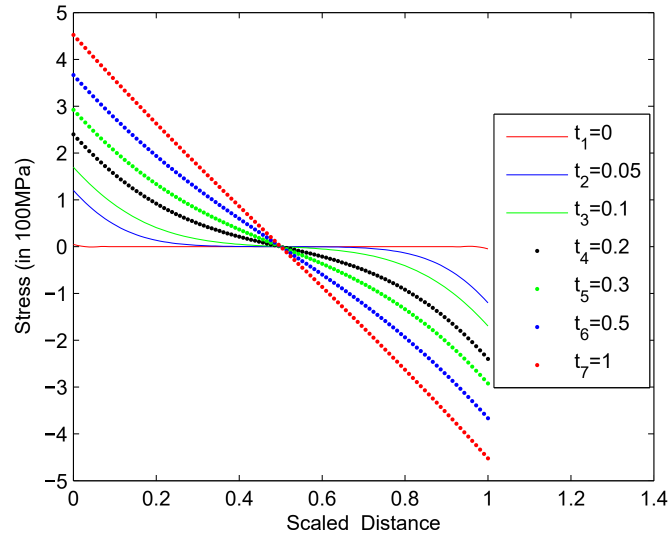
void nucleation time (t_{nuc}), which is determined as an instant in time when the developing stress reaches the critical value (σ_{crit}) needed for nucleation of the thermodynamically stable void [57, 63],

$$
\begin{aligned}
t_{nuc} \approx & \tau^{\star} e^{\frac{E_V}{kT}} e^{-\frac{f\Omega}{kT}\left(\sigma_{Res}+\frac{eZ\rho l}{4\Omega}j\right)} \\
& ln\left\{\frac{\frac{eZ\rho l}{4\Omega}j}{\sigma_{Res}+\frac{eZ\rho l}{4\Omega}j-\sigma_{CR}}\right\}
\end{aligned}
\tag{2.2}
$$

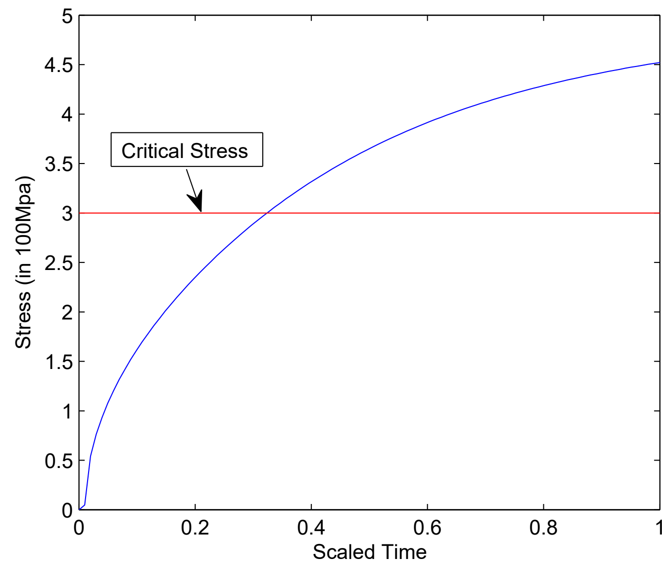
where $\tau^{\star} = \frac{l^2}{D_0}e^{E_D/kT}\frac{kT}{\Omega B}$. Here, j is the current density, T is temperatures, k_B is the Boltzmann's constant, l is the segment length, E_V and E_D are the activation energies of vacancy formation and diffusion, f is the ratio of volumes occupied by vacancy and lattice atom, σ_{crit} is the critical stress needed for the failure precursor nucleation (void/hillock). σ_{Res} is the residual stress of the metal segment from the cooling process and other factors.

Stress evolution of nucleation phase based on the model is shown in Fig. 2.2 [3]. Fig. 2.2(a) shows the EM-induced stress development for a single metal wire over time from the finite element analysis for a given current density and temperature setting [3]. Fig. 2.2(b) shows the stress evaluation over time. Each time unit here is 10^7 seconds. During this process, tensile stress (positive stress) will be developed at the cathode side (the left node), while compressive stress (negative stress) will be generated at the anode side of the wire (right node). When the tensile stress hits critical stress, a void will be generated, which is called nucleation time (t_{nuc}).

The second phase is the void size growth: voids are formed at t_{nuc} and grow at $t > t_{nuc}$. The wire resistance starts to increase over the time in the growth phase. In Fig. 2.3, the evolution of stress starting from the void nucleation time $t = 0$ till the steady state is achieved [3].



(a)



(b)

Figure 2.2: (a) EM-stress distribution change over time in a simple metal wire in nucleation phase. (b) EM-stress evaluation on cathode versus time using physics-based model [3].

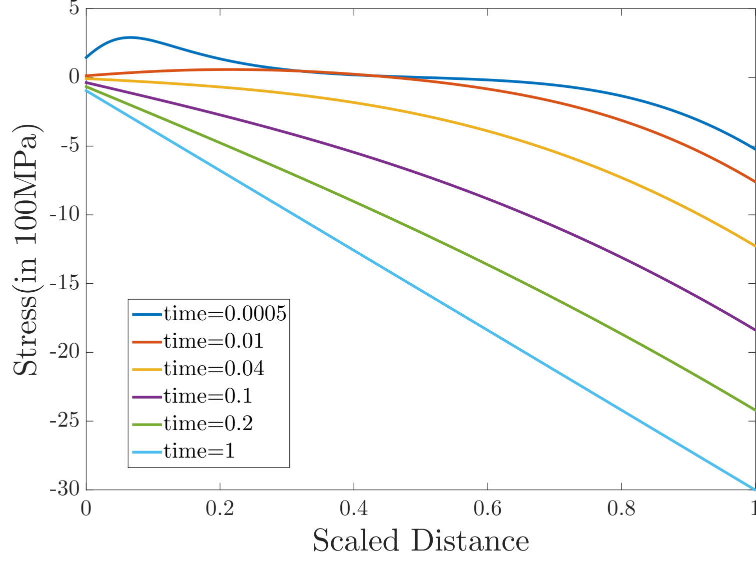


Figure 2.3: EM-stress distribution change over time in simple metal wire for void growth [3].

Since the drift velocity of the void edge relates to atomic flux as $\vartheta = \Omega j$, [64] we can express it as

$$\vartheta = \frac{D_a}{k_B T} eZ\rho_{Cu} j. \tag{2.3}$$

Kinetics of the line resistance change can be approximately described as:

$$\Delta r(t) = \vartheta(t - t_{nuc}) \left[\frac{\rho_{Ta}}{h_{Ta}} \left(\frac{1}{2H} + \frac{1}{W} \right) - \frac{\rho_{Cu}}{HW} \right] \tag{2.4}$$

Here ρ_{Ta} and ρ_{Cu} are the resistivity of the barrier material (Ta/TaN) and copper, W is the line width, H is the copper thickness, and h_{Ta} is the barrier layer thickness. As a result, the p/g network becomes a time-varying network and its voltage drops will keep changing over the time [57].

The saturated void size and the instant in time when saturation happens, is determined by a computation of the EM induced atomic flux by the back-flux of atoms generated

by the developed stress gradient, [65, 57]. Knowledge of the void kinetics and its saturated volume developed in a particular segment allows calculating the segment resistance evolution caused by passing current density. Upon the void saturation the further resistance change is stopped until the current density is changed due to other voids developed somewhere inside the same interconnect tree or in other trees.

2.1.2 EM assessment at power grid level

Because of the concern with the long-term average effects of the current, in EM related work, such as [66], a DC model of the power grid is generally assumed. At the chip level, EM failure typically is defined not by the failure of a single wire, but by the voltage drop of a power grid network as power grid network is most vulnerable to EM failures. In our problem formulation, each mortal wire, which is subject to the EM effect, will start to change its resistance value upon achieving the nucleation time. Finally, we end up with the power grid systems, which is linear, time-varying, and driven by the DC effective currents, which is modeled as $G(t)v(t) = I_{eff}$, where, $G(t)$ is a $n \times n$ time-varying conductance matrix; I_{eff} is the effective DC current source vector; $v(t)$ is the corresponding vector of nodal voltages and n is the size of unknown voltages. Because EM is a long term effect, the time scale used in this thesis is measured in months or years. In the EM-induced reliability analysis at a power/ground level, the voltage drops of the grid can be computed at the fixed time step for each core. Specifically, the resistance of one or more wires begins to increase after the nucleation time defined in (2.2) and then the resistance of a wire will start to increase based on (2.4). At each time step, we collect new wires whose nucleation times were reached, and compute the new resistance for existing wires in the growth phase

and the corresponding voltage drops of the whole grid. This process is repeated until the voltage drop of one or more nodes exceeds the critical voltage drops allowed (defined as 10% of V_{DD}) [59].

Fig. 2.4 shows one example in which the voltage of one node in a p/g network varies time after the creation of the first void in the network and its value can be tracked [4].

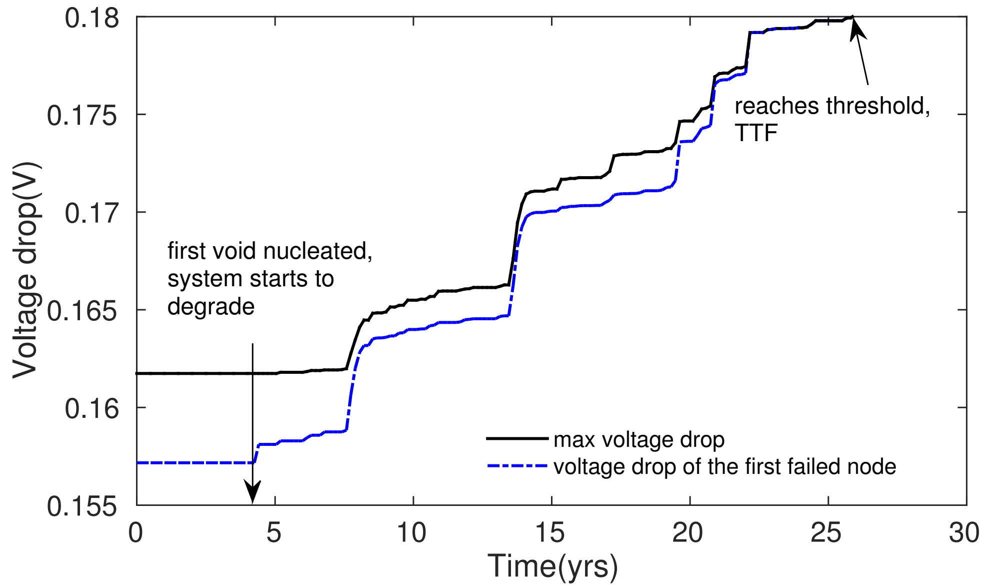


Figure 2.4: Voltage of the first failed node in different simulation time [4].

In the new EM-induced reliability analysis algorithm for p/g networks, we compute the voltage drops of the grids at fixed EM time step. The resistance of one or more wires begins to change (increase) starting with their nucleation times. At each time step, we collect new wires whose nucleation times were reached, and compute the new resistance for existing wires in the growth phases and corresponding voltage drops of the whole grids. This process is repeated until the voltage drop of one or more nodes exceeds the critical voltage drops allowed (defined as 10% of V_{DD}).

2.2 Summary

This chapter presented a review of the fundamental physics-based EM model and explained void dynamics, such as nucleation and growth phases. Stress evolution of void nucleation and growth phases can be represented as the physics-based model. An accurate estimation of the void nucleation and growth times have been obtained from the hydrostatic stress evolution kinetics. For the full chip analysis, EM-induced reliability analysis method for power grid network has been introduced.

Chapter 3

Reliability-aware lifetime optimization for real-time embedded systems

This chapter presents a new lifetime optimization techniques for real-time embedded processors considering the electromigration-induced reliability. This chapter aim at maximizing the EM-induced reliability of the embedded processor subject to the hard timing constraints. System-level EM reliability model will be introduced. To optimize at the system-level, dynamic voltage frequency scaling (DVFS) is applied. To solve the resulting problem, two problem formulations and corresponding solutions will be explored. First, We formulate the optimization problem as the continuous constrained nonlinear optimization problem using task's mean time to failure as the variables, which is solved by the simulated annealing (SA) method. In the second formulation, the linearized resulting problem

is solved by the mixed-integer linear programming (MILP) method to find the optimal solutions with regard to the proposed EM-induced reliability model and assumptions.

3.1 System-level EM-induced reliability model

At the system level, the embedded system will run on different tasks under different p-states. Let's assume that have a set of different time intervals Δp_k characterized by different workloads or p-state in terms of current density j_k and temperature T_k for a processor or a core. It means that $P = \sum_{k=1}^{n} \Delta p_k$ is the total execution time. Each kth workload, if it lasts till imaginary failure, provides time to failure TTF_k. Thus the failure rate at the kth workload, which last Δp_k is $\lambda_k = 1/TTF_k$. Then the average failure rate for the considered set of workloads is

$$\lambda_{avg} = \sum_{k=1}^{n} \frac{\Delta p_k}{\sum_{j=1}^{n} \Delta p_j} \lambda_k = \sum_{k=1}^{n} \frac{\Delta p_k}{P} \lambda_k \tag{3.1}$$

As a result, the expected time to failure or average lifetime of the whole processor, TTF is [67],

$$MTTF = \frac{1}{\lambda_{avg}} = \frac{1}{(\sum_{k=1}^{n}(\Delta p_k \frac{1}{MTTF_{R,k}}))/P} \tag{3.2}$$

where $MTTF_{R,k}$ is the actual MTTF under the k-th power and temperature settings for Δp_k period, assuming the chip works through n different power and temperature settings and $P = \sum_{k=1}^{n} \Delta p_k$. Each $MTTF_{R,k}$ will be computed based on the EM models discussed in the previous section.

3.2 Real-time embedded system models

In this section, we review the real-time task models considered in this work. Most of the embedded systems are real-time systems, where tasks are activated periodically, so timing and deadline should be carefully considered. We mainly consider two kinds of models for real-time systems: (1) Single-rate model, where all tasks in the system have the same activation period and deadline, and (2) Multi-rate model, where each task can have its own activation period and deadline. A task set is represented as $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_n\}$ where all tasks in the system are independent of each other. All tasks are scheduled on a single-core processor. For single-rate system, every task has the same activation period T. For multi-rate system, every task has its period and we use T_i to denote the period of task τ_i. In this work, we consider the deadline of each task as its activation period, where every task must finish the execution before its next activation, or missing deadline may have detrimental impacts on the whole system.

In *single-rate system*, every task has the same period and deadline, so the worst case for timing is that one task has to wait for all the other tasks to finish execution. As long as the sum of all task execution times is no greater than deadline T, the system is schedulable. We use Δt_i to represent the execution time of each task, so the timing constraints for single-rate system is expressed as $\sum_i \Delta t_i \leq T$.

In *multi-rate system*, the task scheduling is repeated at a hyper period that is the LCM (least common multiple) of all periods, so we just need to consider the p-state selection for tasks within a hyper period for multi-rate system. Thus, we assume that for every activation the task uses the same p-state in the hyper period. We apply a fixed-

priority scheduling method, in which higher priority tasks can preempt lower priority tasks, so the order of tasks is obtained from the priority. This scheduling method is widely used and is supported by standards like OSEK [68].

Task response time is an important metric to analyze timing in multi-rate system. It represents the time between the task is ready to execute and it finishes the execution. In fixed-priority scheduling, higher priority tasks can preempt lower priority tasks, so the *response time* of one task contains the time it is preempted by higher priority tasks besides its execution time. We use r_i to represent the response time of task τ_i and it is expressed as (3.3) where $hp(\tau_i)$ denotes the task set containing higher priority tasks than task τ_i. The first term of (3.3) represents the execution time of task τ_i and the second term denotes the preemption time of higher priority tasks.

$$r_i = \Delta t_i + \sum_{\tau_k \in hp(\tau_i)} \lceil \frac{r_i}{T_k} \rceil \Delta t_k \tag{3.3}$$

The timing constraints for multi-rate system is that the response time of any task should be no greater than its period: $\forall i : r_i \leq T_i$. Every task must finish its execution before its next activation. In our real-time system model, we use *Rate Monotonic scheduling*, which is a common preemptive fixed-priority scheduling method. As an illustration, a real-time system model for single- and multi-rate is shown in Fig. 3.1 where task 2 has the highest priority due to its smallest period and task 3 has been preempted two times by task 2 within its response time. Further, we use *core utilization* to measure the percentage of total execution time in the processor, which has significant impacts on the lifetime improvement by the different slack time from DVFS. The core utilization u is expressed as $u = \sum_i \frac{\Delta t_i}{T_i}$, where Δt_i is the execution time for task i and T_i is the period of the task.

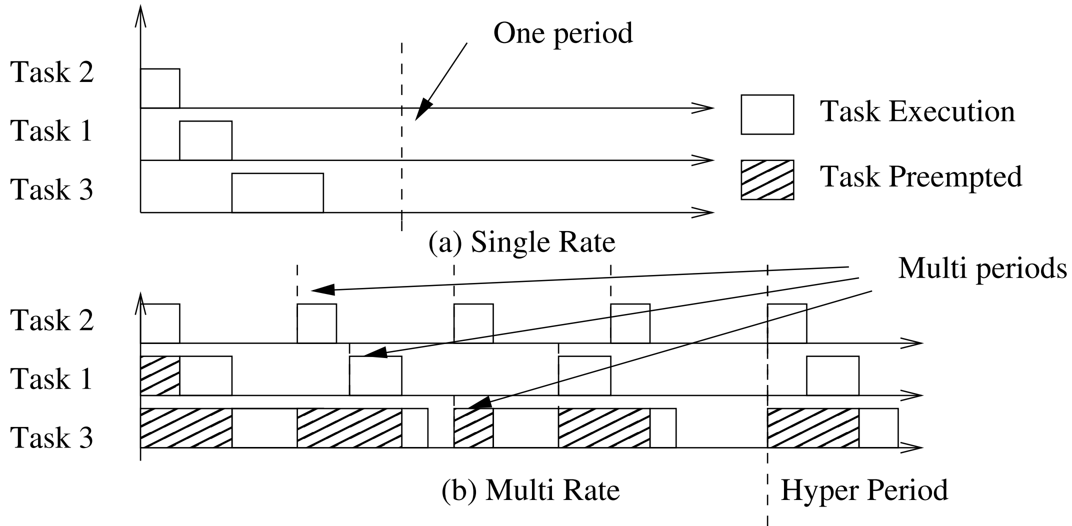


Figure 3.1: Single-rate and multi-rate task scheduling models

3.3 Proposed new lifetime optimization method

In this section, we introduce the proposed lifetime optimization method for real-time embedded systems considering EM effects. The goal is to optimize the EM-based lifetime of the embedded system subject to the timing and performance constraints. To effectively manage the lifetime of the chips due to EM effect, we still apply the dynamic voltage and frequency scaling (DVFS), which is implemented by performance-state (p-state) selections for a processor [69]. We assume that a task scheduling for an embedded system has been finished already with Rate Monotonic as described in Section 3.2.

First, we formulate continuous nonlinear objective function and use a heuristic optimization solution, this we can find a feasible solution. However, due to the limitation of heuristic global optimization solution, which can hardly find the optimal solutions, we show a linearization of the formulation to solve with linear solution easily. We build continuous

nonlinear and discrete linear objective functions and use Simulated Annealing (SA) and Mixed Integer Linear Programming (MILP), respectively.

To solve the resulting optimization problem, we first formulate the problem as continuous constrained nonlinear programming problem and solve it by the simulated annealing method. However, due to the limitation of SA, global optimality is not guaranteed. Second, we reformulate the same problem into a mixed-integer linear programming (MILP) problem and find the optimal solutions with regard to the same assumptions of the above nonlinear programming problem. We also compare the two solving solutions.

3.3.1 The new lifetime optimization flow

In this subsection, we first explain the new optimization flow and its major steps. The whole algorithm flow is shown in Algorithm 1. First, we start with a single- or multi-rate task set in either single-rate or multi-rate model. The tasks have non-optimized initial p-state, which has a pair of highest operating frequency and voltage. The temperature values from the profiled power and execution time can be measured by running HotSpot [70] with respect to every possible p-state selection for each task.

Once we have all the power and temperature information, we compute the MTTF for each p-state for the task based on the after-mentioned EM simulator discussed in the previous section. As a result, for each p-state p_j of a task i, the execution time Δ_{t_i} is the function of its $MTTF_i = MTTF(p_j)$ under the p-state p_j. We then build a continuous function of $\Delta_{t_i}(MTTF_i)$ using the response surface method (RSM) [71]. We may use up to 3rd order polynomials for our RSM method to handle the nonlinearity. The function is important for the lifetime optimization as shown later. Then, we solve a constrained

nonlinear optimization problem to find the best p-state for each task. To show the quality of this original solution (P1 in Algorithm 1), we use an alternative method (P2 in Algorithm 1). We linearize the objective and the constraints of the nonlinear problem as a MILP problem. We solve a discrete linear optimization problem. Then, we compare two solutions in the experiment.

Lifetime optimization for real-time embedded systems

Input: A task set with execution time (Δt_i), power, and available p-states (p_j)

Output: Optimized p-state selection for a task set (Two methods, P1 with the objective function (3.4) and P2 with the objective function (3.23))

Compute scaling voltage and frequency for each task with every p-state;

Compute a period T_i or hyper period T_{hyper} for single- or multi-rate system. Compute scaled power and measured temperature for each task with every p-state p_j;

With given power and temperature, $MTTF_i = MTTF(p_j)$ can be calculated for each p-state p_j;

Calculate the task execution time Δt_i and MTTF (lifetime) for each task for every p-state ;

if *Simulated Annealing (P1)* **then**

  (P1) Build continuous function of $\Delta_{t_i}(MTTF_i)$ for each task i. ;

  (P1) Perform the MTTF optimization as shown in (3.4) with timing constraint to find best p-state for each task. ;

  (P1) Output each selected p-state for each task as the first solution. ;

end

else if *MILP (P2)* **then**

  (P2) Linearize the continuous function in (3.4) with timing constraint. ;

  (P2) Perform P-state optimization as shown in (3.23) with linearized timing constraints to find best p-state for each task. ;

  (P2) Output each selected p-state for each task as the second solution. ;

end

Algorithm 1: EM-induced lifetime optimization algorithm flow

For a multi-rate system, one important problem is that preemption can lead to interruption of a task execution. Fig. 3.2 shows an example of the mechanism of preemption in which task 1 has been preempted by task 2. Since each task has its power (thus temperature profile), does such re-ordered task execution will affect the EM-induced relia-

bility? It turns out that such task re-ordering or task preemption has marginal impacts on the EM-induced reliability. Table 3.1 shows the results for the MTTF of two executions: one for non-preemption and one for preemption. As we can see that the MTTFs for both cases are almost the same. The reason is that the temperature of the each task execution is mainly determined by its power, its transient thermal effects between task transition are not significant. Thus, the lifetime formulation in (3.2) remains the same for multi-rate system.

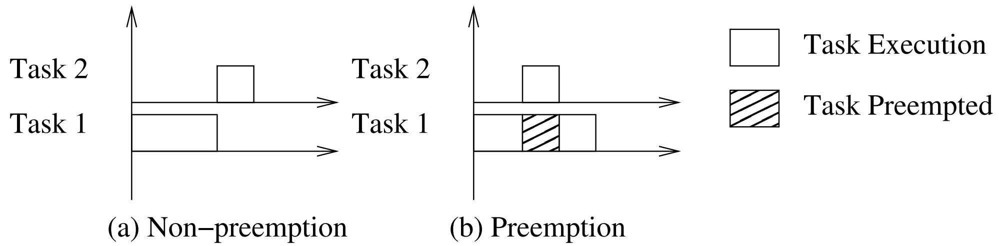


Figure 3.2: Multi-rate preemption

Table 3.1: A preliminary measurement for preemptive effect on lifetime

Non-preemption	**Task1**	**Task2**	**Lifetime**
$MTTF_i$	0.47	83	0.71
Preemption	**Task1 (Divided)**	**Task2**	**Lifetime**
$MTTF_i$	0.47,0.47	84.3	0.71

3.3.2 Formulation one: continuous constrained nonlinear optimization

In this subsection, we first show the resulting problem formulation for continuous constrained nonlinear optimization. Specifically, the whole optimization can be formulated in (3.4).

$$\text{Max } Lifetime(m) = \frac{1}{(\sum_{n}^{i=1}(\Delta t_i \frac{1}{MTTF_i} \frac{T_{hyper}}{T_i}))/T_{hyper}}$$

Subject to:

$$\begin{aligned} & r_i = \Delta t_i + \sum_{k \in hp(i)} \left\lceil \frac{r_i}{T_k} \right\rceil \Delta t_k \leq T_i \\ & MTTF_{i,l} \leq MTTF_i \leq MTTF_{i,u} \end{aligned} \tag{3.4}$$

where $m = [MTTF_1, MTTF_2, \ldots, MTTF_n]^T$, which is the variable vector, i is the task id, Δt_i is the execution time for task i , n is the total number of task, $MTTF_i$ is the segment MTTF for task i r_i is the response time of task i, $hp(i)$ is the task set containing the higher priority tasks than the current task i. T_i is the period of task i, which is a deadline. T_{hyper} is the hyper period for all tasks. At the single-rate T_{hyper} is equal to T_i. T_{total} is the total execution time of all tasks. $MTTF_{i,l}$ is the minimum bound of MTTF for task i and $MTTF_{i,u}$ is the maximum bound of MTTF for task i.

We note that each frequency in the DVFS settings is tighten to each p-state. For instance, we used five p-sate p_j = {(1.6Ghz, 1.484V),(1.4Ghz, 1.409V), (1.22Ghz, 1.339V), (1.07Ghz, 1.272V), (930Mhz, 1.208V)} were chosen from ACPI standard and Enhanced Intel Speedstep Technology [69, 72]. As a result, once we know the $MTTF_i$ for a task, its p-state and the associated frequency and execution time can be determined by finding the p-state giving the closest MTTF to the $MTTF_i$.

To solve the constrained nonlinear optimization problem, the simulated annealing method (SA) is applied. We use the Matlab's global optimization toolbox, which provides the simulated annealing function code based on adaptive simulated annealing (ASA) [73]. The simulated annealing begins with an initial $MTTF_i$ obtained by a median of $MTTF_{i,l}$ and $MTTF_{i,u}$ for each task i. The algorithm allows a large number of moves to gradually improve MTTF. The step length equals to the current temperature, and the moving direction is uniformly random. The problem type is set as *bound constrained* with a set of bounds, which are $MTTF_{i,l}$ and $MTTF_{i,u}$ for each task i. The regression coefficients in our RSM model can be parameterized by the variables ($MTTF_i$) of our objective function and act as constants during the optimization. The temperature will be lowered by 0.95^k at each iteration, where k is the annealing parameter, which is the same as the iteration number before the annealing. The stopping criterion is set to 10^{-6}, where the iteration stops when the average lifetime variation in the objective function is smaller than this tolerance. The maximum number of evaluation is set to 3000. Once a solution m is found, the corresponding p-state p_j for each $MTTF_i$ for each task is calculated by finding the shortest-distance between $MTTF_i(p_j)$ and computed $MTTF_i$.

3.3.3 Formulation two: mixed-integer linear programming

Since global optimality is not guaranteed in the first problem formulation, a better solution is desired. In this subsection, we try to solve the same optimization problem using an alternative method. The idea is to linearize the nonlinear objective function in (3.4). The resulting problem becomes a constrained linear optimization problem and can be solved by the mixed-integer linear programming (MILP) method, which can lead

to the optimal solutions with regard to the proposed EM-induced reliability model and assumptions. However, MILP will be more expensive to solve for large-scale problem. Nevertheless, it can be used to show the quality of the solutions obtained from the previous constrained nonlinear optimization method.

Multi-rate System: We first calculate the task execution time and MTTF for each task under every p-state, and store the results in the corresponding look-up table. $\Delta t_{\tau_i,p_k}$ represents the execution time of task τ_i under p-state p_k. $MTTF_{\tau_i,p_k}$ represents the MTTF of task τ_i under p-state p_k. The output p-state selection can be denoted by the boolean variable a_{τ_i,p_k}, which equals to 1 if p-state p_k is selected for task τ_i and 0 otherwise. Since one task can only choose one p-state, τ_i, $\sum_k a_{\tau_i,p_k} = 1$ for each task. With the boolean variable, the execution time c_{τ_i} of task τ_i can be formulated as (3.5) and the MTTF m_{τ_i} of task τ_i can be formulated as (3.6).

$$c_{\tau_i} = \sum_k a_{\tau_i,p_k} \cdot \Delta t_{\tau_i,p_k} \tag{3.5}$$

$$m_{\tau_i} = \sum_k a_{\tau_i,p_k} \cdot MTTF_{\tau_i,p_k} \tag{3.6}$$

Then, the *lifetime* in (3.4) can be re-formulated as follows:

$$\text{max: } lifetime(m_\tau) = \frac{T_{hyper}}{\sum\limits_{\tau_i \in \mathcal{T}} \frac{c_{\tau_i}}{m_{\tau_i}} \frac{T_{hyper}}{T_{\tau_i}}} = \frac{1}{\sum\limits_{\tau_i \in \mathcal{T}} \frac{c_{\tau_i}}{m_{\tau_i}} \frac{1}{T_{\tau_i}}} \tag{3.7}$$

The optimal solution remains the same if we minimize the inverse of lifetime $lifetime^{-1}$, and we will prove $lifetime^{-1}$ can be further linearized into (3.8). The intuition is that both c_{τ_i} and m_{τ_i} are decided by the same variable a_{τ_i,p_k}.

$$\text{min: } lifetime^{-1}(a_{\tau,p}) = \sum_{\tau_i \in \mathcal{T}} \sum_k a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k} \cdot T_{\tau_i}} \tag{3.8}$$

$$\frac{c_{\tau_i}}{m_{\tau_i}} = \sum_k a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k}} \tag{3.9}$$

To show that such a linearization is valid, we need to prove that (3.9) is true.

Proof. As we discussed, a_{τ_i,p_k} is the boolean variable and $\sum_k a_{\tau_i,p_k} = 1$, so τ_i has only one p-state. If p_γ is the possible p-state, a_{τ_i,p_γ} is 1, all other a for task τ_i is 0. So,

$$\begin{aligned} \text{LHS} = \frac{c_{\tau_i}}{m_{\tau_i}} &= \frac{\sum_k a_{\tau_i,p_k} \cdot \Delta t_{\tau_i,p_k}}{\sum_k a_{\tau_i,p_k} \cdot MTTF_{\tau_i,p_k}} \\ &= \frac{a_{\tau_i,p_\gamma} \Delta t_{\tau_i,p_\gamma}}{a_{\tau_i,p_\gamma} \Delta MTTF_{\tau_i,p_\gamma}} = \frac{\Delta t_{\tau_i,p_\gamma}}{\Delta MTTF_{\tau_i,p_\gamma}} \end{aligned} \tag{3.10}$$

$$\begin{aligned} \text{RHS} = \sum_k a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k}} &= a_{\tau_i,p_\gamma} \frac{\Delta t_{\tau_i,p_\gamma}}{MTTF_{\tau_i,p_\gamma}} \\ &= \frac{\Delta t_{\tau_i,p_\gamma}}{\Delta MTTF_{\tau_i,p_\gamma}} \end{aligned} \tag{3.11}$$

Thus, LHS equals to RHS, and we proved that (3.9) was true. □

Therefore, if we plug Equation (3.9) into Equation(3.7), we can obtain $lifetime^{-1}$ as follows:

$$lifetime^{-1}(a_{\tau,p}) = \sum_{\tau_i \in \mathcal{T}} \frac{c_{\tau_i}}{m_{\tau_i}} \frac{1}{T_{\tau_i}} = \sum_{\tau_i \in \mathcal{T}} \sum_k a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k} \cdot T_{\tau_i}} \tag{3.12}$$

Equation (3.8) is proved. Therefore, the resulting optimizing problem is shown below. The constraint (3.14) ensures that every task chooses only one p-state. We treat the deadline of each task as its period and this leads to constraint (3.15).

$$\text{min: } lifetime^{-1}(a_{\tau,p}) = \sum_{\tau_i \in \mathcal{T}} \sum_{k} a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k} \cdot T_{\tau_i}} \tag{3.13}$$

subject to:

$$\forall i\colon \sum_{k} a_{\tau_i,p_k} = 1 \tag{3.14}$$

$$\forall i\colon r_{\tau_i} = c_{\tau_i} + \sum_{\tau_j \in hp(\tau_i)} \lceil \frac{r_{\tau_i}}{T_{\tau_j}} \rceil c_{\tau_j} \leq T_{\tau_i} \tag{3.15}$$

However, the constraint (3.15) is still not linear. We first plug in Equation (3.5), and the constraint becomes

$$\forall i\colon r_{\tau_i} = \sum_{k} a_{\tau_i,p_k} \Delta t_{\tau_i,p_k} + \sum_{\tau_j \in hp(\tau_i)} \sum_{k} \lceil \frac{r_{\tau_i}}{T_{\tau_j}} \rceil a_{\tau_j,p_k} \Delta t_{\tau_j,p_k} \leq T_{\tau_i} \tag{3.16}$$

We can observe the nonlinearity comes from the ceiling function. Therefore, we use integer variable x_{τ_i,τ_j} to represent the ceiling function $\lceil r_{\tau_i}/T_{\tau_j} \rceil$ with a newly added constraint $\forall i,j\colon 0 \leq x_{\tau_i,\tau_j} - r_{\tau_i}/T_{\tau_j} < 1$. Constraint (3.15) further becomes

$$\forall i\colon r_{\tau_i} = \sum_{k} a_{\tau_i,p_k} \Delta t_{\tau_i,p_k} + \sum_{\tau_j \in hp(\tau_i)} \sum_{k} x_{\tau_i,\tau_j} a_{\tau_j,p_k} \Delta t_{\tau_j,p_k} \leq T_{\tau_i} \tag{3.17}$$

However, the nonlinearity still exists as an integer variable is multiplied by a binary variable $x_{\tau_i,\tau_j} * a_{\tau_j,p_k}$. We introduce a non-negative integer variable $\delta_{\tau_i,\tau_j,p_k}$ to represent $x_{\tau_i,\tau_j} * a_{\tau_j,p_k}$ and use 'big M' method to linearize the multiplication of an integer variable and a binary variable as follows.

$$\forall i,j,k\colon\ x_{\tau_i,\tau_j} - M*(1-a_{\tau_j,p_k}) \le \delta_{\tau_i,\tau_j,p_k} \tag{3.18}$$

$$\forall i,j,k\colon\ \delta_{\tau_i,\tau_j,p_k} \le x_{\tau_i,\tau_j} \tag{3.19}$$

$$\forall i,j,k\colon\ 0 \le \delta_{\tau_i,\tau_j,p_k} \le M*a_{\tau_j,p_k} \tag{3.20}$$

The 'big M' is a large integer M in a set of linear inequalities to represent the equation $\delta_{\tau_i,\tau_j,p_k} = x_{\tau_i,\tau_j} * a_{\tau_j,p_k}$. Simply put, we want to guarantee that if $a_{\tau_j,p_k} = 1$, $\delta_{\tau_i,\tau_j,p_k} = x_{\tau_i,\tau_j}$, and if $a_{\tau_j,p_k} = 0$, $\delta_{\tau_i,\tau_j,p_k} = 0$. In inequality (3.18), if $a_{\tau_j,p_k} = 1$, $x_{\tau_i,\tau_j} \le \delta_{\tau_i,\tau_j,p_k}$. Combined with (3.19) $x_{\tau_i,\tau_j} \ge \delta_{\tau_i,\tau_j,p_k}$, we can guarantee $\delta_{\tau_i,\tau_j,p_k} = x_{\tau_i,\tau_j}$. In inequality (3.18), if $a_{\tau_j,p_k} = 0$, $x_{\tau_i,\tau_j} - M \le \delta_{\tau_i,\tau_j,p_k}$. Combined with constraints (3.20) $0 \le \delta_{\tau_i,\tau_j,p_k} \le M*0$, we can guarantee that $\delta_{\tau_i,\tau_j,p_k} = 0$. Therefore, constraints (3.18) to (3.20) are equivalent to equation $\delta_{\tau_i,\tau_j,p_k} = x_{\tau_i,\tau_j} * a_{\tau_j,p_k}$, and thus constraint (3.15) can be further written as

$$\forall i\colon\ r_{\tau_i} = \sum_k a_{\tau_i,p_k}\Delta t_{\tau_i,p_k} + \sum_{\tau_j \in hp(\tau_i)} \sum_k \delta_{\tau_i,\tau_j,p_k}\Delta t_{\tau_j,p_k} \le T_{\tau_i} \tag{3.21}$$

with constraints (3.18) to (3.20) added. One last thing is to represent the set of higher priority tasks. Constant P_{τ_i,τ_j} represents the priority between task τ_i and τ_j. Constant P_{τ_i,τ_j} equals to 1 if τ_j has higher priority than τ_i and 0 otherwise. Finally, constraint (3.15) becomes a linear equation with the above constraints added.

$$\forall i\colon\ r_{\tau_i} = \sum_k a_{\tau_i,p_k}\Delta t_{\tau_i,p_k} + \sum_{\tau_j \in \mathcal{T}} \sum_k \delta_{\tau_i,\tau_j,p_k} P_{\tau_i,\tau_j}\Delta t_{\tau_j,p_k} \le T_{\tau_i} \tag{3.22}$$

After all these linearization steps, we have the following linear objective function with linear timing constraints, which can be solved by the mixed-integer linear programming (MILP) method:

$$\text{minimize: } lifetime^{-1}(a_{\tau,p}) = \sum_{\tau_i \in \mathcal{T}} \sum_{k} a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k} \cdot T_{\tau_i}} \tag{3.23}$$

subject to:

$$\forall i\colon \sum_{k} a_{\tau_i,p_k} = 1 \tag{3.24}$$

$$\forall i,j\colon 0 \leq x_{\tau_i,\tau_j} - r_{\tau_i}/T_{\tau_j} < 1 \tag{3.25}$$

$$\forall i,j,k\colon x_{\tau_i,\tau_j} - M * (1 - a_{\tau_j,p_k}) \leq \delta_{\tau_i,\tau_j,p_k} \tag{3.26}$$

$$\forall i,j,k\colon \delta_{\tau_i,\tau_j,p_k} \leq x_{\tau_i,\tau_j} \tag{3.27}$$

$$\forall i,j,k\colon 0 \leq \delta_{\tau_i,\tau_j,p_k} \leq M * a_{\tau_j,p_k} \tag{3.28}$$

$$\forall i\colon r_{\tau_i} = \sum_{k} a_{\tau_i,p_k} * \Delta t_{\tau_i,p_k} + \sum_{\tau_j \in \mathcal{T}} \sum_{k} \delta_{\tau_i,\tau_j,p_k} * P_{\tau_i,\tau_j} * \Delta t_{\tau_j,p_k} \leq T_{\tau_i} \tag{3.29}$$

Note that, The linearization techniques in this work do not change optimality as shown above and no approximation is conducted during the derivation. Therefore, we can guarantee our solution is exactly optimal with regard to our lifetime model.

Single-rate System: As discussed in Section 3, the timing constraints of single-rate system is simply $\sum_i \Delta t_{\tau_i} < T$. So, there is no ceiling function for linearization, and the optimization is much simpler. For single-rate system, we can simplify the problem as follows to make the execution much faster.

$$\text{minimize: } lifetime^{-1}(a_{\tau,p}) = \sum_{\tau_i \in \mathcal{T}} \sum_{k} a_{\tau_i,p_k} \frac{\Delta t_{\tau_i,p_k}}{MTTF_{\tau_i,p_k} \cdot T_{\tau_i}} \tag{3.30}$$

subject to:

$$\forall i\colon \sum_{k} a_{\tau_i,p_k} = 1 \tag{3.31}$$

$$\sum_{\tau_i \in \mathcal{T}} \sum_{k} a_{\tau_i,p_k} * \Delta t_{\tau_i,p_k} \leq T \tag{3.32}$$

We note that in this work, we only consider the EM effects reliability. But there are many other reliability effects for back end of the lines such as TDDB (time dependent dielectric breakdown) and TC (thermal cycling). We want to stress that the proposed optimization method is orthogonal to other reliability effects as long as they can be modeled properly in terms of system level parameters such as temperature and p-state. Our work can be viewed as a demonstration of considering long-term reliability and performance for embedded systems and it can be easily extended to deal with other reliability effects. Actually considering TDDB effect is a trivial as TDDB induced time to failure based on TDDB models such as $1/E$ and $\sqrt{E}$ models [74] shows similar trend with respect to the power/temperature with respect to the temperature. We should expect to see the similar tradeoff between lifetime and the performance. For thermal cycling, the effects may contradict the EM effects as number of cycling and temperature ranges will make differences as shown in [75]. On the other hand, these effects are more significant for package and die interface (solder joints). Experiments show that very large thermal cycles (more than 140 degrees Celsius) are required to cause any damage to the silicon substrate and interconnects [10]. As a result, it seems less concern for normal chip working conditions as we assumed in this work.

3.4 Numerical results and discussions

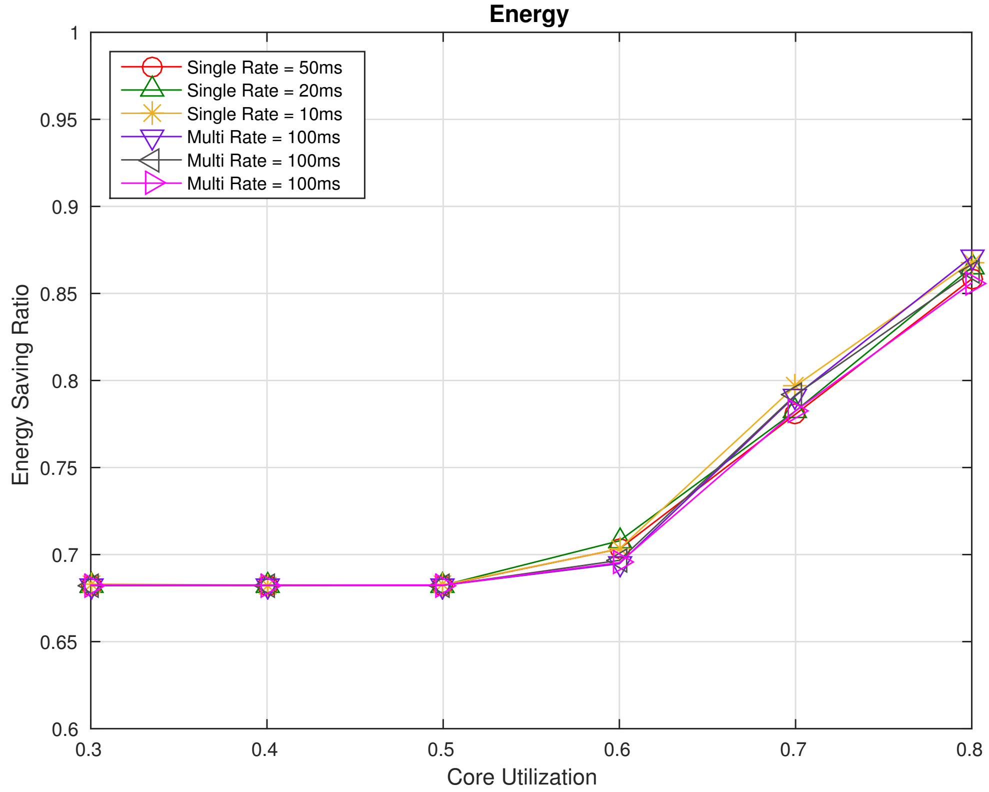


Figure 3.3: Core utilization effect - energy savings

We first show the experimental setups for evaluating the proposed optimization algorithms. Then we show the reliability and performance results on a number of real-time embedded systems. Finally, we study the results from the two optimization methods to analyze the solution quality of the two proposed methods. We also provide some insight look at the resulting trade-off between performance and lifetime.

Table 3.2: Optimization method evaluation for low core utilization single-rate task (SA and MILP)

Non optimized	6 Tasks at 40% core utilization at single-rate					
i (task id)	1	2	3	4	5	6
P_k (p-state)	1	1	1	1	1	1
Δt_i(ms)	3.70	5.82	2.43	5.10	0.92	1.99
$Period_i$ (ms)	50	50	50	50	50	50
MTTF (year)	0.47	0.47	0.46	0.47	0.47	0.47
Energy (Wh)	0.11	0.17	0.07	0.15	0.02	0.05
Total Energy	**0.59**		**Lifetime**		**1.19**	
SA Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	5	5	5	5	5	5
Δt_i (ms)	6.32	9.93	4.15	8.69	1.57	3.40
$Period_i$ (ms)	50	50	50	50	50	50
$MTTF_i$ (year)	44.8	44.7	48	44.7	50	48
$Energy_i$ (Wh)	0.07	0.11	0.04	0.10	0.01	0.04
Total Energy	**0.40**		**Lifetime**		**66.97**	
MILP Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	5	5	5	5	5	5
Δt_i (ms)	6.32	9.93	4.15	8.69	1.57	3.40
$Period_i$ (ms)	50	50	50	50	50	50
$MTTF_i$ (year)	44.8	44.7	48	44.7	50	48
$Energy_i$ (Wh)	0.07	0.11	0.04	0.10	0.01	0.04
Total Energy	**0.40**		**Lifetime**		**66.97**	

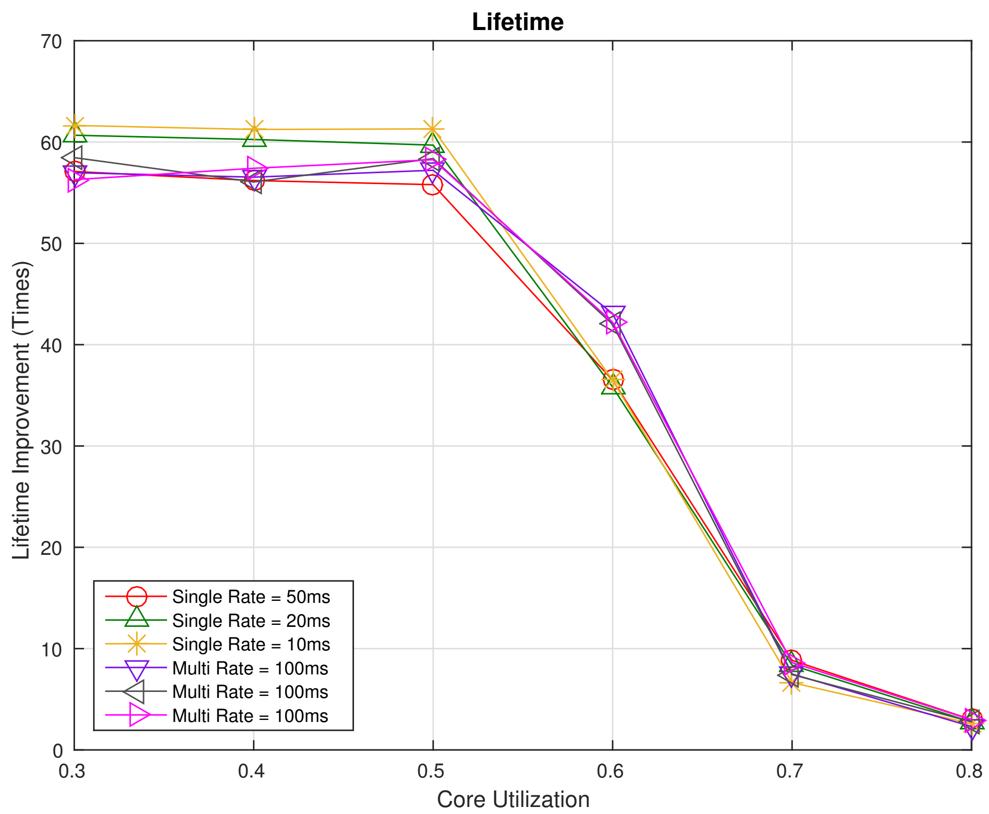


Figure 3.4: Core utilization effect - lifetime improvement

Table 3.3: Optimization method evaluation for high core utilization single-rate task (SA and MILP)

Non optimized	6 Tasks at 80% core utilization at single-rate					
i (task id)	1	2	3	4	5	6
P_k (p-state)	1	1	1	1	1	1
Δt_i(ms)	6.81	5.49	8.37	5.25	7.45	6.51
$Period_i$ (ms)	50	50	50	50	50	50
$MTTF_i$ (year)	0.47	0.47	0.46	0.47	0.47	0.47
$Energy_i$ (Wh)	0.20	0.16	0.25	0.15	0.22	0.19
Total Energy	**1.19**		**Lifetime**		**0.59**	
SA Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	2	3	3	3	2	3
Δt_i (ms)	7.79	7.18	10.93	6.86	8.52	8.50
$Period_i$ (ms)	50	50	50	50	50	50
$MTTF_i$ (year)	0.95	3.06	3.06	3.06	0.95	3.06
$Energy_i$ (Wh)	0.18	0.13	0.20	0.13	0.20	0.16
Total Energy	**1.027**		**Lifetime**		**1.78**	
MILP Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	2	3	3	3	3	2
Δt_i (ms)	7.79	7.18	10.93	6.86	9.74	7.44
$Period_i$ (ms)	50	50	50	50	50	50
$MTTF_i$ (year)	0.95	3.06	3.06	3.06	3.06	0.954
$Energy_i$ (Wh)	0.18	0.13	0.20	0.13	0.18	0.17
Total Energy	**1.025**		**Lifetime**		**1.83**	

Table 3.4: Optimization method evaluation for low core utilization multi-rate task (SA and MILP)

Non optimized	6 Tasks at 40% core utilization at multi-rate					
task id i	1	2	3	4	5	6
P_k (p-state)	1	1	1	1	1	1
Δt_i (ms)	6.93	4.02	4.19	1.91	1.64	3.03
$Period_i$ (ms)	100	100	50	20	20	100
$MTTF_i$ (year)	0.47	0.47	0.46	0.47	0.47	0.47
$Energy_i$ (Wh)	0.20	0.12	0.25	0.28	0.24	0.09
Total Energy	**1.20**		**Lifetime**		**1.18**	
SA Optimized						
task id i	1	2	3	4	5	6
P_k (p-state)	5	5	5	5	5	5
Δt_i (ms)	11.8	6.58	7.15	3.27	2.79	5.18
$MTTF_i$ (year)	44.7	44.8	44.8	48.2	49	48
$Energy_i$ (Wh)	0.14	0.08	0.17	0.19	0.16	0.06
Total Energy	**0.82**		**Lifetime**		**68.02**	
MILP Optimized						
task id i	1	2	3	4	5	6
P_k (p-state)	5	5	5	5	5	5
Δt_i (ms)	11.8	6.58	7.15	3.27	2.79	5.18
$MTTF_i$ (year)	44.7	44.8	44.8	48.2	49	48
$Energy_i$ (Wh)	0.14	0.08	0.17	0.19	0.16	0.06
Total Energy	**0.82**		**Lifetime**		**68.02**	

Table 3.5: Optimization method evaluation for high core utilization multi-rate task (SA and MILP)

Non optimized	6 Tasks at 80% core utilization at multi-rate					
i (task id)	1	2	3	4	5	6
P_k (p-state)	1	1	1	1	1	1
Δt_i (ms)	9.33	7.25	16.38	13.94	10.05	3.30
$Period_i$ (ms)	100	50	100	100	100	20
MTTF (year)	0.47	0.47	0.46	0.47	0.47	0.47
Energy (Wh)	0.27	0.43	0.49	0.41	0.30	0.49
Total Energy	**2.42**		**Lifetime**		**0.58**	
SA Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	3	4	3	1	2	2
Δt_i (ms)	12.18	10.82	21.39	13.94	11.49	3.77
MTTF (year)	3.06	15.2	3.06	0.47	0.95	0.95
Energy (Wh)	0.23	0.16	0.40	0.41	0.27	0.09
Total Energy	**2.10**		**Lifetime**		**1.36**	
MILP Optimized						
i (task id)	1	2	3	4	5	6
P_k (p-state)	2	2	3	3	2	3
Δt_i (ms)	10.66	8.28	21.39	18.21	11.49	4.31
MTTF (year)	0.95	0.95	3.06	3.06	0.95	3.06
Energy (Wh)	0.25	0.19	0.40	0.34	0.27	0.08
Total Energy	**2.09**		**Lifetime**		**1.65**	

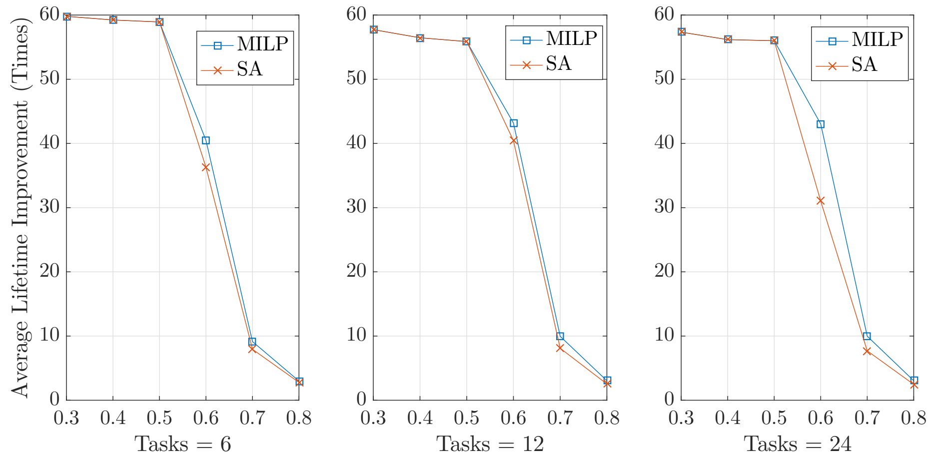


Figure 3.5: The comparisons of simulated annealing and mixed-integer linear programming methods for the lifetime optimization for single-rate with 6, 12, and 24 tasks per one task set under different core utilizations (0.3 to 0.8 in x-axis)

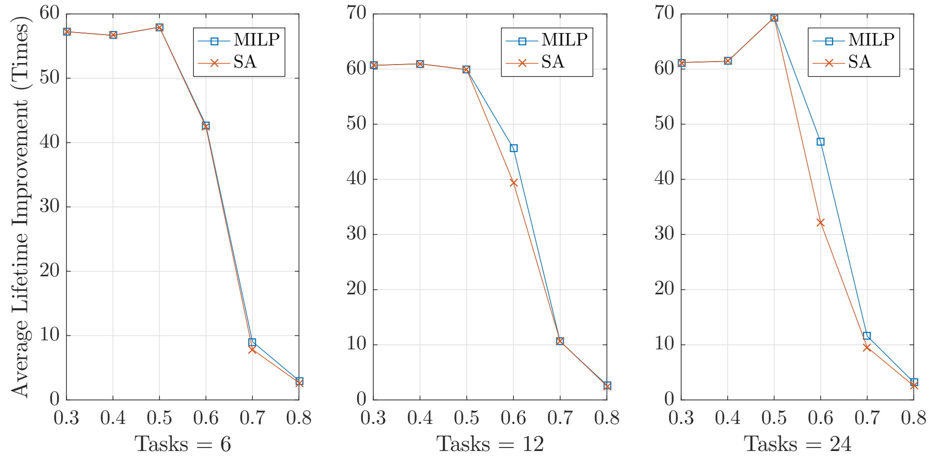


Figure 3.6: The comparisons of simulated annealing and mixed-integer linear programming methods for the lifetime optimization for multi-rate with 6, 12, and 24 tasks per one task set under different core utilizations (0.3 to 0.8 in x-axis)

3.4.1 Experimental setup

The proposed new lifetime task optimization method has been implemented in Matlab and C++. We applied the recently proposed physics-based EM model and analysis method for our EM analysis [57]. We use simple mesh-structured power grid for embedded

processor. We used the interconnect material is copper and the power grid fails when the largest voltage drop exceeds 10% of V_{DD}. Hotspot [70] is used for the temperature modeling. We used 60 different task sets and each taskset has a different number of tasks per one task set (6 tasks, 12 tasks, and 24 tasks). To generate task sets, we implement a random real-time task generator based on the core utilization factor (different ratio from 0.3 to 0.8). The core utilization is defined as the percentage of total execution time in the processor for the tasks in the real-time system for single- and multi-rate. We notice that core utilization is not considered as a constraint on our work, implicitly, instead, it was considered implicitly as it is a function of the task execution times and its given deadlines, which are considered as constraints. For single rate task sets, the period is set to be 100ms. For multi-rate task sets, we randomly choose task period among 100ms, 50ms, 20ms, and 10ms. The hyper period for both cases is 100ms. Based on the period, we randomly generate task execution time between 0 to its period considering target core utilization. All tasks are independent with each other. All benchmarks and Matlab environment are running on a 4-core 3.0Ghz Xeon server with 16GB RAM running Linux. Five p-states $p_j = \{$(1.6Ghz, 1.484V),(1.4Ghz, 1.409V), (1.22Ghz, 1.339V), (1.07Ghz, 1.272V), (930Mhz, 1.208V)$\}$ are chosen from ACPI standard and Enhanced Intel Speedstep Technology [69, 72]. Lastly, IBM ILOG CPLEX optimizer [76] is used for mixed-integer linear programming.

3.4.2 Evaluation of proposed lifetime optimization

First, we evaluate our lifetime optimization method (see Section 3.3) by comparing its lifetime and energy with non-optimized p-state tasks to the optimized p-state tasks. Here, we use 6 tasks per one task set and 36 task sets. In the evaluation of our lifetime

optimization (SA method), we consider four different task sets (low and high core utilization, and single and multi-rates) for comparison. Table II and III summarize the results for 40% and 80% core utilization of single-rate real-time tasks. As we can see, for the low core utilization tasks, the optimization method finds the lowest-energy p-state solution for each task and total energy consumption decreases about 67% as each task p-state is selected as 5, which is the lowest-energy DVFS for each task execution. With SA optimization method, the lifetime is improved to 66.97 years from 1.19 years. However, for the high utilization case, lifetime improvement will be limited as shown in the 80% utilization case. In this case, the most of the p-states are moved to the middle range (3 in this case). However, there is still 85% energy saving and 3X lifetime improvement achieved by the proposed lifetime optimization method. This indicates the significant improvement can be made for both energy and reliability from the simple task scheduling.

In addition to single-rate task sets, we also show results for multi-rate task sets summarized in Table IV and V. Again, depending on the core utilization, the energy and lifetime can be both improved and the improvement can be significant.

3.4.3 Core utilization effects and trade-off on energy and lifetime

As we can see, the core utilization factor can have significant impacts on the final results. Fig. 3.3 and Fig. 3.4 show the core utilization versus energy and lifetime for the SA method. The experiments are simulated with single rate tasks at 50 ms, 20 ms, and 10 ms periods, and three random multi-rate tasks at 100 ms hyper period. In the low core utilization, the system can run the task under higher p-state as the core utilization is the ratio of the task execution time and its period. In the higher utilization, energy saving

and lifetime improvements decrease as the higher utilization leads to less number of p-state selection.

3.4.4 Solution quality study and analysis

In this subsection, we compare the both optimization algorithms on a number of examples. The results are shown in Fig 3.5 and Fig 3.6, which show lifetime improvement for both single- and multi-rate real-time embedded systems under different core utilization with different tasksets solved by the MILP method and simulated annealing method respectively. To show the scalability of our proposed algorithm, we used different tasksets with 6 tasks per one task set (36 task sets), 12 tasks per one (12 task sets) and 24 tasks per one (12 task sets). MILP formulation can deliver the optimal results but at high computational costs for large-scale optimization. For the lower utilization cases, both results from both MILP and continuous simulated annealing are almost same. But, for the higher utilization and higher taskset cases, MILP show better results as expected as the optimization becomes more difficult to reach optimal solutions. These results indicate that the proposed constrained optimization can lead to near-optimal results in general.

Table 3.6: Elapsed CPU Time to solve the proposed Simulated Annealing and MILP problems (second per one taskset)

Multi-Rate Task Case	Simulated Annealing (P1)	MILP (P2)
6 tasks per one taskset (80% core utilization)	3.29 s	1.54 s
12 tasks per one taskset (80% core utilization)	6.18 s	155.40 s
24 tasks per one taskset (80% core utilization)	15.19 s	6003.98 s

We remark that it is rather difficult to fairly compare CPU times for these two optimization methods as the simulated annealing method was running on MATLAB Toolbox, whereas the MILP was running on commercial CPLEX Optimizer. For the large case, however, MILP will be very expensive as it basically uses the branch-and-bound algorithm, which involves constructions of a search tree as seen in Table 3.6. It shows MILP is more expensive with larger scale tasks and its running time was 6003.98s with 24 tasks, whereas the simulated annealing method can only take 15.19s to solve the large scale tasks. Due to the exponential growth in the size of the search trees, MILP method would quickly become computationally expensive [77]. Moreover, due to the nature of the search tree usage, it results in excessive memory usage [77]. Thus, MILP does not scale very well for large problem sizes.

3.4.5 Trade-off between performance and lifetime

Finally, we show the Pareto-like trade-off between the performance (core utilization) and lifetime obtained from the proposed lifetime optimization as shown in Fig. 3.7. We assume that the higher utilization implies more tasks can be scheduled and executed in the given time, thus it can be considered as higher performance in Fig. 3.7. For long-term reliability, such as electromigration effects, high performance will always lead to the shorter lifetime and vice versa. But, this is not true for soft error short term reliability, in which low performance/power lead to much worse reliability [78]. With low utilization, we can have more room for such performance/power and reliability trade-off. The proposed method can lead to the best (Pareto-like) trade-off.

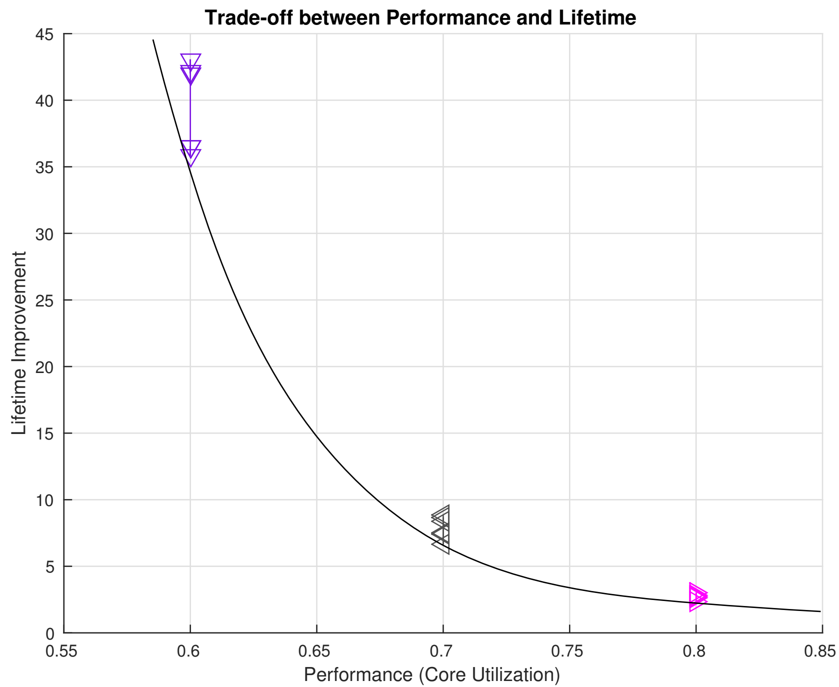


Figure 3.7: Trade-off between lifetime and performance (each triangle is different set of core utilization)

3.5 Summary

In this chapter, we have developed new lifetime task optimization techniques for real-time embedded processors considering the electromigration-induced reliability. The new approach was based on a recently proposed physics-based electromigration (EM) model for more accurate EM assessment of a power grid network at the chip level. We applied the dynamic voltage and frequency scaling (DVFS) (by selecting the performance states or p-states of the tasks to manage the power) and thus the lifetime of the processor running different tasks over their periods. We considered both single-rate and multi-rate embedded systems with preemption. We explored to problem formulations and found the corresponding solutions with different solution qualities and computational costs. Experimental re-

sults have shown that for low utilization systems, significant reliability improvement can be achieved with even smaller power consumption than existing reliability-ignore scheduling method. We also compared the results from the two formulations and showed that the solutions given by the constrained nonlinear optimization method is close to the ones given by the MILP-based method, which is considered to be an optimal solution with regard to the proposed EM-induced reliability model and assumptions.

Chapter 4

Learning-based reliability management and energy optimization for mary-core dark silicon processors

In this chapter, we develop a new energy and lifetime optimization techniques for emerging dark silicon manycore microprocessors considering both long-term reliability effects (hard errors) and transient single event upset errors (soft errors). We employ both dynamic voltage and frequency scaling (DVFS) and dark silicon core state using On/Off switching action as two control knobs. We develop on two optimization techniques for improving lifetime and reducing energy. To optimize EM-induced lifetime, we first apply the adaptive Q-learning based method, which is suitable for dynamic runtime operation as

it can provide cost-effective yet good solutions. The second lifetime optimization approach is the mixed-integer linear programming method, which typically yields better solutions but at higher computational costs.

4.1 Review of system-level EM and soft error reliability models

4.1.1 System-level EM reliability model

At the system-level EM reliability, the manycore system will run on different tasks under different p-states. Let's assume that we have a set of different time intervals Δp_k characterized by different workloads or p-state in terms of current density j_k and temperature T_k for a processor or a core. It means that $P = \sum_{k=1}^{n} \Delta p_k$ is the total execution time. Each kth workload, if it lasts till imaginary failure, provides time to failure TTF_k. Thus the failure rate at the kth workload, which last Δp_k is $\lambda_k = 1/TTF_k$. Then the average failure rate for the considered set of work loads is

$$\lambda_{avg} = \sum_{k=1}^{n} \frac{\Delta p_k}{\sum_{j=1}^{n} \Delta p_j} \lambda_k = \sum_{k=1}^{n} \frac{\Delta p_k}{P} \lambda_k \tag{4.1}$$

As a result, the expected time to failure or average lifetime of the whole processor, TTF is [67],

$$MTTF = \frac{1}{\lambda_{avg}} = \frac{1}{(\sum_{k=1}^{n} (\Delta p_k \frac{1}{MTTF_{R,k}}))/P} \tag{4.2}$$

where $MTTF_{R,k}$ is the actual MTTF under the k-th power and temperature settings for Δp_k period, assuming the chip works through n different power and temperature settings and $P = \sum_{k=1}^{n} \Delta p_k$. Each $MTTF_{R,k}$ will be computed based on the EM models in Section 2.1.

To consider a system-level EM reliability on a manycore dark silicon processor, we use the shortest lifetime among all the cores as the lifetime for all manycore processors [79]. We want to stress that the proposed techniques are orthogonal to other long-term reliability effects (such as NBTI, HCI, TDDB for devices and stress and thermal migration for interconnects). The proposed techniques are orthogonal to other long-term reliability managements as those long-term reliability effects generally behave similarly or in a similar trend under their stressing conditions in terms of voltage, current and temperature [41]. Specifically, the power and temperature typically have the same impacts on the NBTI, HCI and TDDB as those failure effects follows the Arrhenius equation for the relationship between the failure rate and temperature (which is a function of powers or energy) [80], as a result, those long-term reliability effects will become worse when temperature increases. As a result, DVFS based optimization will lead to similar trade-off between long-term reliability and soft errors.

4.1.2 Soft error reliability model considering DVFS impacts

Soft errors, or single event upset, are defined as the transient faults inside the logic or memory on a chip, and result in an incorrect system output. The soft errors can be caused by cosmic radiation, alpha particle decay, and thermal neutrons. Soft error rate (SER) is the rate in which a chip or system encounters soft errors and typically can be expressed as the number of failures in the given time. Although there is still a lack of consensus on the exact soft error rate (SER) of specific chips and systems, it is obvious that the SER per chip is practically increasing due to the increasing number of components or cores on a chip. Recently it has been reported that the dynamic voltage frequency scaling

(DVFS) method, used for energy saving, negatively affects the system reliability because the transient fault rate increases and the critical charge decreases by lowering the voltage and frequency. As a result, new exponential soft error models have been introduced to account for those effects [81, 82].

For our problem, we employ an existing exponential model considering DVFS effects on soft error rate, which assumes that the radiation-induced failure follows a Poisson distribution, so the average soft error rate can be express as terms of operating frequency f, supply voltage V_{dd}, where SER_0 is the average failure rate at the maximum frequency f_{max} and voltage V_{max}, (so, $f_{min} < f < f_{max}$, $V_{min} < V < V_{max}$) in (4.3) [81].

$$SER(f, V_{dd}) = SER_0 e^{\frac{d(f_{max}-f)}{(f_{max}-f_{min})}} \tag{4.3}$$

where d is an architecture dependent constant, which is the sensitivity of failure rate with DVFS. We also employ the previous work to model the relationship between operating frequency and supply voltage to further simplify (4.3) from [83].

$$f = \beta \frac{(V_{dd} - V_{th})^2}{V_{dd}} \tag{4.4}$$

where β is a technology-related constant, and V_{th} is the threshold voltage. By substituting (4.4) into (4.3), DVFS-aware SER equation can be derived as the function of only supply voltage V_{dd} only [82]:

$$SER(V_{dd}) = SER_0 e^{\frac{d(f_{max}-\beta V_{dd}-2V_{th}+\frac{V_{th}^2}{V_{dd}})}{f_{max}-f_{min}}} \tag{4.5}$$

System-level soft error model for dark silicon manycore processor

To estimate system-level soft error reliability, the sum of failure rate (SOFR) with architecture vulnerability factor (AVF) method has been widely accepted in the semiconductor industry [84, 85] for combining soft error rates from each core to estimate whole system-level soft error. The AVF is used to express the probability that a visible soft error will occur with given a raw error event in a core [86]. The previous study shows the SOFR model can be used to show the whole system soft error rate (SER_{SYS}) [85]:

$$SER_{sys} = SOFR = \sum_{i=0}^{m} AVF_i \times SER_i(V_{dd}) \tag{4.6}$$

where, m is the total number of cores in a processor and $SER_i(V_{dd})$ is the soft error rate for given voltage setting (V_{dd}) and AVF_i is architecture vulnerability factor for i-th core.

4.1.3 Impact of process technology on soft error reliability model

In the past, soft errors in microprocessor logic have not been greatly concerned as the number of flops/latches in a microprocessor is much fewer than the number of SRAM cells and microprocessor single event upset (SEU) rates were lower than SRAM SEU rates. After 90nm technology, however, microprocessor SEU rates are larger than SRAM SEU rates because flop protection mechanisms (machine encoding and invariant checking) are more difficult to implement than simple memory protection mechanisms (parity, error correction code) [87]. Thus, the SEU in microprocessor will be the dominant factor to system soft error rate as technology scales to smaller feature size. Table 4.1 shows that the normalized SEU rate in microprocessor reported from the real silicon data [7]. As technology scales to smaller

Table 4.1: Raw SEU Rate Per Microprocessor on different technologies [7]

Technology	**Normalized SEU rate per microprocessor (over 45nm)**
45nm	1X
32nm	1.38X
22nm	1.59X

feature size, SEU rates keep increasing for different technologies. To assess technology scaling impacts, we used different raw SER values and other parameters (different threshold voltage (V_{th}) and supply voltage (V_{dd})) in (4.5) based on the experimental data from [7, 88]. The technology scaling impact on our proposed dynamic reliability management method will be discussed in Subsection 4.3.4.

4.2 New dynamic lifetime and energy optimization methods for dark silicon

In this section, we formulate our dynamic reliability management (DRM) problem as maximizing the (EM-induced) lifetime of dark silicon manycore processors by controlling the number of active cores and the suitable performance state (p-state) subject to the performance and temperature constraints.

To improve EM-induced lifetime, We first present the Q-learning based solution to this problem. Then we re-formulate the same problem as mixed-integer linear programming (MILP) problem. Moreover, for energy saving, we re-formulate our dynamic reliability man-

agement (DRM) problem with a Q-learning method as the minimizing energy consumption considering EM-induced lifetime and soft error-induced lifetime of dark silicon manycore processors by controlling number of active cores and the suitable performance state (p-state) subject to the performance and temperature constraints.

4.2.1 Q-learning based formulation and solution for lifetime and energy optimizations

State and action determination

Q-learning [89] , a reinforcement learning method, performs the control by maximizing expected long-term rewards [90]. Q-learning can handle problems with stochastic transition without any adaptation and is a method to be able to converge close to the optimal solution of a state-action function for an arbitrary policy [91]. In our problem, the state (s) consists of the configurations of DVFS and active status (power on/off) for each core. DVFS uses performance state (p-state) which can represent operating voltage and frequency. Action (a) is defined as a state transition from one state to the another state. An action updates the learning agent's Q-value with the reward/penalty calculation in the Q-table, also known as the state-action table. Transiting an action in a state makes the agent with a reward (negative penalty) scoring that is calculated with the quantity of state-action combination (Q). Q can be defined as a set of states (S) and a set of action (A) table, $S \times A$, which is Q-table. Q-table can be updated by a Q-value function which a long-term penalty function with state and action.

Fig. 4.1 shows the proposed learning-based (Q-learning) reliability-aware lifetime/energy optimization framework (both lifetime and energy optimizations use Q-learning method). The framework consists of an environment containing the dark silicon manycore microprocessor, and the learning agent which is the Q-learning algorithm. The learning agent obtains the environment state, calculates the penalty function, and finally decides the next action.

Table 4.2 illustrates an example of state, p-state, and active-core for small 3-core dark silicon chip. In p-state, 1 is low power mode, 2 is full power mode, and 0 means the core is turned off. Clearly, state 0 is the state with a minimum number of active cores, which are in the lowest power modes and state 8 is the state with a maximum number of active core, which are in the highest power modes.

Table 4.2: An example of control states for a 3-core processor

State	p-state	active-core	**State**	p-state	active-core
0	0,0,1	off,off,on	**1**	0,0,2	off,off,on
2	0,1,1	off,on,on	**3**	0,1,2	off,on,on
4	0,2,2	off,on,on	**5**	1,1,1	on,on,on
6	1,1,2	on,on,on	**7**	1,2,2	on,on,on
8	2,2,2	on,on,on			

Q-value function and Q-learning process

In the Q-learning process, one critical issue is to define the Q-value function with penalty term. Specifically, let's formally define *State i*: $s_i = \{PS_i, CS_i)$. PS_i is the set of

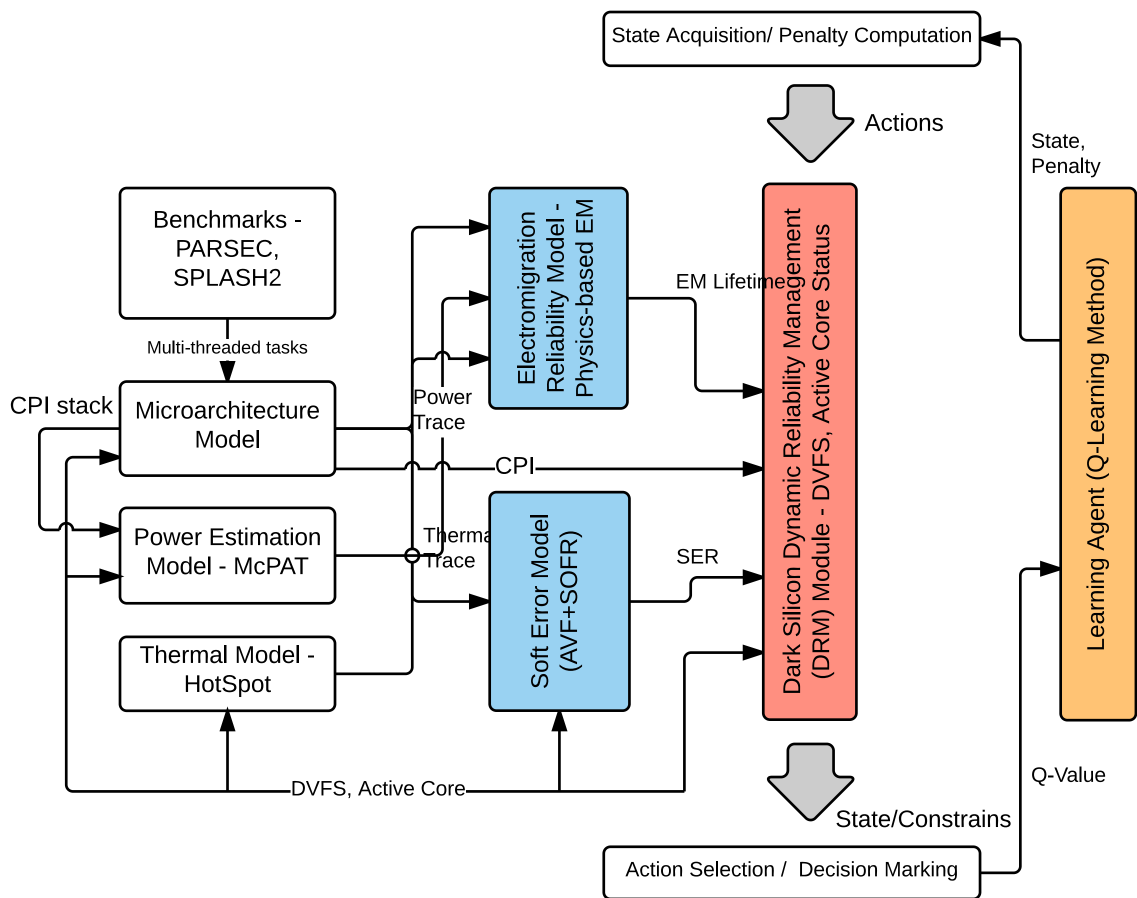


Figure 4.1: Q-Learning model with reliability-aware dark silicon framework

p-state (DVFS) statuses for all the cores. CS_i is the set of core status for all cores. Each state s_i will determine the total power of the whole chip $Power(s_i)$, worse case performances of all cores $Perf_{max}(s_i)$, the maximum temperature incurred $Temp_{max}(s_i)$, total core energy consumption in the whole chip $E_(s_i)$, and the minimum lifetime among cores $EM_{min}(s_i)$, which is defined as the EM-induced lifetime of the chip. $SER_{min}(S_i)$ is defined as system-level soft error rate (SER_{sys}) of the chip. The total core energy consumption can be obtained from $\sum_k E_k(s_i)$ which is k-th core's core energy, each core's energy can be calculated by $Power_k(s_i) \times Perf_k(s_i)$ where $Power_k(s_i)$ is average k-th core's power and $Perf_k(s_i)$ is each k-th core's performance. An action $a_{i,j}$, can be viewed as the transition from state i to state j. Then the penalty function Q determines a penalty and a new state which is related to the previous state and selected action. Q-value is updated at every step Δt.

$$\begin{aligned} Q^{t+1}(s(t), a(t)) = Q^t(s(t), a(t)) + \\ \alpha(t) \times \Big(PT(t+1) + \gamma \min_a (\forall Q^t(s(t+1), a)) \Big) \\ - \alpha(t) \times Q^t(s(t), a(t)) \end{aligned} \tag{4.7}$$

where $\alpha(t)$ is learning rate between 0 and 1 which determines how much newly calculated Q-value will be applied. For instance, for α is 0, the agent is not learning anything, or for 1, the agent is always considering the most recent state-action. In practice, the constant learning rate is used $(\alpha(t)) = 0.1, \forall t)$ as the algorithm needs to converge, so it requires a learning rate close to zero [90]. $s(t+1)$ is determined by action $a(t)$, so $Q^t(s(t+1), a)$ are all possible actions' Q-values from future state. So the discount factor γ (between 0 and 1) affects the importance of future penalty. A small discount factor gives more penalties in

the near future penalty, and high discount factor accounts more for the far future penalty. This parameter needs to be tuned experimentally. $\min(\forall Q^t s(t+1), a)$ can be viewed as the estimate of the optimal future value. The difference between old Q-value (Q^t) and learned value ($PT(t+1)+\gamma \min\limits_{a}(\forall Q^t(s(t+1), a))$) updates the new Q-value (Q^{t+1}) with the learning rate.

The penalty term, ($PT(t+1)$) in (4.7) at $t+1$ time, is the penalty obtained after performing action $a(t)$ in state $s(t)$ on the dark silicon manycore processor. In our problem, we have three main constraints: total core power, performance deadline of all the tasks, and temperature upper limit. EM-induced lifetime is what we want to maximize. As a result, we define the penalty function PT in [37, 34] to consider multiple constraints.

We can build a penalty term (PT) as shown in (4.8) for each EM-induced and energy optimization. PT_E is a penalty term for total core energy, PT_{EM} is a penalty term for EM-induced lifetime, PT_{SER} is a penalty term for system-level soft error rate, PT_{power} for power, PT_{temp} for temperature, and PT_{perf} for performance deadline of all tasks. Each penalty term (PT_x) is normalized in (4.8). We use the feature scaling method to bring all values between 0 and 1. For instance $PT_E = \frac{E(t+1)-E(t)}{E_{Max}-E_{Min}}$ for energy related penalty, where $E(t)$ is the total energy consumption in the previous time t and $E(t+1)$ is energy of the system at current $t+1$. For the EM lifetime, $PT_{EM} = \frac{MTTF(t)-MTTF(t+1)}{MTTF_{Max}-MTTF_{Min}}$ for EM related penalty where $MTTF(t)$ is the MTTF of the system for EM-induced in the previous time t and $MTTF(t+1)$ is the MTTF of system at current $t+1$. Similarly, for the soft errors, $PT_{SER} = \frac{SER(t+1)-SER(t)}{SER_{Max}-SER_{Min}}$ for soft error related penalty where $SER(t)$ is the soft error rate of the system in the previous time t and $SER(t+1)$ is the soft error rate of system

at current $t+1$. Energy and EM terms can be interchangable, so both energy and life optimizations can be achieved with similar penalty term as seen in (4.8).

$$
\begin{aligned}
PT &= PT_E + C \sum_{x=\{EM,SER,power,temp,perf\}} \delta_x PT_x \\
&\text{for energy optimization} \\
PT &= PT_{EM} + C \sum_{x=\{E,power,temp,perf\}} \delta_x PT_x \\
&\text{for EM-induced lifetime optimization} \\
\delta_x &= \begin{cases} 0 & \text{if } PT_x \leq B_x + \Delta_x \\ 1 & \text{if } PT_x > B_x + \Delta_x \end{cases}
\end{aligned}
\tag{4.8}
$$

where δ_x is a binary function to active ($\delta_x = 1$) or inactive ($\delta_x = 0$) of user defined or given constraint bounds, B_{power}, B_{perf}, and B_{temp} in the penalty term. They are also normalized power, performance, temperature bounds respectively. Each Δ_x is the difference between each bound and average penalty (PT) for power, performance, and temperature. Δ_x is negative if the system violates the given constraint, otherwise, it is positive and the system is bounded and performs well. Therefore, if the system violated the user constraints in the past, then the penalty can be quite significant (due to large value for constant C in (4.8)).

Our learning-based lifetime/energy optimization algorithm steps can be explained as follows: the input is an initial state set for each core with p-state and core status and output is the selected p-state and core states. First, all the Q-values in the Q-table are initialized to zero. The current state, denoted as $s(t)$, finds an action $a(t)$ with the lowest Q^t in (4.7) and switches to next state with corresponding p-states and active cores. For

every step, EM lifetime, soft error rate, performance, temperature, and power are evaluated and thus, the whole environment can be updated. Then, it calculates the new corresponding penalty $PT(t+1)$ in (4.8) and Q^{t+1} can be updated (learning process). After the update, the current state is discarded in exchange for a new action and subsequent iterations yield more updates with new states. Finally, when all the Q-values changes are less than a certain threshold, the best policy will be chosen.

4.2.2 MILP based formulation and solution for lifetime optimization

The second approach for lifetime optimization that we apply is the mixed-integer linear programming (MILP) method. MILP formulation of the performed constrained lifetime optimization method is more straightforward than the Q-learning based method. MILP also deliver better results than Q-learning based method as shown in this work. However, in general, MILP has higher computation costs than the Q-learning method. Also, MILP solver is very heavier, so it is not suitable for online management method. Hence we can use MILP solution to measure the solution quality of the Q-learning based method.

We know that the MTTF of a core (or lifetime) stressed by different periods with different temperature can be approximated by formula (4.2). For the manycore processor, we assume that the MTTF of the overall chip is determined by the minimum MTTF of all cores [79]. Let's define the lifetime of a core k for a given state s_i as $L(k, s_i)$, which will be built as a look-up first. Then the lifetime optimization problem for dark silicon manycore processor can be formulated as the following MILP problem:

$$
\begin{aligned}
&\max_{i} \quad \min_{k} \quad \sum_{i}\sum_{k} b_i c_k \; L(k, s_i) \\
&\text{subject to} \quad \forall i\colon \sum_{i} b_i = 1, \qquad \forall k\colon \sum_{k} c_k = 1, \\
&\qquad\qquad \sum_{k} Power_k \le B_{power}, Temp_{max} \le B_{temp}, \quad Perf_{max} \le B_{perf}
\end{aligned}
\tag{4.9}
$$

where $Power_k$ is k-th normalized core power, $Temp_{max}$ is maximum normalized temperature among cores, and $Perf_{max}$ is maximum normalized performance deadlines among cores (or all the tasks). B_{const}, B_{const}, and B_{const} are normalized performance, temperature, and power bounds allowed. Note that a selection for a chip lifetime can be denoted by a boolean variable b_k, which equals to 1 if the k-th core's lifetime is selected and 0 otherwise. Similarly, a state selection for a core is also denoted by a Boolean variable c_i, which equals to 1 if state s_i is selected for the chip and 0 otherwise.

4.2.3 Implementation of the dark silicon evaluation platform

To evaluate the proposed DRM algorithms, we implement a simulation-based platform for dark silicon processor. The platform is shown in Fig. 4.1. We first describe the major component models of the framework such as microarchitecture, power estimation, thermal and reliability models. Our proposed framework uses Sniper as a microarchitecture model, which is an accurate and fast application-level interval-based microarchitecture simulation [92]. The interval simulation is a recently proposed multi/manycore simulation framework at a higher level of abstraction which is faster than cycle-accurate full-system simulation. The interval simulation uses mechanistic analytical model, which is constructed from the mechanism of a superscalar processor core. The cycle-accurate full-system simu-

lator, such as gem5 (full-system mode) [93], GEMS [94], MARSSx86 [95] and SimFlex [96] can run both application and operating system (OS). These frameworks have the merit of having an accurate evaluation of I/O activities and OS extensive kernel function. However, these full-time simulations are extremely slow and not very suitable for our framework because they rely on the existing OS systems, which currently do not support manycore and dark silicon architectures in their simulators [97]. Thus, to support dark silicon and maycore processor, we choose application-level Sniper simulator. This Sniper interval-based model is accurately matching well with the Intel x86 multi-core architecture [92]. PARSEC [98] and SPLASH-2 [99] benchmarks are used for our platform workloads. PARSEC benchmark is recently released multithreaded benchmarks, which provide an up-to-date collection of modern workloads for multi/manycore systems, and SPLASH-2 has been used many multi/manycore research for a long time. We use both workloads to evaluate our proposed framework and algorithm in Section 4.3.

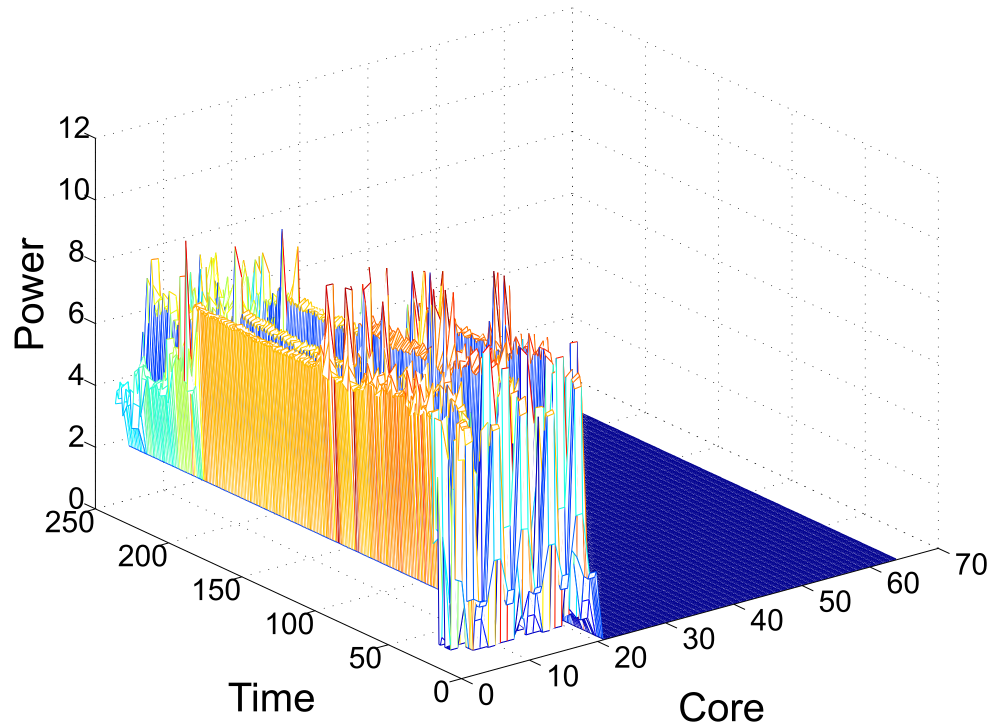
For the power estimation, we use McPAT (multicore power, area, and timing), which is a recently proposed full integration modeling framework. McPAT can provide dynamic and static, even short-circuit power dissipation and provides multi-threaded and multi-core processor models. For the thermal model, we use HotSpot to accurately characterize the thermal traces from the given multithreaded task run in each core [70]. To enable the dark silicon feature, the floor plan and power trace are dynamically controlled by the dark silicon DRM module in Fig. 4.1

As shown in Fig. 4.1, once the cycle per instruction (CPI) stacks and power/energy traces are achieved in the microarchitecture model with the power model, the thermal model

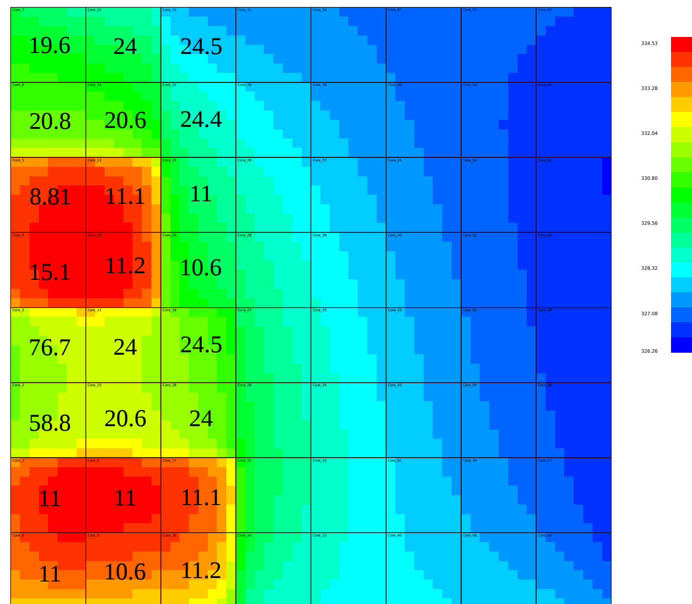
can generate thermal traces for given task run. With each core's power trace, thermal trace, core voltage, core frequency, and active cores, we can perform EM and soft-error reliability effects analysis and the system-level assessment for microprocessor lifetime based on the reliability models. Fig. 4.2(a) and 4.2(b) show the results from the proposed framework, which are the power traces, thermal measurement, and EM lifetime on a 64-core dark silicon chip. There are 20-core-enabled at the normal DVFS setting (2.0Ghz, 1.2V) and 64 multi-threaded tasks (16x CHOLESKYs, 16x RADIXs, 16x RAYTRACEs, 16x VOLRENDs) on a 64-core dark silicon chip. Fig. 4.2 only shows the core area. Power budgeting is not applied here.

We remark that the soft error affects the short term hardware functionality and it has a different way to impact the reliability of the circuits than the long-term reliability. However, both effects hurt the reliability of a chip and we think it is necessary to consider both as the power/energy and performance have contradicting impacts on them. The trade-off has to be found among the robustness, costs and performance of the many-core processor systems to mitigate the both soft and hard reliability effects.

In our formulation, both soft errors and EM-induced reliability are modeled in terms of system soft error rates (SER) and mean time to failure (MTTF) respectively and parameterized by the chip and system parameters such as $V_{DD}/frequency$, temperature, p-state, etc. As shown in (4.7) and (4.8). The SER and MTTF will contribute the constraints of the optimization in terms of penalty terms. As seen in Fig. 4.1, our framework implemented two reliability models, such as EM [57] for long-term reliability and DVFS-aware soft-error effects [81, 82, 85] for short-term reliability. Each reliability has been assessed



(a)



(b)

Figure 4.2: (a) SPLASH2 benchmark 64 multithreaded tasks power traces with 44 cores off(b) Thermal (color:degree) and EM lifetime (number:yrs) analysis on 64 cores

for DRM module to provide constraints in our Q-learning. We also stress that DVFS and task scheduling may not be the most effective way to mitigate the soft errors and other techniques are required when the soft errors are high due to the conflicting requirement from hard reliability.

4.2.4 Time complexity analysis

It has been proved that MILP problems are NP hard. Though branch and bound techniques can be used to solve the problem, the time complexity is not easy to analyze as we use commercial CPLEX as the solver. For the Q-learning, each value iteration can be performed in $O(|A||S|^2)$ steps, or faster if there is sparsity in the transition function (where A is a number of actions and S is a number of states). In practice, policy iteration converges in fewer iterations than value iteration, and there is no known tight worst-case bound available [100]. As a result, we report the running CPU times of Q-learning and MILP in our experiments with more p-states in Table 4.3 in Section 4.3 to compare the time complexities of the two methods. We further remark that it is rather difficult to fairly compare CPU times for these two optimization methods as the Q-learning was running on Python, whereas the MILP was running on commercial CPLEX Optimizer. We only measure solving overhead time for Q-learning and MILP. Nevertheless, the numerical results still show that MILP has much higher computation cost than the Q-learning method.

4.2.5 Practical application of the proposed DRM with reliability models

Currently, it is very hard to build on-chip EM sensor to directly measure time to EM failure of a core for its current temperature and power consumption (Although we have

made some early efforts on on-chip EM sensors recently [101]). But at the full chip level, as far as power grid is concerned, EM can be measured by the voltage drops in power grids as we discussed in Section 2.1. In our many-core dark silicon simulation framework, the EM-induced time to failure (TTF) assessment technique at circuit level discussed in this chapter (which was proposed in [57, 59]) was used for each core.

For practical application of proposed reliability management method for a real chip, such EM and SER assessment techniques need to know the detailed of the power grid networks and power consumptions of gates or function modules for some real workloads. As a result, it can be built when the chip was designed and then power grid voltage drop sensors (for EM measurement) or look-up table/other behavior models can be constructed for time to failure as the function of temperature and power inputs, which can be measured or estimated accurately. The accuracy of the reliability assessments with respect to the real silicon data need to be calibrated in the accelerated testing conditions for real chip under practical workloads, which will go beyond the scope of this works and can be future research. We further stress that the on-chip temperature and the powers, which actually can be measured or accurately predicted. For instance, Intel's multi-core CPU has the one thermal sensor per core [102]. The power or functional block of the cores can be measured or estimated accurately using the performance counters [103]. Then our models proposed in this work can be applied to core-level reliability management.

We notice that the learning-based DVFS managements have been used before to deal with difficulty of controlling the dynamics of the multi/many core processors [104, 105]. Recently, reinforcement learning have been successfully applied for DVFS management of

multi/many core systems [35, 33, 106]. These approaches employ a simple type of reinforcement learning (Q-learning) because this method has a relatively low overhead in terms of execution time and memory foot print. In this work, we assume the similar light scheduling overheads or time costs by the Q-learning methods. We have added the execution time for the proposed Q-learning method in Section 4.3.5.

Furthermore, as the state space of many core systems become large, we can explore overhead-aware scheduling method [106], the discrete lookup table method [107, 108] or pretraining based method [33] during the Q-learning to avoid expensive Q-value updating operations. In our work, we use lookup table based method.

4.3 Numerical results and discussions

4.3.1 Experimental setup

The proposed new energy optimization algorithm in the dark silicon framework has been implemented in Python 2.7.9 with the numerical libraries (Numpy 1.9.2 and Scipy 0.15.1). For dark silicon framework, we modified the architectural simulator (Sniper 6.1), power estimator (McPAT 1.0.32), and thermal simulator (HotSpot 5.02 [70]) to estimate EM-induced lifetime and system-level soft error rate on top of the new physics-based EM model [57] and DVFS-aware soft error model [81, 82, 85]. In the proposed framework as shown in Fig 4.1, each simulator module is connected with a plugin connector, so that one simulator's result can dynamically feed the other's inputs. The learning agent and Q-learning method have been implemented in Python 2.7.9 with extensive solutions from Python Numpy Extension.

Our energy optimization method is validated with a 64-core processor on the PARSEC and SPLASH-2 multi-threaded benchmarks. A small number of tasks with PARSEC (1 BLACKSCHOLES, 1 CANNEAL, 1 FREQMINE, and 1 VIPS), and for a large number of tasks with SPLASH-2 (16 CHOLESKYs, 16 RADIXs, 16 RAYTRACEs, 16 VOLRENDs) are used with the same 64 threads.

We chose two performance states (p-state) with the clustered DVFS [109], which have been employed to reduce the simulation time with small solution quality degradation due to the large number of cores in our experiment. To show that our method can find the lowest possible energy optimization, we compare our results with the global DVFS method, in which all active cores have the same p-state.

The full power mode (2.0GHz, 1.2V) and the low power mode (1.0GHz, 0.9V) have been set for our framework. For the soft error model, we use system-level soft error rates, the architecture constant ($\beta = 1.5 \times 10^{10}$ and $d = 2$), AVF (a constant = 0.5) for each core and the threshold voltage ($V_{th} = 0.9$) have been obtained from [85] and Enhanced Intel Speedstep Technology [69] with 45nm technology.

4.3.2 Evaluation of the proposed Q-Learning lifetime optimization method

First, we evaluate our learning-based DRM method (see Section 4.2) by showing lifetime improvements with different sets of power budgets and performance deadline. Fig. 4.3 shows the lifetime improvements given power budget and performance deadline for a small and large task set on 64-core dark silicon chip. As we can see in 4.3(a), for the small task set case our method finds relatively high lifetime improvement (87.9 yrs) as the task loads are small and more cores can be in low power mode or turned off (dark silicon)

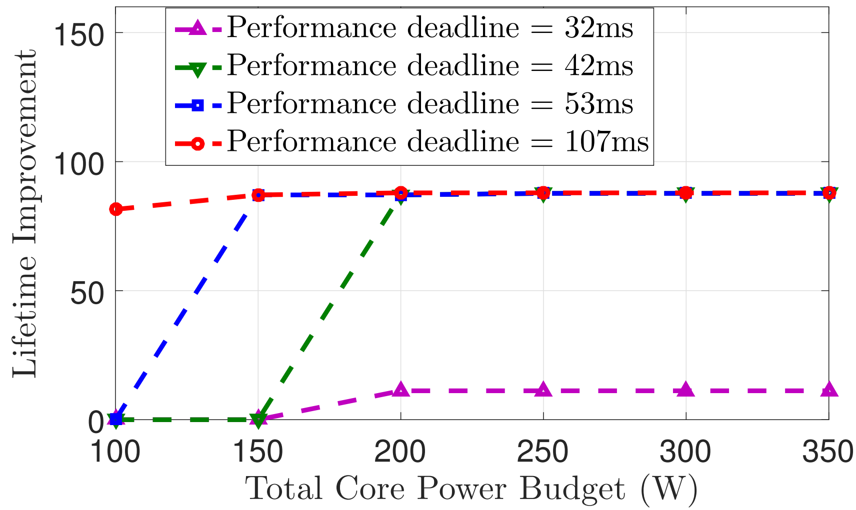
with the given power budget and performance deadline. In small performance deadline (42ms), there is still chance to improve lifetime (11.2 yrs) in the high power budgets (200-350W). However, for the large task set case, lifetime improvement will be limited as shown in Fig. 4.3(b). The highest lifetime improvement is 28 yrs with highest power budget, and there are still 10.5 yrs lifetime improvement in the middle range of power budget and performance deadline (40ms-80ms). This indicates significant improvement can be made for both small and large tasks with given and power budget and performance deadline.

Fig. 4.4 and Fig. 4.5 show the power consumptions and performances from our proposed DRM method and it indicates all the results can meet the given power budgets and performance deadline. Furthermore, no violations were found in either small or large task set results.

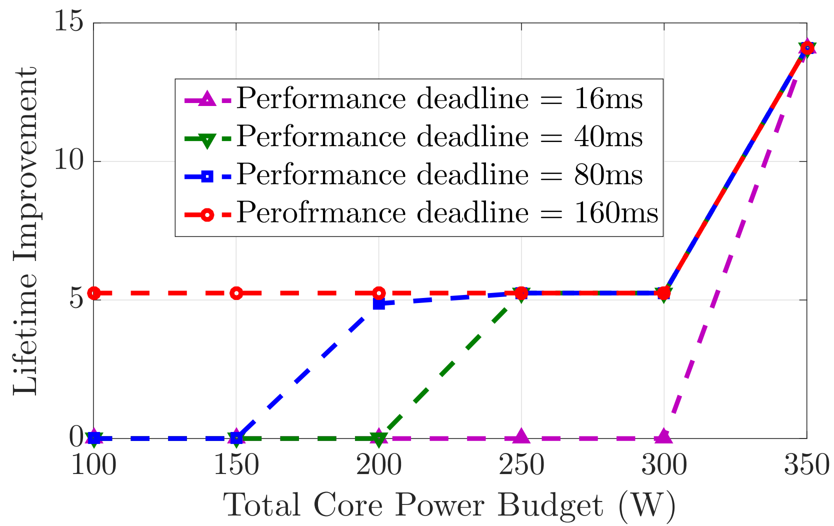
4.3.3 Accuracy and convergence rate of proposed Q-learning DRM method

Now we show some results from our second method for lifetime optimization method, MILP solver. To see the accuracy, we use MILP formulation (4.9) with the given solver, which is limited to post-validation as MILP method is very expensive to solve for large-scale problem. Nevertheless, it can be used to show the quality of the solution obtained from the learning-based DRM method.

A comparison of the Q-learning DRM method and the MILP method shows lifetime improvement for both small and large cases with results shown in Fig. 4.6. MILP method can deliver better results but with higher computational costs for large scale optimization. To see the accuracy comparison, 100 iterative tests are carried out for each case. For small and large cases in Fig. 4.6, our proposed Q-learning DRM method can achieve relatively high



(a)



(b)

Figure 4.3: Lifetime improvements given power budget and performance deadline on 64-core dark silicon chip (a) PARSEC small task set (b) SPLASH-2 large task set

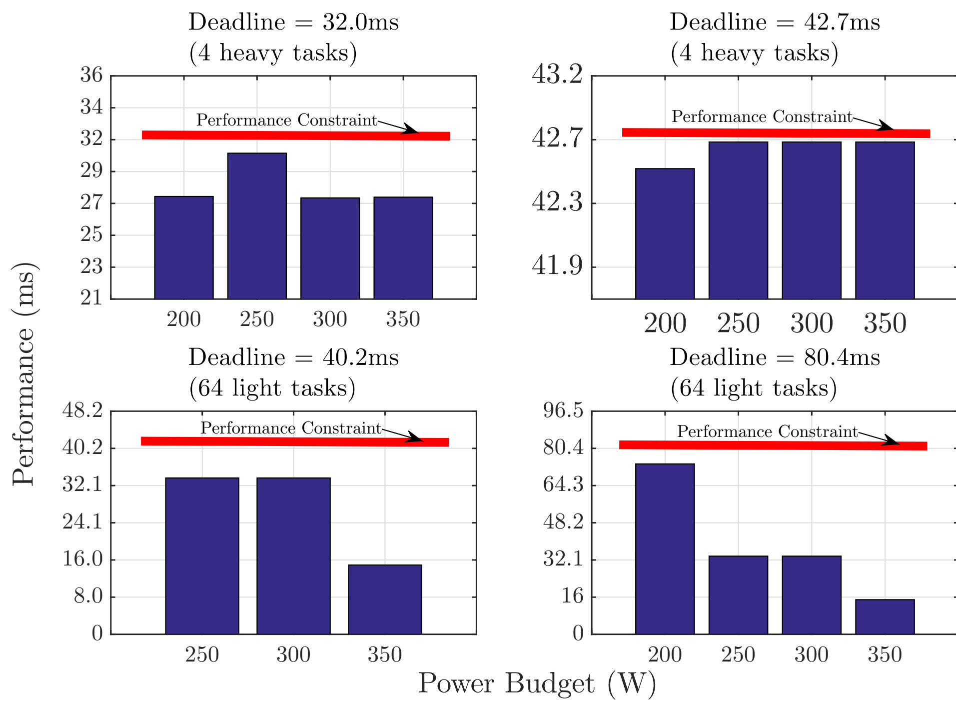


Figure 4.4: Q-learning result for performance deadline from (a,b) PARSEC tasks and light (c,d) SPLASH-2 tasks in 64-core dark silicon chip

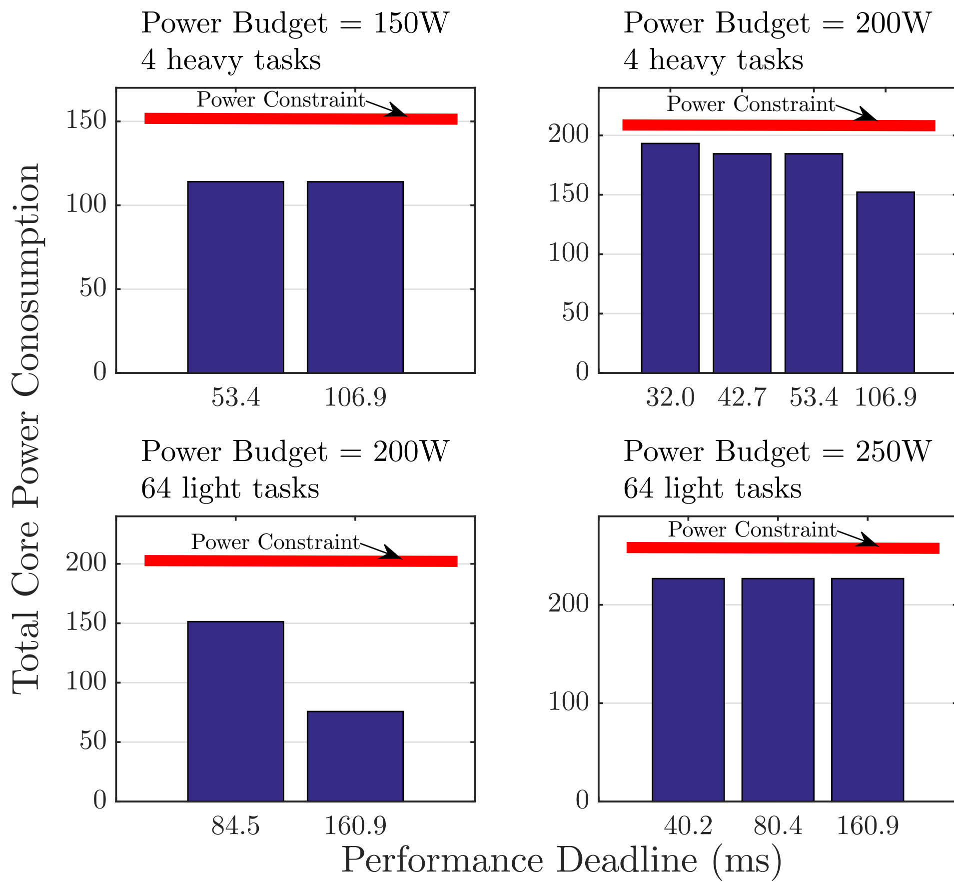
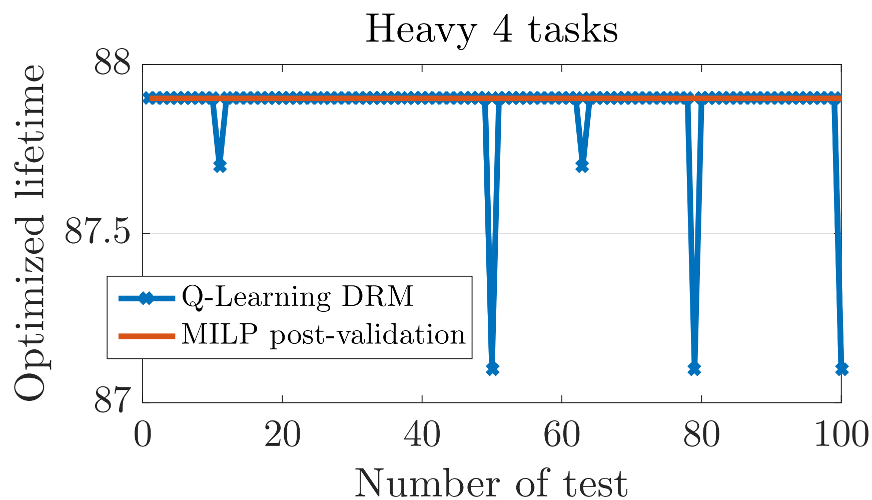
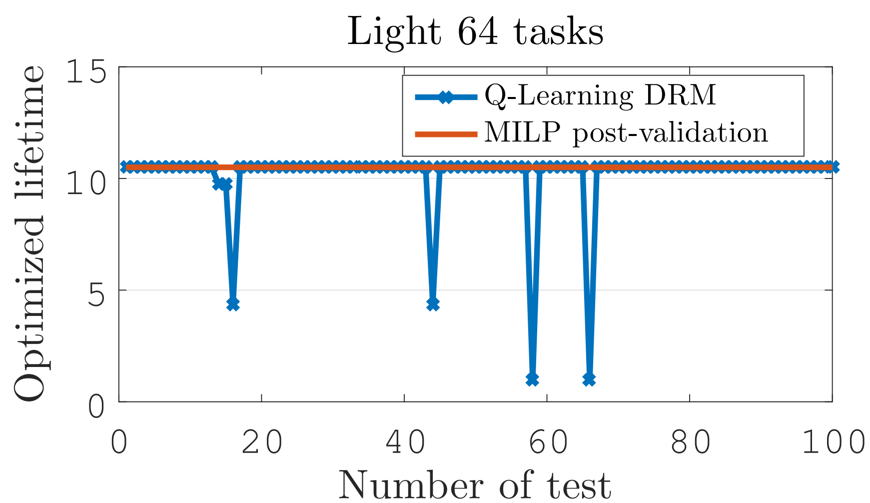


Figure 4.5: Q-learning result for power constraints from heavy 4 PARSEC tasks and light 64 SPLASH-2 tasks in 64-core dark silicon chip



(a)



(b)

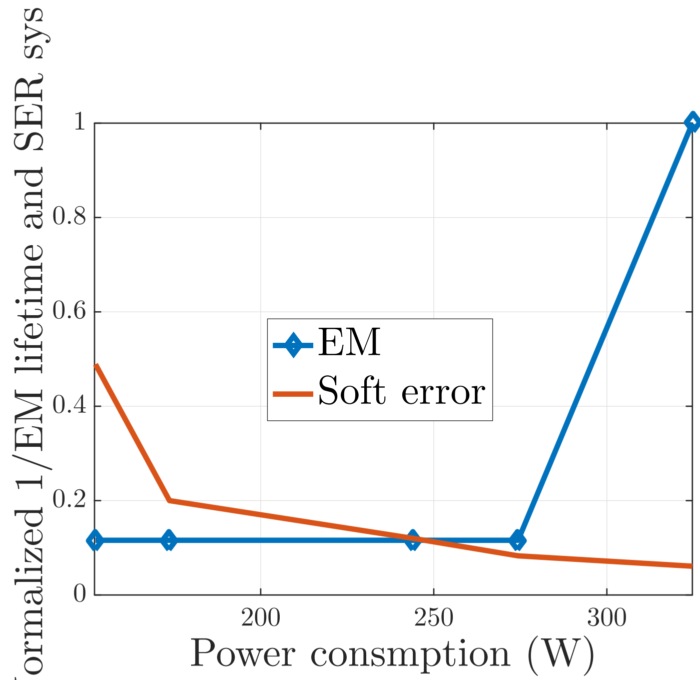
Figure 4.6: Post-validation with MILP for Q-learning accuracy (a) 4 heavy task PARSEC workloads (b) 64 light tasks SPLASH-2 workloads

accuracy, 95% and 94% respectively. It also shows that system violation can be effectively prevented by our proposed penalty function (4.8).

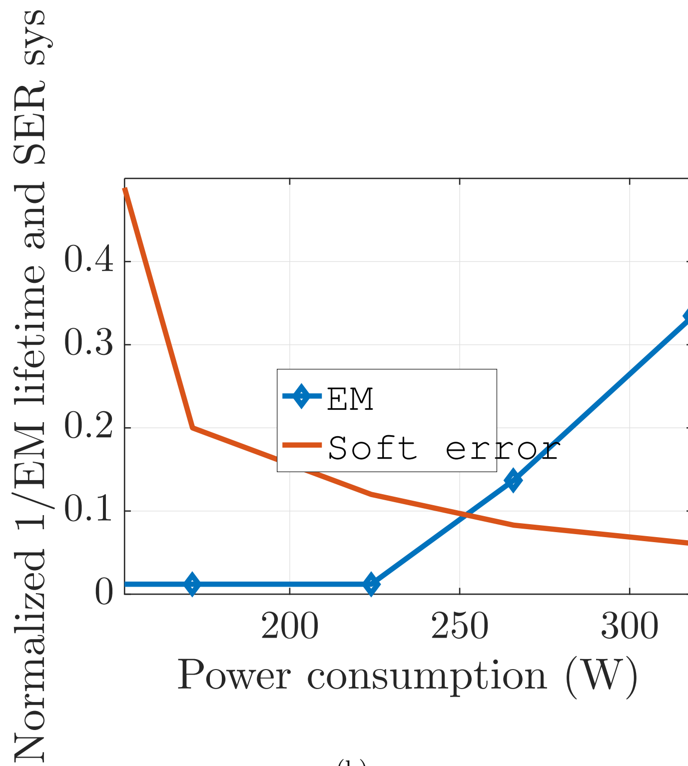
4.3.4 Hard and soft errors in dark silicon manycore processor

For the both reliability effects, we show the different impacts that power consumption has on both EM and soft error related reliability effects. We try to observe two reliability effects (EM-induced lifetime and system-level soft error rate) on the 64-core dark silicon manycore processor with the different task sets. Fig. 4.7 shows how two failure rates ($1/MTTF_{EM}$ and SER_{sys}) change over power determined by different DVFS settings when 12 cores are turned on with a 64 thread multi-threaded benchmarks. As we can see, for long-term hard reliability (electromigration effect), high power leads to short lifetime. However, for short-term/transient soft errors, low power will lead to much worse reliability issues. As a result, the system level optimization subject to both reliability constraints is no longer simple trade-off between performance/power versus reliability effects.

Fig. 4.8 shows that our DRM method in the different process technologies can affect to our soft error reliability model. Also, as we discussed in Subsection 4.1.3, technology scaling can be impacted on soft error rate and its constraint. Smaller technology has higher soft error rate. As we seen in Fig. 4.8, with smaller technology nodes, our DRM method under loose EM (5 years) and loose SER (0.6) constraints finds less energy saving as smaller technology can affect higher soft error rate for 32nm and 22nm cases (case D and E) than 45nm cases (case C). However, our method still can find better energy saving point.



(a)



(b)

Figure 4.7: Comparison between EM-induced lifetime and system-level soft error rate at different powers (by different DVFS configurations) on (a) PARSEC small tasks and (b) SPLASH-2 tasks

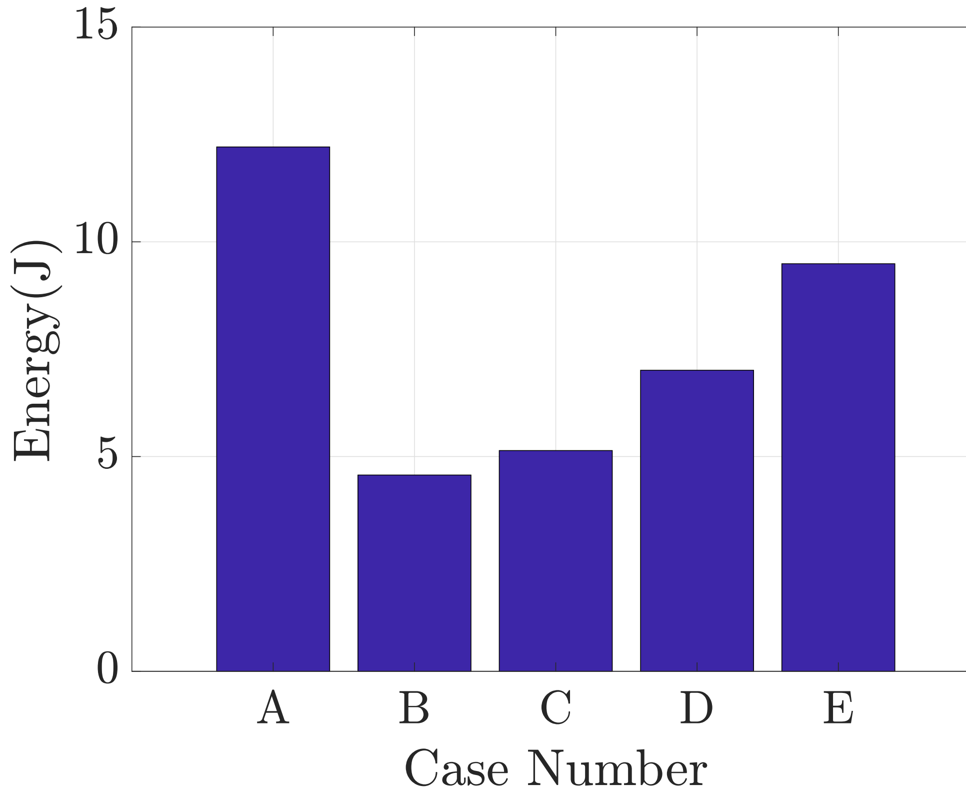


Figure 4.8: Impact of different process technologies on system-level soft error rate, from left bar, case A) Global DVFS, case B) our proposed DRM with only EM constraint, our proposed DRMs with both EM and SER constraints of case C) 45nm, case D) 32nm, and case E) 22nm

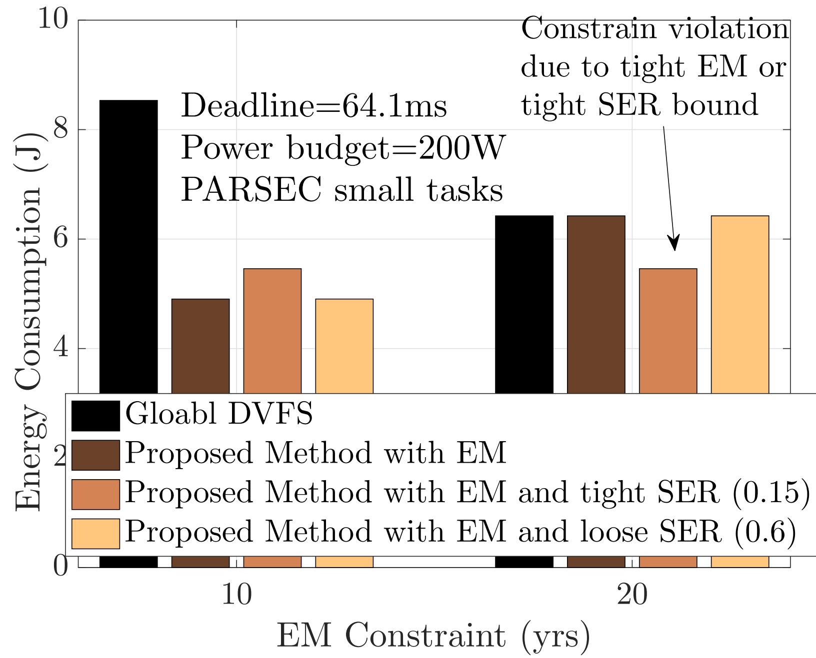
4.3.5 Evaluation of proposed Q-Learning based energy optimization method

To evaluate the proposed Q-learning based energy optimization method in Section 4.2, we show the total energy consumption with the different sets of EM-induced lifetime constraint, system-level soft error rate constraint, power budgets, and performance deadlines. Energy optimization results for a small task set and a large task set cases are shown in Fig. 4.9 and Fig. 4.10, respectively. Each figure includes two groups with loose and tight EM, where the left 4 bars are loose EM constraints and the right 4 bars are tight EM

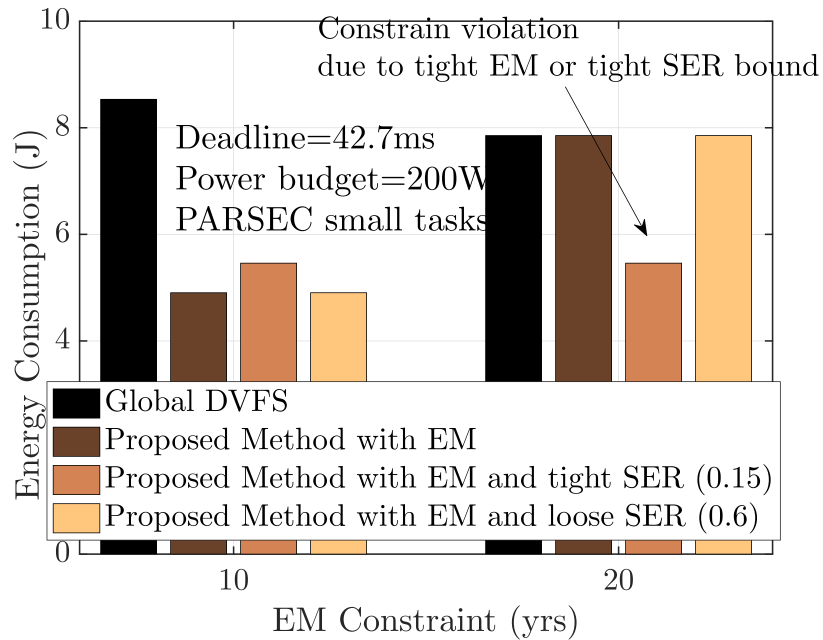
constraints. In each group, the third bar is tight soft error constraints and the fourth bar is loose soft error constraints. As we can see, if we consider only EM constraints for the different EM lifetimes, power budgets and performance deadlines for small task set case, the proposed method finds relatively high energy saving with both large and small performance deadlines (see Fig. 4.9(a) and Fig. 4.9(b)) than the global DVFS method because more cores can be in low power mode or turned off (dark silicon constraint) with the given power budgets and performance deadlines. When the loose soft error constraint is further considered ($SER_{sys} = 0.6$), our proposed method can still find good energy savings (similar or slightly higher than the method considering only EM constraint as shown in Fig. 4.9 and Fig. 4.10).

However, for the tight soft error constraint ($SER_{sys} = 0.15$) with tight EM constraint cases, our method violates the given EM constraints because soft errors and EM constraints have a contradictory relationship as seen in Fig. 4.7. For the large task set case in Fig. 4.10, more violations and less energy savings are observed. Those violations are caused by the higher task utilization of active cores, tight EM lifetime constraints, and power budget. However, our method can still find relatively good energy savings with the loose soft error constraints and tight EM constraints even at the large task sets. Thus, with exception to the violation cases (where both EM and soft error constraints are tight), our proposed method can find decent energy savings satisfying both EM and soft errors reliability constraints.

In Fig. 4.11, we further show the EM (a) and soft error (b) constrain violation cases under different EM and soft error constraints and power constraints (case 1,2,3,4 for

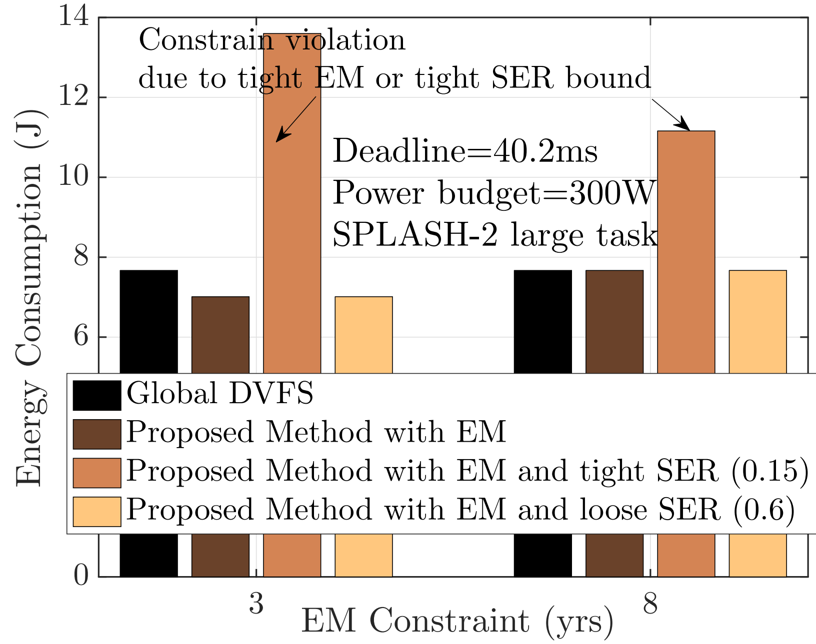


(a)

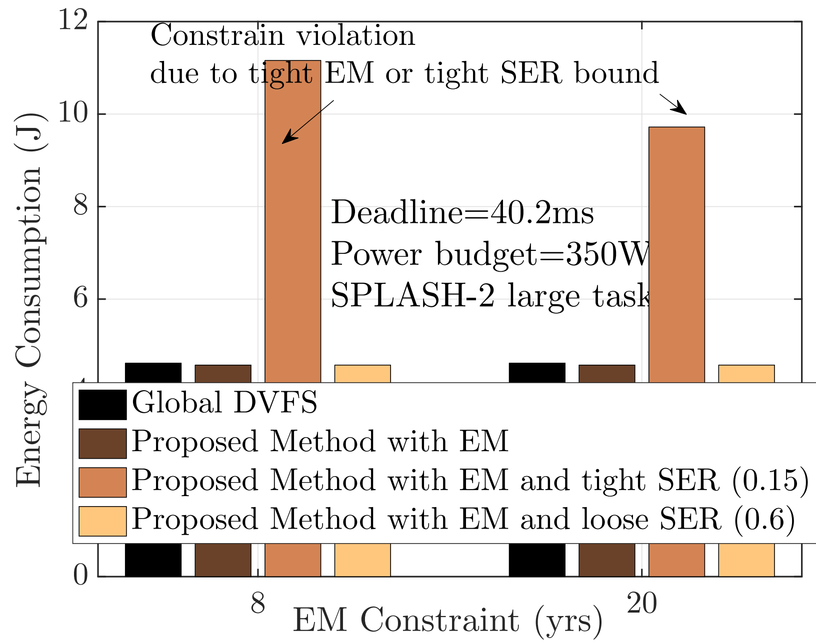


(b)

Figure 4.9: Energy optimization results (Global DVFS, proposed with EM, and with/without tight and loose soft error constraint from small task set on PARSEC benchmarks (different performance deadlines in (a) and (b))



(a)



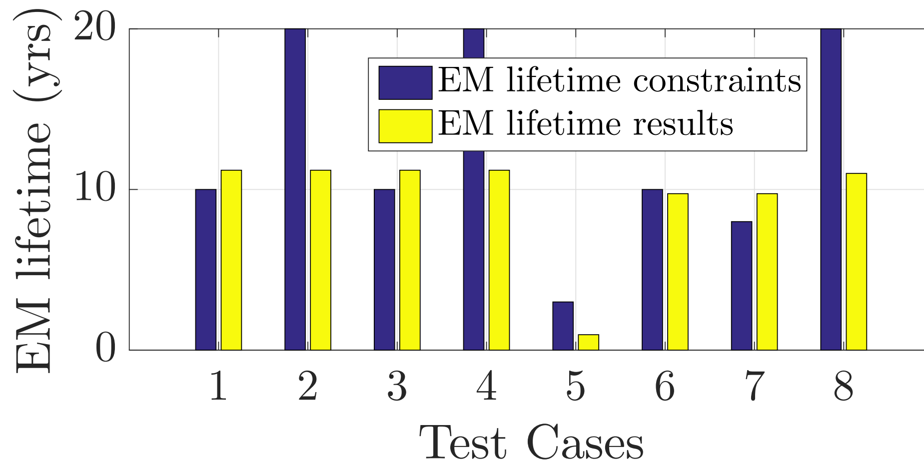
(b)

Figure 4.10: Energy optimization results (Global DVFS, proposed with EM, and with/without tight and loose soft error constraint from large task set on SPLASH-2 benchmarks (different power budgets in (a) and (b))

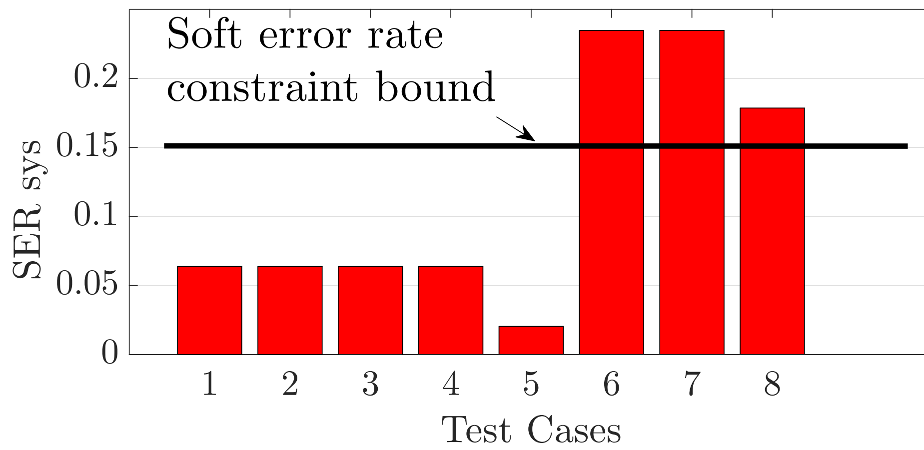
small task sets, 5,6,7,8 for large task sets). In all the case, (1-8), the soft error constraints are all tight. For case 2,4,8 cases, the EM constraints are tight. First, as we can see, both EMs and soft errors are violated in cases 6 and 8 due to the tight soft error constraint with large task sets. For cases 2 and 4, only EMs are violated due to both tight EM and soft error constraints with small task sets. More interesting is in case 5 where EM is violated even the EM constraint is not tight. The reason is that we have very small power budget (tight power constraint). For case 7, which is large case, the soft error is very tight. As a result, soft error constraint is still violated even with large power budget. Therefore, as we can see, for energy-efficient computing on many-core systems with power constraints (dark silicon), and competing hard and soft error constraints, the results are no longer simple trade-off among different factors, and instead, depend on those factors in a complicated way. In other words, under tightened power and performance constraints, we cannot satisfy both hard and soft errors at the same time and some other soft-error mitigation techniques become necessary in this case.

The proposed Q-learning method converges around 8% of explorations out of all possible state-action solution space as shown in Fig. 4.12. It also shows that system violation can be effectively prevented by our proposed penalty function (4.8).

To show the scalability of our proposed algorithm, we have added one more performance state on 64-core dark silicon chip with 64 tasks SPLASH-2 multi-threaded benchmark. The full power mode (2.0GHz, 1.2V) and the low power mode (1.0GHz, 0.9V), and very low power mode (800MHz, 0.7V) have been set for our framework. We compare two different numbers of p-states, case 1 has two p-states (full power and low power) and dark



(a)



(b)

Figure 4.11: Constraint violation cases

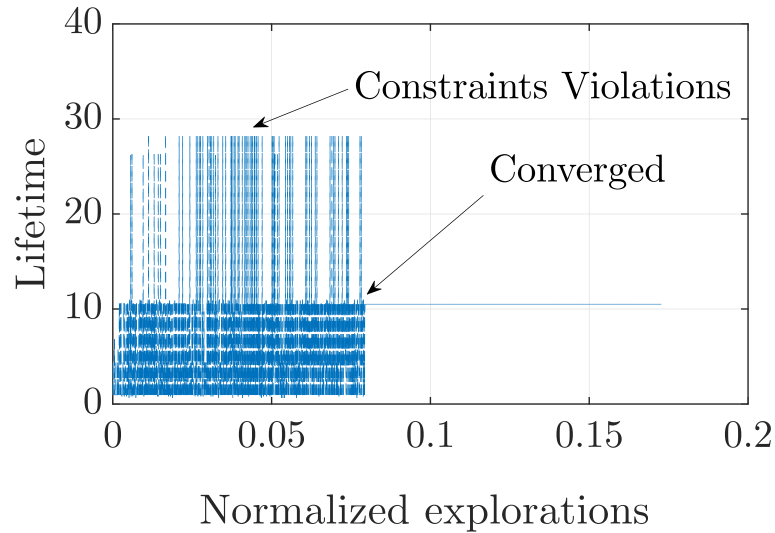


Figure 4.12: Convergence rate of proposed DRM method with EM-induced lifetime constraint in 64-core Dark Silicon (SPLASH-2 Tasks)

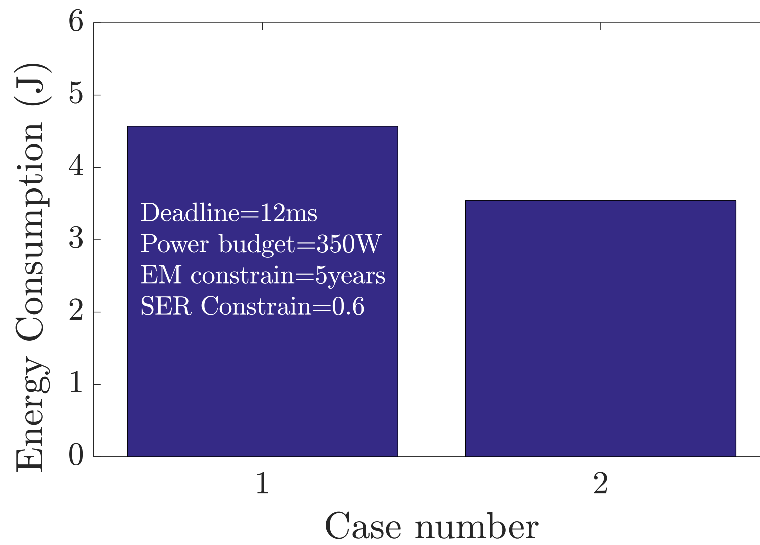
silicon states, and case 2 has three p-states (full power, low power, very low power) and dark silicon states. Due to very low power mode, we choose relatively loose constraints (deadline = 12ms, power budget = 350W, EM = 5 years, SER = 0.6). As seen in Fig. 4.13(a), our proposed algorithm can find low energy consumption with lower p-state cases. We chose three performance states (p-state) with the clustered DVFS [109], which have been employed to reduce the simulation time with small solution quality degradation due to a large number of cores in our experiment, so, the case 1 is 150 states, and the case 3 is 192 states. As seen in 4.13(b), case 3 has 28% more states, but the iteration requires only 6.58% more steps. Table 4.3 showed elapsed CPU time to solve the proposed Q-learning and MILP problem for each case.

Q-learning elapsed CPU time was obtained on iPython 5.1.0 with using only single core of Intel Xeon E5 system (clock 2.3 GHz) platform. Moreover, in order to show more

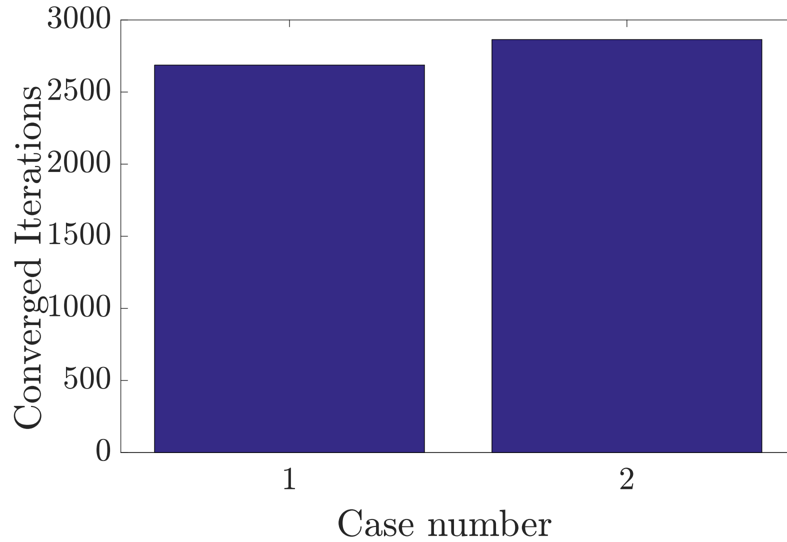
higher scalability of the proposed algorithm, we increase p-states to five different ones for two large cases (128-core and 256-core), and their voltage and frequency (V,f) sets are (2.0GHz, 1.2V), (1.6GHz, 1.1V), (1.2GHz, 1.0V), (1.0GHz, 0.9V), and (800MHz, 0.7V) on 64 tasks SPLASH-2 multi-threaded benchmark. We used 16-core as one cluster for the clustered DVFS. Also, we use 32-core as one cluster for dark silicon states in 128-core case and 64-core in 256-core case, which have been employed for a very large number of cores in our experiment. Table 4.4 shows the elapsed CPU time results for solving the proposed Q-learning result and energy savings over global DVFS. 128-core with 5 p-states (case 3) has 775 different states, which is 516% more states than case 1, but the iteration requires only 406% more steps. 256-core with 5 p-states (case 4) has 7160 states, which is 4773% more states than case 1, but only 4081% more steps are required. Thus, our proposed DRM can be a scalable solution for large number of cores and higher p-states cases since iteration steps and CPU times are close to increase linearly with respect to the total number of states. Both large cases also can find decent energy savings. Due to extremely high cost of MILP method for large-case (128-core and 256-core with 5 p-states), the elapsed CPU times of Q-learning method have been only shown in Table 4.4.

Table 4.3: Elapsed CPU Time to solve the proposed Q-learning and MILP problems

64 tasks in 64-core dark silicon (total states)	**Q-learning time (s)**	**MILP**
Case 1 - 2 p-states (150 states)	0.172 s	9.1 s
Case 2 - 3 p-states (192 states)	0.183 s	88.08 s



(a)



(b)

Figure 4.13: (a) The scalability analysis for our proposed DRM (case 1: two p-states with dark silicon, case 2: three p-states with dark silicon (b) Total average iteration number for both two cases

Table 4.4: Large-scale experiments with five p-state on 128-core and 256-core

number of core with 64 tasks with 5 p-states (total states)	**Q-learning time (s)**
Case 3 - 128-core (775 states)	0.699 s
Case 4 - 256-core (7160 states)	7.02 s

4.4 Summary

In this chapter, we developed a new energy and lifetime optimization technique for emerging dark silicon manycore microprocessors considering hard and soft errors. The new approach was based on a newly proposed physics-based electromigration (EM) reliability model to predict the EM reliability of full-chip power grid networks for hard error. DVFS-aware soft-error rate (SER) model and the Sum Of the Failure Rates (SOFR) method were employed for system-level soft-error model, which has been widely used to estimate microprocessor level soft errors. We employed both dynamic voltage and frequency scaling (DVFS) and dark silicon core state using On/Off pulsing action as the two control knobs. The impact on DVFS for hard and soft errors was investigated.

We focused on two optimization techniques for improving lifetime and reducing energy. To optimize lifetime, we first applied the adaptive Q-learning based method, which was suitable for dynamic runtime operation as it was able to provide cost-effective yet good solutions. The second lifetime optimization approach was the mixed-integer linear

programming (MILP) method, which typically yields better solutions but at higher computational costs. To optimize the energy of a dark silicon chip, we applied the Q-learning reinforcement learning method, which was suitable for our reliability management for the energy optimization considering hard and soft errors. Experimental results on a 64-core dark silicon chip showed that proposed methods work well for performance and lifetime optimizations considering the both soft and hard reliability constraints.

Chapter 5

Recovery-aware dynamic reliability management for near-threshold dark silicon processors

This chapter presents a new dynamic reliability management (DRM) techniques at the system level for emerging low power dark silicon manycore microprocessors operating in near-threshold region. We mainly consider the electromigration (EM) failures with considering recovery effects. To leverage the EM recovery effects, which was ignored in the past, at the system-level, we develop a new equivalent DC current model to consider recovery effects for general time-varying current waveforms so that existing compact EM model can be applied. The new recovery EM model can allow EM-induced lifetime to be better managed at the system level.

5.1 Recovery-aware Electromigration modeling at system levels

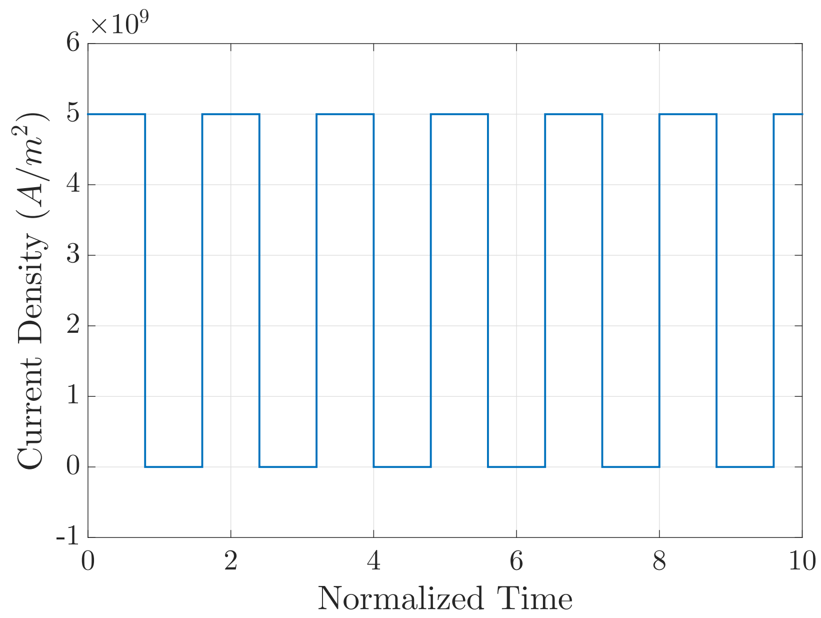
5.1.1 New equivalent DC current based modeling for EM recovery effects

For EM failures, one of the important phenomenon is that the EM-induced stress can go down when the stressing current becomes small. This effect is called “EM recovery effect” as it represents important transient effects due to time-varying currents. Fig. 5.1 shows the EM-induced stress changes over time over a periodic current pulse. As we can see the stress can go down significantly. The net effect for such recovery effect is that the lifetime of wire due to EM can be extended significantly as it will take longer time for the stress to reach to the critical stress over time.

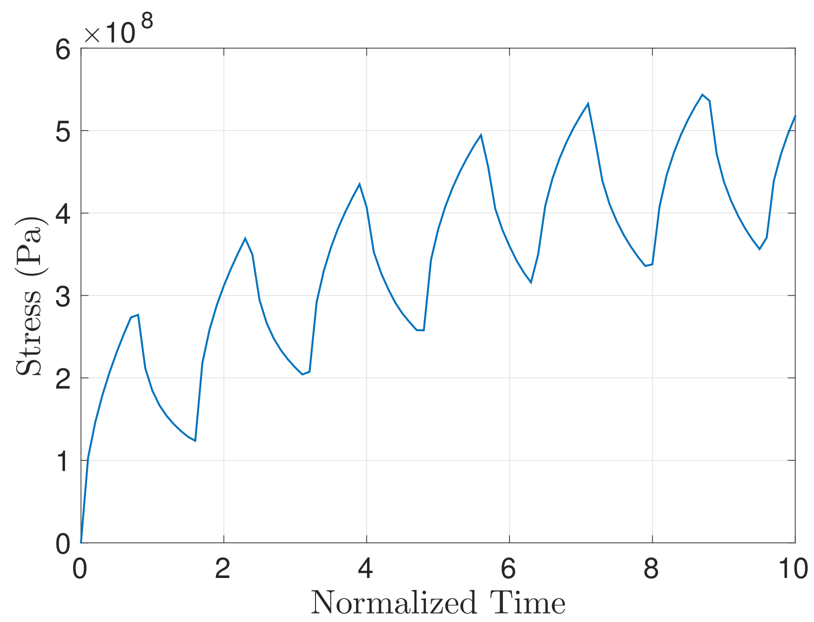
However, EM recovery effects were ignored completely in the existing EM models as most of those models assume constant current or current density. To mitigate this problem, a physics-based EM recovery model was proposed recently [110, 111] by obtaining an analytical solution of the Korhonen's equation describing the stress evolution kinetics of EM effects. Although the accuracy of this model is high, it is still too complicated for practical use.

For practical chip design, EM assessment and signoff still uses simple EM models like Black's model [60] or recently proposed, more accurate EM model in (2.2), which takes constant current and temperature as inputs. In order to consider practical no-DC currents, a simple time-varying equivalent DC current is computed as following,

$$j_{trans,EM,eff} = \frac{1}{P}(\int_0^P j_+(t)dt - \gamma \int_0^P |j_-|(t)dt); \tag{5.1}$$



(a)



(b)

Figure 5.1: Stress evolution caused by periodical current density

where $j_+(t)$ and $j_-(t)$ are the current densities of the positive and negative phases of the bipolar current, γ is the EM recovery factor, P is the period of the current waveform. When the current density is unidirectional, $j_{trans,EM,eff}$ essentially the time averaged current density. However, using the effective current formula in (5.1) will create a number of problems [112]. First of all, the recovery factor depends on the specific current waveforms which is not constant. Also, it ignores important transient effects such as the recovery and peak stress effects. Fig. 5.2 shows the stress evolutions over time driven by two current waveforms, the actual one and the time-varying equivalent DC current. As we can see, the peak stress due to the actual current waveform can exceed the critical stress while the average current never leads to void nucleation (wire is immortal).

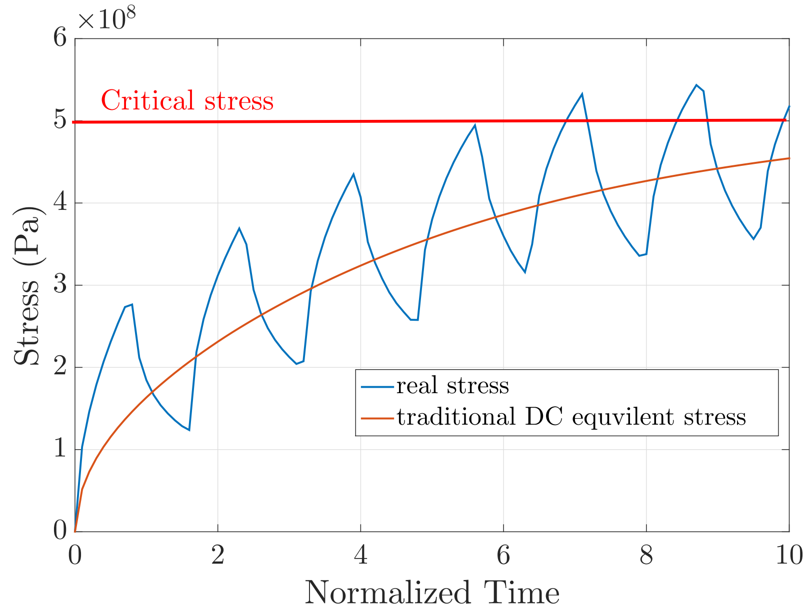


Figure 5.2: Stress evolutions caused by actual currents and traditional effective DC current

In order to solve the problems of these models, we propose a new and novel equivalent DC current method to consider the transient EM recovery effects. The new model

is based on first-principle based numerical analysis of EM effects. Here we use nucleation phase to compute the time to failure of a wire as a demonstration of this proposed method. The idea is that for a given EM model, the DC equivalent current will lead to the same time to failure (TTF) computed from the detailed numerical EM analysis of stress diffusion equation. This is better illustrated in Fig. 5.3(b) in which the periodic current density and a DC current gives the same nucleation time t_{nuc}. Unlike the traditional method which could ignore the case that the peak stress exceeds the critical stress but the equivalent current density never leads to void nucleation, so the transient effects are explicitly taken into account in this method.

That model works well for standard periodic square waveforms with one high current density (j_1) and one low current density (j_2). As shown in Fig. 5.3(a), j_1, j_2, period (P) and duty cycle(D) are used as the variables in the model. Also we find that the temperature (T) is one of the dominant parameters for the equivalent EM DC equivalent current density (j_{em}).

To further derive the parameterized equivalent DC current in terms of two currents, period and duty cycle and temperature, response surface methodology (RSM) [71] is carried out over many different waveforms from measured or detailed numerical analysis information. (5.2) is the fitted model to obtain equivalent DC current in terms of the five parameters.

$$
\begin{aligned}
j_{em} =&4.988 * 10^9 - 0.0663 * 10^9 * X_1^2 - 1.114 * 10^9 * X_1 * X_2 - 0.9981 * 10^9 * X_1 * X_3 \\
&- 0.1390 * 10^9 * X_1 * X_4 - 0.3485 * 10^9 * X_1 * X_5 - 0.0315 * 10^9 * X_2^2 \\
&- 0.1728 * 10^9 * X_3^2 - 0.3461 * 10^9 * X_3 * X_4 \\
&+ 0.0181 * 10^9 * X_4^2 + 0.0934 * 10^9 * X_5^2
\end{aligned}
\qquad (5.2)
$$

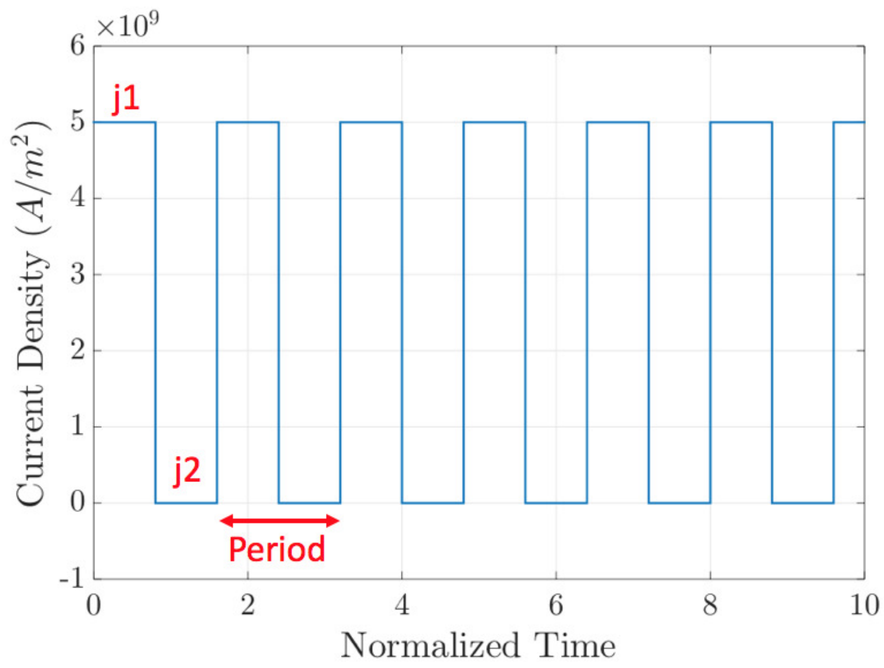
where

$$
X_1 = \frac{j_1 - 7.5 * 10^9 (A/m^2)}{2.5 * 10^9 (A/m^2)} \qquad X_2 = \frac{j_2 - 1.75 * 10^9 (A/m^2)}{2.75 * 10^9 (A/m^2)}
$$

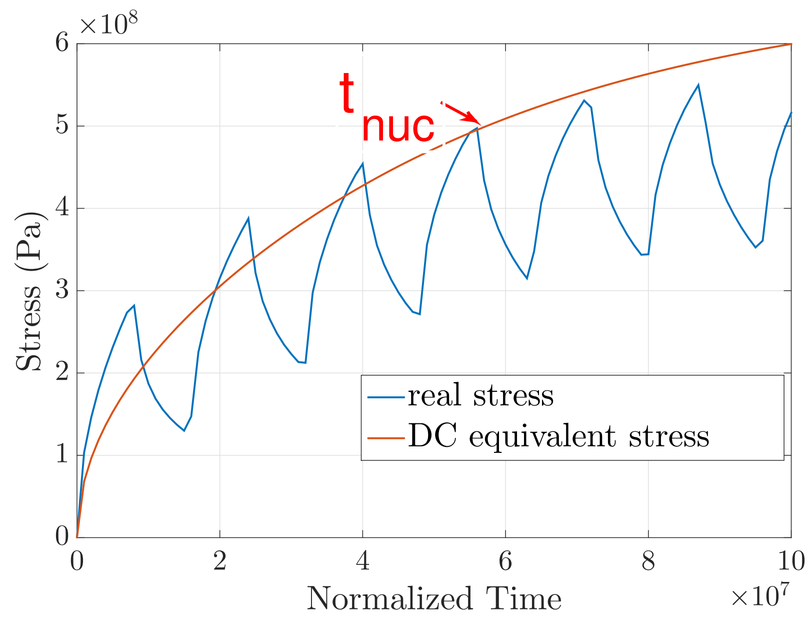
$$
X_3 = \frac{D - 70\%}{25\%} \qquad X_4 = \frac{P - 5.5 * 10^3 (s)}{4.5 * 10^3 (s)}
$$

$$
X_5 = \frac{T - 380(K)}{20(K)}
$$

However, this model can only handle regular square waveforms, but for practical cases, the current waveforms are arbitrary. To mitigate this problem, one of the ideas is to convert the arbitrary current waveform to an equivalent square waveform before we apply the aftermentioned parameterized equivalent DC current modeling. In this conversion process, we make sure that the stresses derived by the square waveform and the actual current waveform will match at both the highest peak stress and the final stress (end of period or time) as shown in Fig. 5.4(b). By matching the two stress points, we can find the two currents j_1 for highest stress point and j_2 for the end of period stress as shown in Fig. 5.4. During this conversion process, we assume that the given current waveform will repeat itself over time so that it becomes a periodic waveform. This assumption is reasonable as the future current or power of a chip cannot be predicted precisely in general and the recurrent assumption is a good guess.



(a)



(b)

Figure 5.3: (a) Original input driving current density. (b) Calculated EM DC equivalent current density with t_{nuc}

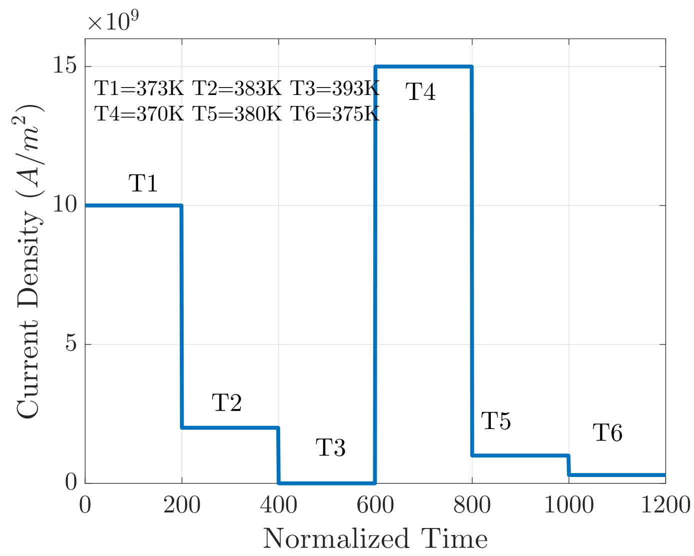
The other idea is to convert the arbitrary current waveform directly to a DC equivalent current so that the stresses from two waveforms match at the end of period time as shown in Fig. 5.4(b). But this approach may lead to large errors for time to failure estimation as it ignores the peak stress, which can be significant to determine the time reaching the critical stress (time to failure).

To study the accuracy of the two modeling methods, the two-step method (square waveform modeling and RSM fitting and the direct equivalent DC current method) is proposed. We compare stress generated by two-step method and the stress given by direct equivalent DC current method against the stress generated by the original current waveform. The results are shown in Fig. 5.5. As we can see, equivalent square DC current density (two-step method) has smaller error compared to the direct DC equivalent method in terms of time to failure estimation. As a result, we will use the two-step method to compute the parameterized equivalent DC current.

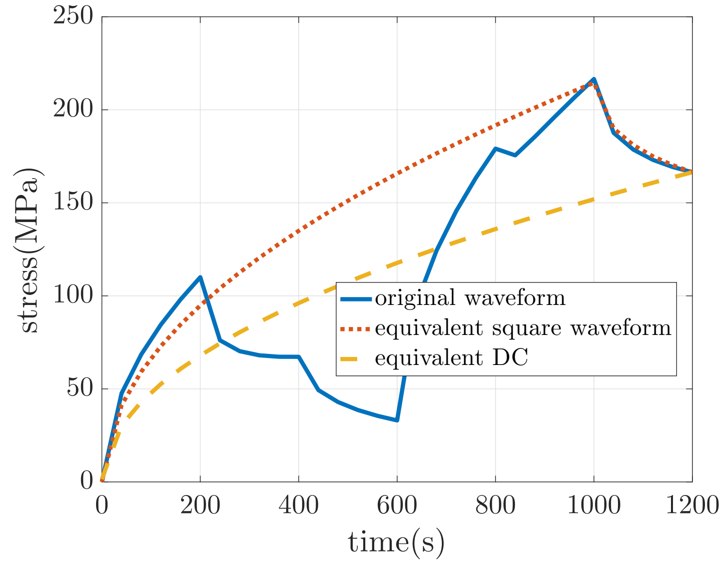
5.1.2 EM modeling for varying temperature effects

At the system-level EM reliability, the manycore system will run on different tasks under different voltages and frequencies. As a result, its temperature and current densities will change with time. However existing EM models including the new physics-based model can only take a constant temperature. The previous study shows that whole system MTTF or lifetime under different temperature can be approximated by [67]:

$$MTTF_{EM} = \frac{1}{(\sum_{m=1}^{n}(\Delta t_m \frac{1}{MTTF_m}))/T} \tag{5.3}$$



(a)



(b)

Figure 5.4: (a) Original input driving current density. (b) Calculated EM DC equivalent current density with two methods

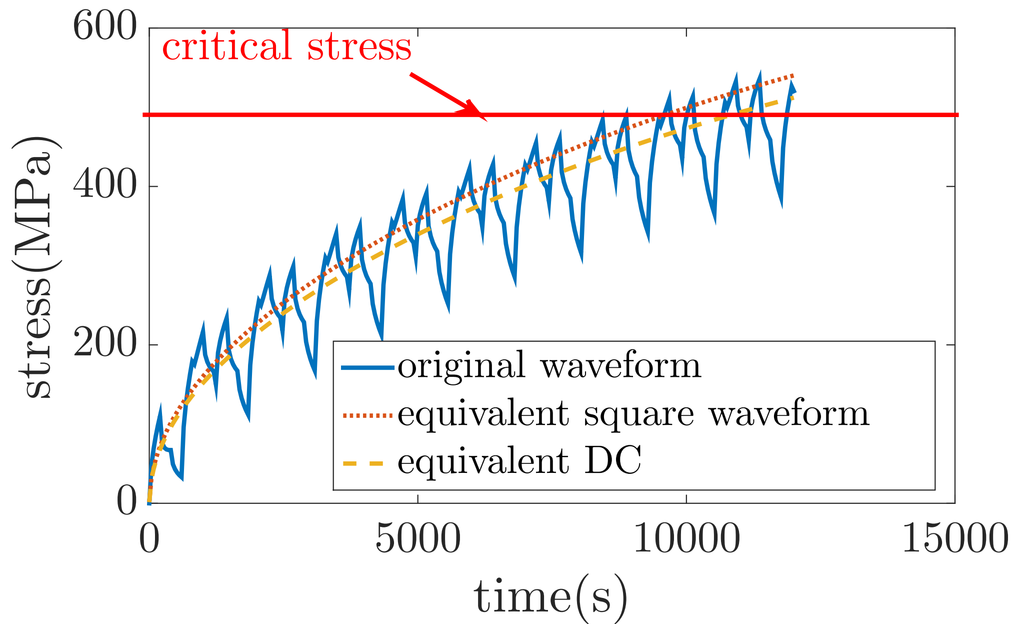


Figure 5.5: Comparing the nucleation time of two equivalent methods and original stress

where $MTTF_m$ is the actual MTTF (mean time to failure) under the m-th power and temperature settings for Δt_m period, assuming the chip works through n different power and temperature settings and $T = \sum_{m=1}^{n} \Delta t_m$. Each $MTTF_m$ will be computed based on the EM models discussed in the previous section. To consider a system-level EM reliability on a manycore dark silicon processor, we use the shortest lifetime among all the cores as the lifetime for all manycore processors [79].

5.2 New learning-based reliability management for near threshold dark silicon for EM recovery effects

5.2.1 Near threshold dark silicon

Near-Threshold Computing (NTC) has been proposed as a viable solution to overcome the limit of energy efficient computing by using optimal near-threshold voltage between super-threshold and sub-threshold region.

NTC cores are operated at or near their threshold voltage V_{th}. By reducing supply voltage V_{dd} from nominal 1.1 V to 500 mV, a 10X energy efficiency gain can be achieved at the expense of 10X performance degradation [113]. Compared to the sub-threshold region where 20X energy efficiency can be achieved, but the 50X performance degradation due to increased circuit delay is too big a factor to ignore for large-scale applications. Applications with significant standby times benefit greatly from NTC. Memories have to retain their contents even though digital logic is to be powered off. Thus supply voltage scaling results in a significant reduction in leakage power.

On the other hand, NTC is also a promising technique to mitigate the effects of dark silicon as cores can reduce power and temperature under a given power budget, thus, allowing a larger number of cores to be turned on simultaneously at costs of allowed performance losses. Recently, instead of operating the entire cores at either nominal voltage or near-threshold voltage (NTV), voltage islands have been defined such that only partial cores are operated at NTV and the rest is operating at a nominal voltage for more flexible trade-off between power and performance [114]. Supply voltage is proportional to the threshold

voltage of the transistors in the core. The core with the highest threshold can determine supply voltage for the voltage island. However, the different types of parallel workloads can lead to performance degradation and energy waste. Efficient dynamic management and scheduling to find suitable NTC regions are needed.

In addition to energy and performance, NTC has an effect on reliabilities. NTC may exhibit better long-term EM reliability, as a lower voltage can lead to lower temperature, current density and residual stresses, which are the major factors of EM effects [57]. The NTV, which is a lower supply voltage, can improve EM-induced lifetime of dark silicon processors. However, using NTV for many-core can make significant performance issue since NTV still use some cores operated at the nominal voltage and a many core system's reliability can be highly affected by those core's reliability [79].

5.2.2 Framework of dark silicon in near-threshold computing region

We present the framework for Dynamic Reliability Management (DRM) at NTC region in dark silicon. The DRM framework employs several simulator models (microarchitecture, power, thermal), a policy optimization module, all in conjunction with EM recovery model. Additionally, the DRM has policy optimizer that cores can choose the best NTC policy to maximize energy efficiency while meeting performance limit and power budgets. This work uses a 45nm-based 64-core dark silicon simulation framework with the threshold voltage of $V_{th} = 0.20V$, core On/Off knobs, and NTV capabilities.

The DRM module sets the voltage and frequency policy for the chip. Each core (k) can be assigned a voltage (V_k), nominal ($V_k^{nom} = 1.0v$) or near threshold (V_k^{nt}) which is defined as $0.40v \leq V_k^{nt} < V_k^{nom}$. Additionally, the DRM can turn a core off (V_k^{off}) and

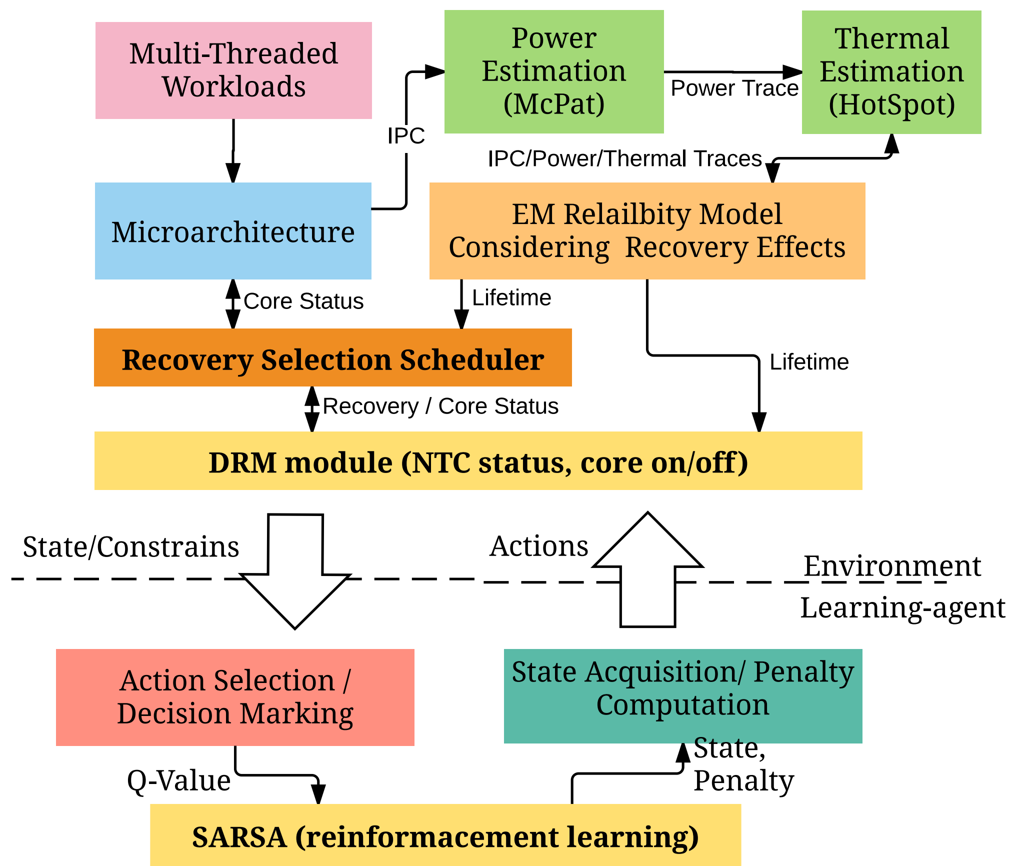


Figure 5.6: The DRM and NTC Framework

then turn a core back on($V_k^{nom/nt}$). The DRM makes these decisions based on an on-line policy optimization module that employs the SARSA algorithm which is explained later. In the framework, the DRM receives the new policy from the optimization module. It then sets each V_k or turns the core off. Additionally, each core operating frequency (f_k) is affected by V_k, because of this we use 5.4 as a relation to calculate f_k based on its respective V_k. This ensures that the performance degradation from NTV is reflected in the simulation framework. This policy is then propagated to the architecture, thermal, and power simulators as well as the EM recovery model, and optimizer.

$$f_k \propto \frac{(V_k - V_{th})^2}{V_{th}} \tag{5.4}$$

The framework uses the Sniper architecture simulator [92] to generate system performance for given workloads on a specified architecture. Parameters (chip floorplan, number of cores, frequency, and cache design) describing the architecture are passed to Sniper. Benchmarks representing the desired workloads are also used as inputs to sniper to simulate the system's functionality. Sniper then outputs system performance, such as performance characteristics, instruction per cycle, of the chip for each given benchmark run. This is repeated in our experimental setup for several different set of workloads. The whole near threshold dark silicon framework is illustrated in Fig. 5.6.

Once architectural system performance is generated and transferred to the physical simulators Hotspot [115] and MCPAT [116]. Based on the architecture of the chip, its system performance from Sniper, and the voltage scheduling from the DRM, MCPAT (Multi-Core Power Area and Timing simulator) will generate a power trace for each chip component

including each core P_k. Hotspot then uses the chip floorplan in conjunction with the power trace generated by MCPAT, to produce a thermal trace for each chip component and core T_k.

After the power/thermal/voltage characteristics have been generated by the various simulators, the EM Recovery model can use these parameters (P_k, T_k, and V_k) to estimate the chip's time to failure considering any recovery effects the chip may experience, from V_k^{nt} and V_k^{off} for that given policy.

Lastly, all the information generated, in addition to the current policy enacted by the DRM, are passed to the policy optimizer which will generate a new policy. This new policy will find the best voltage schedule for the various cores to optimize the energy of the chip while meeting MTTF, performance, and power budgets.

5.2.3 SARSA-based learning algorithm for DRM considering long-term recovery

We can model our DRM problem as a Markov Decision Process (MDP) with states $s(t)$, and actions $a(t)$ where states are the parameters of the framework for the time-step Δt, e.g., $f_{k(t)}$, $T_{k(t)}$, $MTTF(t)$, $P_{k(t)}$, and $V_{k(t)}$. Actions are defined as changes to these parameters which in our case is the tuning of $V_{k(t)}$. In our case, our goal is to achieve the best policy that minimizes energy E while meeting all constraints and budgets.

The reinforcement learning algorithm used to optimize the DRM policy is the State Action Reward State Action algorithm, or SARSA, first presented in [117]. SARSA is a combination of Q-learning and the traditional Temporal Difference method (TD) [117]. This algorithm exchanges the greedy updates of Q-learning with a policy driven update that

is closer to the TD method. The result is an on-policy reinforcement learning algorithm with faster convergence when compared to Q-learning [117].

The major differences with traditional Q-learning, is that the maximum reward (minimum penalty) for the next state is not used for updating the Q-values. Instead, a new action is selected using the same policy that determined the original action.

The SARSA algorithm works first by populating a Q-table $Q(s(t), a(t))$, where $s(t)$ is a state and $a(t)$ is an action for time-step Δt. It then selects an action from the states using some policy. This action is taken and the penalty $PT(t+1)$ (negative reward) and new state $s(t+1)$ are generated. From this new state, another action $a(t+1)$ is selected from $s(t+1)$. The Q-table is then updated using a penalty function shown below.

$$\begin{aligned} Q^{t+1}((s(t), a(t)) = Q^{t}(s(t), a(t)) + \\ \alpha(t) \times (PT(t+1) + \gamma Q(s(t+1), a(t+1)) - Q(s(t), a(t))) \end{aligned} \tag{5.5}$$

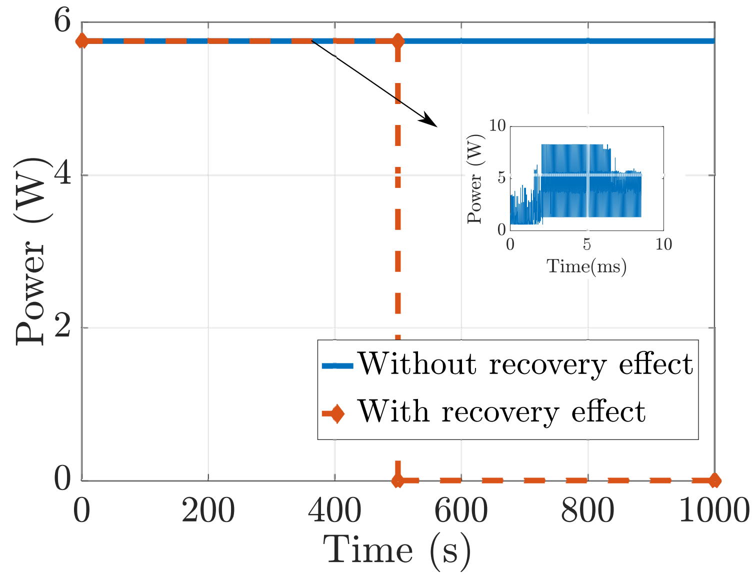
Here, α is the learning rate and γ is the discount factor. In our DRM, we employ multiple-constrained penalty function ($PT(t+1)$, [34]) and modify to accommodate each value (EM, power, temperature, and performance) and to also incorporate the power budgets, performance/thermal limits assigned as constraints.

In order to provide long-term shut-off time for leveraging EM recovery effects, our DRM have a recovery selection scheduler. It is a periodic scheduler on the top of SARSA for long-term recovery cycle ($T_{recovery}$), which is the required time for recovery effects. The selected cores can be turned off long-term recovery cycle as recovery effects needs some long-term as seen in Subsection 5.1.1. Every $T_{recovery}$ cycle, we use greedy-based selection algorithm by EM-induced lifetime evaluation and find the worst lifetime core set below the

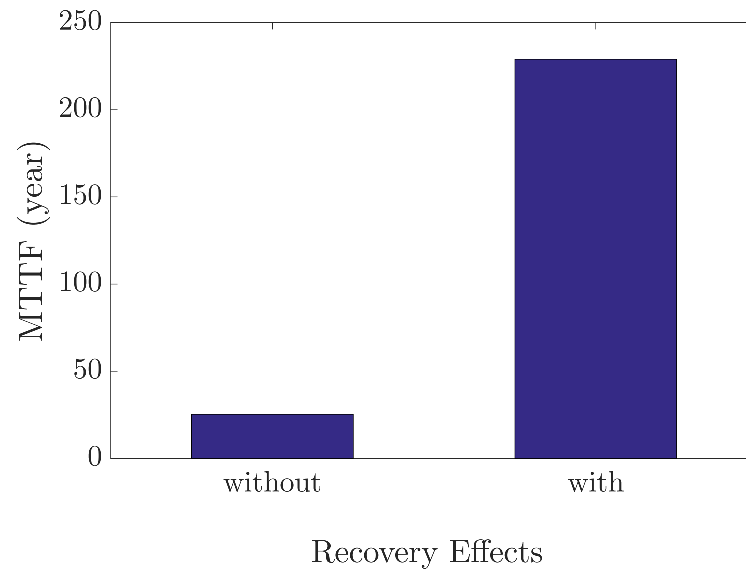
certain lifetime threshold ($EM_{threshold}$), then SARSA will work only for the cores except recovering cores for $T_{recovery}$. After each long-term cycle, we find new long-term recovery core set.

The proposed new energy optimization algorithm in the near-threshold dark silicon framework has been implemented in Python 2.7.9 with the numerical libraries (Numpy 1.9.2 and Scipy 0.15.1). For near-threshold dark silicon framework, we used the architectural simulator (Sniper 6.1), power estimator (McPAT 1.0.32), and thermal simulator (Hotspot 5.02 [70]) to estimate recovery-considered EM-induced lifetime on top of the new physics-based EM model [57]. In the proposed framework as shown in Fig 5.6, each simulator module is connected with a plugin connector, so that one simulator's result can dynamically provide the other's inputs. The learning-based SARSA method and recovery selection scheduler have been implemented for our dynamic reliability management (DRM).

The DRM module assigns each core as the near-threshold or nominal voltage, and calculates its frequency from 5.4. The near threshold voltages as 0.45V, and a nominal voltage of 1.2V are defined [118]. Additionally, each core can be turned off completely, or turned back on for dark-silicon. Once the simulation runs, the optimization module can then send a new policy to the DRM which it will use to schedule the core voltages. Our energy optimization method is validated with a near-threshold 64-core processor on the SPLASH-2 multi-threaded benchmarks.



(a)



(b)

Figure 5.7: (a) two cases of power traces from proposed framework and (b) and the resulting MTTF without/with recovery effects

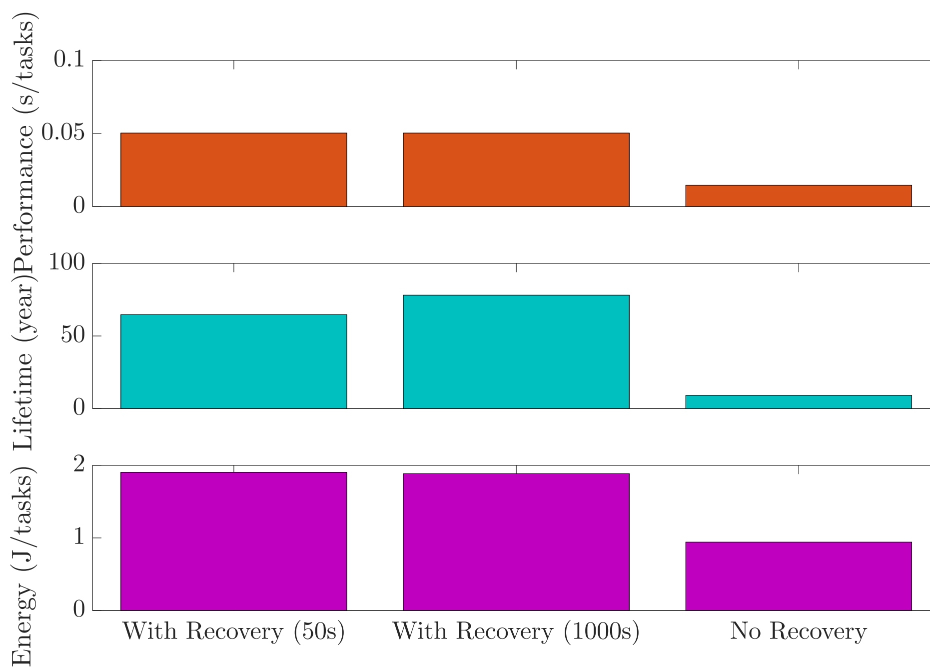


Figure 5.8: Performance, energy and EM-induced lifetime from proposed DRM considering recovery effects for three cases (1) Recovery effects with $T_{recovery} = 50s$ (first column) (2) Recovery effects with $T_{recovery} = 1000s$ (the second column) (3) Only DRM without recovery effects (the third column)

Table 5.1: Results for dynamic reliability management for 64-core near-threshold dark silicon

64-Core	**DRM (No recovery baseline)**	**All NTV cores**	**Half-NTV/Half-Dark**	**Half-Nominal/Half-Dark**
Performance (s/tasks)	0.014	0.23	0.48	0.03
EM lifetime (year)	9.03	221	221	7.1
Energy (Joule/tasks)	0.942	14.54	29.05	2.07

5.3 Numerical results and discussions

5.3.1 Evaluation of the lifetime impacts considering EM recovery effects

In order to evaluate system-level EM-induced lifetime considering recovery effects, a single core long-term task example case from our framework is shown here. Our proposed framework, shown in Fig. 5.6, can properly manage and control for both power-on and shutdown of each core, so we can significantly extend the system-level reliability by leveraging the EM recovery effects introduced in subsection 5.1.1. We present two different simple power traces and calculate EM-induced MTTF (mean time to failure) in Fig. 5.7. In this example, the time period is 1000 seconds and switch off for 500 seconds that can be recognized as a sufficient period for recovery effect. The original power traces (5.72 W) is converted to an equivalent power, which equals to 2.122W using our recovery model. Our duty cycle for

turning-off is 50% in the recovery case. As a result, it leads to 50% performance degradation. On the other hand, the EM-induced lifetime considering recovery effect is $9.16X$ higher than that of the original case without recovery effect case, which is quite significant.

5.3.2 Evaluation of the DRM for near-threshold dark silicon processors

To evaluate the proposed learning-based energy optimization method in Section 5.2, we show the comparison results of performance, EM-induced lifetime, energy consumption on processing 64 multi-threaded tasks on a 64-core near-threshold dark silicon processor. Our experiment uses performance and energy metrics as $s/tasks$ and $J/tasks$, which are total execution time and energy consumption for selected 64 multi-threaded tasks (16 CHOLESKYs, 16 RADIXs, 16 RAYTRACEs, 16 VOLRENDs) on our framework. The results are shown in Table 5.1. We compare our DRM case (without recovery effects) with all-NTV cores (64 cores are near-threshold voltages(NTV)), half-NTV/half-dark (32 cores are NTV and 32 cores are turned off), and half-nominal/half-dark (32 cores are nominal and 32 cores are turned off) cases. The optimization results show the energy consumption improvements for the given budget constraints (power budget is $250W$, performance limit $5s$ per given tasks, and EM-induced lifetime limit is more than 5 years). For all-NTV and half NTV/dark cases, EM lifetimes are significantly improved, but their performance results are $16X$ and $35X$ slower than the DRM baseline result. On the other hand, for only half nominal/half dark case, energy and performance are quite improved but EM lifetime is relatively short. Therefore, our DRM result effectively finds better EM lifetime (9.03 years) with significantly high performance with the lowest energy.

5.3.3 DRM considering recovery effects

Finally, we evaluate our proposed DRM method considering the EM recovery effects. As seen in Fig. 5.8, we evaluate two different recovery cycles ($T_{recovery} = 50s$ and $1000s$), so all the cores' MTTFs are periodically evaluated for every $T_{recovery}$ to determine which core needs to be turned off for the whole period when the core is below the certain lifetime threshold, $EM_{threshold}$. For our experiment, we set 8 years as the $EM_{threshold}$ in the recovery selection scheduler. As we can see, both DRM cases with recovery effects can significantly improve our EM-induced lifetime ($8.6X$ longer compared to the baseline results, which are shown in the third column in Fig. 5.8). However, the costs are the $2.0X$ more energy consumption ($1.9J/tasks$ vs $0.94j/tasks$) and $3.3X$ performance degradations ($0.05s/tasks$ vs $0.014s/tasks$). But this is still a better trade off for the higher EM lifetime (64.7 years and 78.1 years for $T_{recovery} = 50s$ and $1000s$, respectively) compared with the baseline case in Fig. 5.8.

5.4 Summary

In this chapter, we developed a new dynamic reliability management (DRM) techniques for emerging near-threshold dark silicon manycore microprocessors considering electromigraion (EM) reliability. To leverage the EM recovery effects, which was ignored in the past, at the system level, we developed a new equivalent DC current model to consider recovery effects for general time-varying current waveforms so that existing compact EM model can be applied. The new EM current model allows EM recovery effects to be effectively considered at the system level for the first time.

To leverage the EM recovery effects, we considered the energy optimization problem for dark silicon manycore processors with Near-Threshold Voltage (NTV) capabilities considering EM reliability. We showed that the on-chip power consumptions have different impact on reliability. The resulting optimization problem was solved with State-Action-Reward-State-Action (SARSA) reinforcement learning algorithm to optimization the near-threshold dark silicon cores' voltage policy to minimize energy considering reliability. Experimental results on a 64-core near-threshold dark silicon processor showed that the new equivalent EM DC currents model was able to fully exhibit the recovery effects at the system level so that trade-off between EM lifetime and energy/performance were easily made. We further showed that the proposed learning-based energy optimization can effectively manage and optimize energy subject to reliability, given power budget and performance limits.

Chapter 6

Cross-layer modeling and optimization for electromigration-induced reliability

This chapter presents we develop a new approach for cross-layer EM-induced reliability modeling and optimization at physics, system and datacenter levels. We consider a recently proposed physics-based EM reliability model to predict the EM reliability of full-chip power grid networks for long-term failures. We show how the new physics-based dynamic EM model at the physics level can be abstracted at the system level and even at the datacenter level. Our datacenter system-level power model is based on the BigHouse simulator. To speed up the online optimization for energy in a datacenter, we develop a new combined datacenter power and reliability compact model using a learning based approach in which a feed-forward neural network is trained to predict energy and long term reliability

for each processor under datacenter scheduling and workloads. To optimize the energy and reliability of a datacenter, we apply the efficient adaptive Q-learning-based reinforcement learning method.

6.1 EM-induced reliability model for a manycore processor in datacenter

An existing EM model, including the new physics-based model, can only take a constant temperature. The previous study shows that system-level time-to-failure (TTF) or lifetime under different temperature can be approximated by [67]: EM reliability model for individual core can be expressed as follows

$$TTF_{i-core} = \frac{1}{(\sum_{k=1}^{n}(\Delta t_{i,k}\frac{1}{TTF_{i,k}}))/T} \tag{6.1}$$

where $TTF_{i,k}$ is the actual TTF under the k-th power and temperature settings for Δt_k period, assuming i-th core works through n different power and temperature settings and $T = \sum_{k=1}^{n} \Delta t_{i,k}$. Each $TTF_{i,k}$ will be computed based on the recently proposed physics-based EM model and assessment techniques [57].

A manycore processor lifetime can be defined as the shortest lifetime among the cores [119, 79]. The individual core lifetime can be obtained from (6.1). Recently, one study used *performability* as the ratio of number of non-failure cores over total number of cores [120] to explain chip multiprocessor (CMP). But the specific mechanism was not introduced and is too abstract, so we use the shortest lifetime in this chapter, however, our framework easily extends to support performability later.

6.2 EM-induced reliability-aware datacenter model

To evaluate datacenter-level EM-induced reliability, we use the BigHouse simulator [121], a simulation infrastructure for datacenter. BigHouse is based on stochastic queueing simulation, a validated methodology for simulating the power/performance behavior of datacenter workloads. The BigHouse simulator is widely used in academia, as well as in Google datacenter research [121, 122, 123].

BigHouse uses synthetic arrival/service traces that are generated through empirical interarrival and interservice distributions collected from real systems [121, 122]. We evaluate two major workloads, Domain Name Service (DNS) and Apache World Web Service (WWW), provided with the BigHouse simulator. These workloads are modeled by workloads distribution, which represents the average, standard deviation(σ), and coefficient of variation(C_v) for the interarrival and service time distributions of the workloads. The interarrival distribution is used to drive the queueing model, while the service time distribution is used for the service nodes. These synthetic arrival/service traces are fed into a discrete-event simulation of a G/G/k queueing system that models active and idle low-power modes through state-dependent service rates.

During simulation time, measures of interest, such as power consumption and 99-th percentile latency, are obtained by sampling the output of the simulation until each measurement reaches a normalized half-width 95% confidence interval of 5%. The simulation ends when the sample statistics are considered *converged*, that is, once the observed sample size is sufficient to achieve the desired confidence interval of 95%. The sample size required to achieve a certain confidence is given by:

$$N_m = \frac{Z_{1-\alpha/2}^2 * \sigma^2}{\epsilon^2} \quad (6.2)$$

where σ is the standard deviation of the samples, ϵ is the half-width of the desired confidence interval, and $Z_{1-\alpha}$ is the value of the standard normal distribution at the $(1-\alpha/2)^{th}$ quantile. For 95% confidence, this value is 1.96.

To explore the EM effect on datacenter-level reliability, we integrate the EM model into BigHouse simulator. Additionally, we added thermal modeling into BigHouse, and drive the EM model using power, voltage, and temperature measurements. The thermal modeling is achieved using the HotSpot thermal model [115]. This thermal model offers a compact solution with relatively good accuracy and speed. HotSpot is integrated into BigHouse and fed a power trace of each simulated core using the method described above. Each core is then modeled and simulated to produce a thermal trace. It is this thermal trace that is used as the temperature measurements for the EM model.

To explain server-level reliability on datacenters, we use average socket lifetime (Mean-time-to-failure, MTTF). One socket lifetime can be defined as the shortest lifetime among the processors in one server.

We use tail latency as most important service latency for the datacenter since the tail flow completion time (FCT), 99-th or 99.9-th percentile FCT, can be more than 10x larger than the mean FCT. So tail latency is a very crucial performance metric for datacenter as the service response needs to wait for the slowest flow/workload to complete [124].

6.3 New reliability-constrained energy optimizatoin for datacenter

In this section, we introduce new reliability-constrained energy optimization for datacenter. To speed up the online optimization for datacenter energy and reliability, we use feed-forward neural network (FNN), which is trained to predict energy and long-term reliability for each processor under datacenter scheduling and workloads. To further optimize energy and reliability of a datacenter model, we formulate a learning-based optimization method, Q-learning, as minimizing datacenter energy subject to reliability, given power budget and performance.

6.3.1 Neural networks for datacenter energy and reliability models

Review of feed forward neural network

To build a compact energy and reliability model for datacenter systems, learning based techniques such as neural network, which is composed of multiple processing layers, can learn representations of data with multiple levels of abstraction. Processor power consumption and EM-induced lifetime can be considered as supervised learning in neural networks. One advantage of neural networks is its wide applications for nonlinear systems. The universal approximation capability of feed-forward neural networks (FNN) has been proved to show that any Borel measurable function can be approximated with any arbitrary accuracy by an FNN using squashing activation functions [125]. If we have an input vector $\boldsymbol{u} = \{u_1, u_2, \cdots, u_p\}$ and an output vector $\boldsymbol{y} = \{y_1, y_2, \cdots, y_q\}$, then the layer-wise structured FNN without bias node has the form

$$
\begin{aligned}
&\boldsymbol{a}_1 = \boldsymbol{f}_1(\boldsymbol{u}\boldsymbol{W}^{(\mathrm{IN},1)}), \quad \boldsymbol{a}_2 = \boldsymbol{f}_2(\boldsymbol{a}_1\boldsymbol{W}^{(1,2)}), \quad \cdots, \\
&\boldsymbol{a}_i = \boldsymbol{f}_i(\boldsymbol{a}_i\boldsymbol{W}^{(i-1,i)}), \quad \cdots, \quad \boldsymbol{y} = \boldsymbol{a}_k\boldsymbol{W}^{(k,\mathrm{OUT})}
\end{aligned}
\tag{6.3}
$$

where the activation function $\boldsymbol{f}$ is element-wise squashing operator such as a sigmoid or a hyperbolic tangent function; vector $\boldsymbol{a}_i$ is the intermediate activation result of each layer; $\boldsymbol{W}^{(\cdot,\cdot)}$ is the weighting matrix connecting adjacent layers. FNN with bias node requires each activation result vector $\boldsymbol{a}_i$ to be appended with a fixed unit value before it is fed into next level of calculation, and the dimensions of $\boldsymbol{W}^{(\cdot,\cdot)}$ also need to be adjusted accordingly.

Neural network training for datacenter reliability-aware energy model

As seen in (6.3), in theoretical aspect, training a neural network is equivalent to the optimization problem to minimize cost function (without bias node or connections, without regulation terms):

$$
\boldsymbol{J}\left(\boldsymbol{W}^{(\mathrm{IN},1)}, \boldsymbol{W}^{(1,2)}, \cdots, \boldsymbol{W}^{(k,\mathrm{OUT})}\right) = \sum_{j=1}^{m} \|\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\| \tag{6.4}
$$

where $\hat{\boldsymbol{y}}_i$ is a neural network output which can be explicitly written in a nested activation form

$$
\begin{aligned}
\hat{\boldsymbol{y}} =&\boldsymbol{f}_k(\boldsymbol{f}_{k-1}(\boldsymbol{f}_{k-2}(\cdots \boldsymbol{f}_2(\boldsymbol{f}_1(\boldsymbol{u}\boldsymbol{W}^{(\mathrm{IN},1)})\boldsymbol{W}^{(1,2)})\boldsymbol{W}^{(2,3)}\cdots) \\
&\boldsymbol{W}^{(k-2,k-1)})\boldsymbol{W}^{(k-1,k)})\boldsymbol{W}^{(k,\mathrm{OUT})}
\end{aligned}
\tag{6.5}
$$

Therefore, the training problem of neural networks can be solved by applying existing optimization methods such as gradient descent, Broyden-Fletcher-Goldfarb-Shanno (BFGS)

algorithm [126], and the Quasi-Newton method with the cost function $\boldsymbol{J}$. In practice, an algorithm with lower computational cost, Back-propagation, has been widely used for solving the training problem [127].

Neural network structure and data configuration

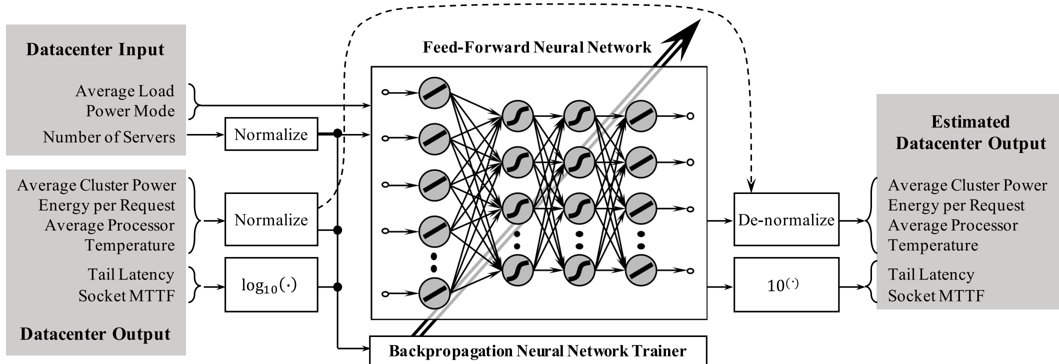


Figure 6.1: Feed-forward neural network structure and data configuration

As shown in Fig. 6.1, the feed-forward neural network (FNN) can be constructed to predict energy and long-term reliability for each processor under datacenter scheduling and workloads. We separately construct and train networks for each individual workloads. The inputs to the neural networks are average load rate, power mode (quantified), and a number of servers in the datacenter. With these inputs, the neural networks can estimate average cluster power, average processor temperature, tail latency, and average socket MTTF. To train the neural networks more efficiently with less numerical stability issues, the scaling of the inputs is required. Otherwise, it can have a large effect on the quality of the final solution. As shown in Fig. 6.1, the number of servers is normalized. The average load rate

can be used without scaling since it already has a good distribution. In the same way, the output data set can be scaled and converted into logarithmic scale since they are served as a part of the training input set in the back-propagation algorithm [127]. We use three hidden layers with sigmoid activation functions, with all layers having 15 nodes respectively. The input and output layer sizes are 3 and 5 respectively.

6.3.2 Q learning optimization for datacenter

State and action determination

Q-learning is a reinforcement learning method used as a controller to maximize long-term rewards. It can converge close to the optimal result of a state-action function for arbitrary policies while handling problems with stochastic transition [91]. In this case, state(s) used in this work consists of workload model, average load rate, power model, and a number of servers. Action (a) is used to describe transitions between two states. Executing an action in a certain state provides a learning agent contained in the model whose goal is to minimize penalty with updated Q-value by penalty calculation in the Q-table, also known as the state-action table. The environment part is reliability-aware BigHouse model, whose learning agent is Q-learning algorithm. The learning agent can obtain the environment state, calculate penalty function, and finally, decide the next action.

Q-value function and Q-learning process

In the Q-learning process, one critical issue is to define the Q-value function with penalty term. Each state s_i will determine average cluster power $Power(s_i)$, tail latency of datacenter $latency(s_i)$, the average processor temperature, $Temp(s_i)$. $EM_{min}(s_i)$, which is

defined as average socket MTTF in datacenter. $E(s_i)$ is energy per request in datacenter. An action, say $a_{i,j}$, can be viewed as the transition from state i to state j. The penalty function Q determines a penalty and a new state which is related to the previous state and selected action. Q-value is updated at every step Δt.

$$\begin{aligned} Q^{t+1}(s(t), a(t)) = Q^t(s(t), a(t)) + \\ \alpha(t) \times \Big(PT(t+1) + \gamma \min_a(\forall Q^t(s(t+1), a)) - Q^t(s(t), a(t))\Big) \end{aligned} \tag{6.6}$$

where $\alpha(t)$ is learning rate between 0 and 1 which determines the percentage of newly calculated Q-value applied. $s(t+1)$ is determined by action $a(t)$, and $Q^t(s(t+1), a)$ are all possible action's Q-values from future state. The discount factor γ (between 0 and 1) determines the importance of future penalty. $\min(\forall Q^t s(t+1), a)$ is considered to be estimated optimal future value. The difference between old Q-value (Q^t) and learned value ($PT(t+1)+\gamma\min_a(\forall Q^t(s(t+1), a))$) updates the new Q-value (Q^{t+1}) with the learning rate. A penalty term (PT) is shown in (6.7). PT_E is a penalty term for total datacenter energy, PT_{EM} is a penalty term for average socket MTTF, PT_{power} for average cluster power, PT_{temp} for average processor temperature, and $PT_{latency}$ is tail latency of datacenter. Each penalty term (PT_x) is normalized in (6.7). This feature scaling method to bring all values between 0 and 1. For instance, $PT_E = \frac{E(t+1)-E(t)}{E_{Max}-E_{Min}}$ is for energy related penalty, where $E(t)$ is the energy per request in the previous time t and $E(t+1)$ is energy per request of the datacenter at current time $t+1$. For the EM MTTF, $PT_{EM} = \frac{MTTF(t)-MTTF(t+1)}{MTTF_{Max}-MTTF_{Min}}$ is for EM related penalty where $MTTF(t)$ is the average socket MTTF of the datacenter for EM-induced in the previous time t and $MTTF(t+1)$ is the average socket MTTF of the datacenter at current time $t+1$.

$$PT = PT_E + C \sum_{x=\{EM,power,temp,latency\}} \delta_x PT_x$$
$$\delta_x = \begin{cases} 0 & \text{if } PT_x \leq B_x + \Delta_x \\ 1 & \text{if } PT_x > B_x + \Delta_x \end{cases} \tag{6.7}$$

where δ_x is a binary function to active ($\delta_x = 1$) or inactive ($\delta_x = 0$) of user defined or given constraint bounds, B_{power}, $B_{latency}$, and B_{temp} in the penalty term. They are also normalized average cluster power, tail latency, average processor temperature bounds respectively. Each Δ_x is the difference between each bound and average penalty (PT) for the power, latency, and temperature. Δ_x is positive if the whole datacenter violated the given constraint, otherwise, it is negative which means the system is bounded and working in acceptable condition. If the datacenter violated user's constraints in the past, penalty would be significant due to large value for constant C in (6.7).

The proposed learning-based energy optimization algorithm goes with the following flow: First, all the Q-values in the Q-table are initialized to zero. Current state $s(t)$ finds an action $a(t)$ with the lowest Q^t in (6.6) and switches to next state corresponding to input values. For every step, average socket MTTF, latency, average processor temperature, average cluster power, and energy per request are evaluated and thus, all environments can be updated. Then, new corresponding penalty $PT(t+1)$ would be calculated in (6.7) and Q^{t+1} would be updated (learning process). After the update, the current state could be replaced by a new action and it would iterate with a newly updated state. Finally, when all the Q-value changes are less than a certain threshold, the best policy will be chosen based on the result.

6.3.3 Proposed new datacenter framework for energy and reliability

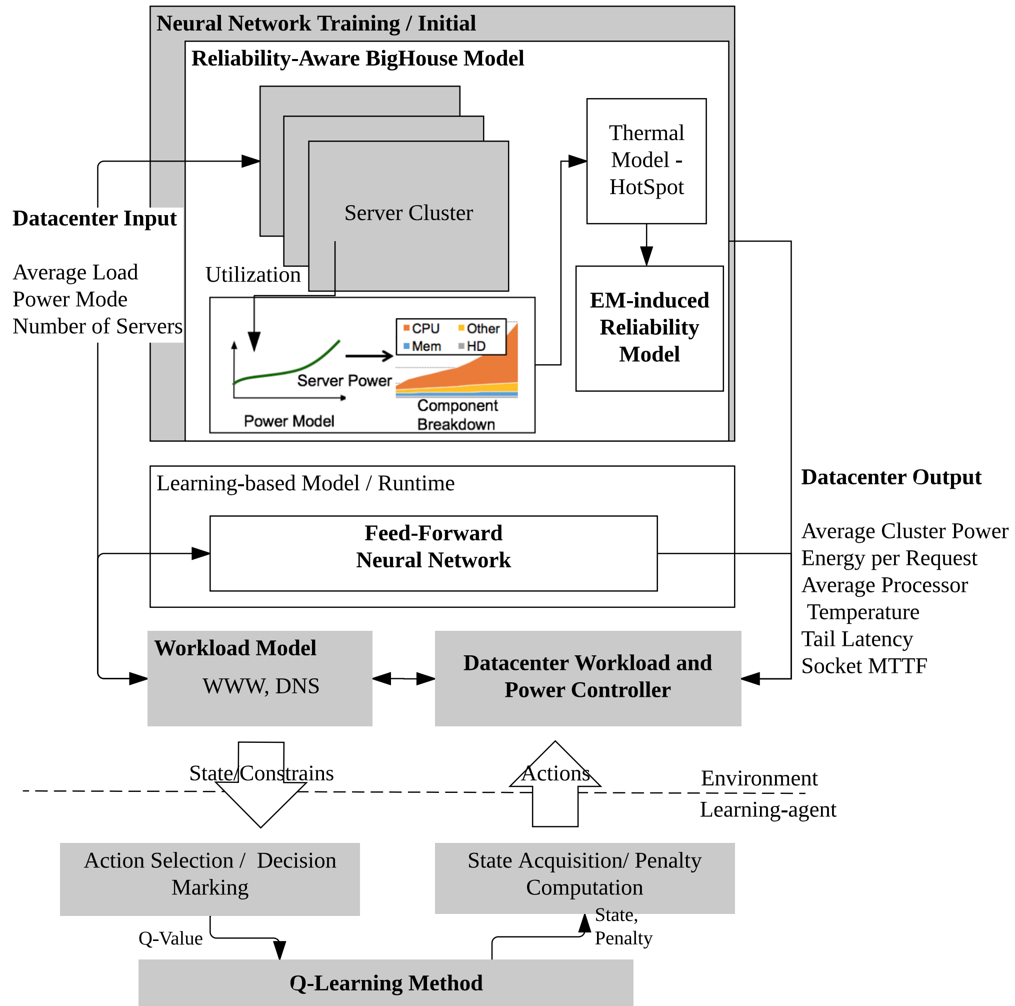


Figure 6.2: The evaluation platform for datacenter and energy and reliability management algorithms

To evaluate the proposed new reliability-constrained energy optimizatoin for datacenter, we use BigHouse model, which can provide cluster power and performance models with different datacenter scheduling and workloads, such as average load rate, power model (Low, Mid, and High), a number of active servers in datacenter. Once BigHouse model gen-

erate the performance and power traces of each core in the server of datacenter, HotSpot can generate each core's temperature from the power traces. For the EM-induced reliability, we use the power traces, the thermal traces, and the core's voltage as input to generate the individual core EM-induced lifetime of a manycore processor. As explained in Section 6.1, system-level (a processor) EM-induced lifetime can be calculated. For the datacenter level lifetime, we use average processor MTTF.

The training data can be obtained from BigHouse simulator with all the possible datacenter scheduling and workloads to train the neural network to speed up online optimization for datacenter power and reliability model. With the trained network, Q-learning method can find the optimal policies for datacenter scheduling and workloads to achieve minimizing energy subject to given reliability, power performance constraints as seen in Fig. 6.2.

6.4 Numerical results and discussions

6.4.1 Experimental setup

The proposed new compact model (FNN-based) and optimization (Q-Learning) for the datacenter framework have been implemented in Python 2.7.9 with the numerical libraries (NumPy 1.9.2 and Scipy 0.15.1). Thermal model (HotSpot 6.0 [115]) to estimate EM-induced lifetime. BigHouse utilizes a simple system-level power model, as shown in figure 6.2, which takes in a server utilization and outputs the power consumption of each server. Two major workloads (DNS and WWW) have been used to evaluate our proposed models. The server power model is based on a highly energy proportional server (Huawei

XH320) derived from reported SPECpower benchmark results [128]. Our EM model requires per core energy. In order to extract per core energy, we instrumented a high energy proportional server to measure per-component power, with component breakdown as shown in Fig. 6.2.

We instrumented each individual component by intercepting the power rails and measuring the current with LTS 25-NP current sensors. The outputs of the current sensors are sampled at 1kHz using a DAQ and logged using LabView. To measure CPU power, we inserted a current sensor in series with the 4-pin ATX power connector. To measure memory power, we inserted a current sensor in series with pins 10 and 12 of the 24-pin ATX power connector which supplies power to the mother board. To measure the power of the hard drive, we inserted a current sensor in series with the hard drive backplane power connector. We use the per-component power breakdown to derive the per core power from the server.

6.4.2 Evaluations of proposed new modeling and optimization

First we evaluate our learning-based datacenter modeling (see Section 6.2) We get normalized root mean square error (RMSE) by calculating $\frac{1}{\max(y_{\text{ref}})-\min(y_{\text{ref}})}\sqrt{\frac{1}{n}\sum(y_{\text{est}}-y_{\text{ref}})^2}$, where y_{ref} and y_{est} are obtained from the reliability-aware BigHouse model (reference) and FNN-based model (estimated), respectively. TABLE 6.1 shows each training error and validation error of the proposed compact model. In validation phase, both estimations have good accuracy on DNS and Web datacenter workloads, where RMSEs are lower than 10%.

Second, we evaluate our learning-based optimization method (see Section 6.3.2) by optimizing for energy savings with different sets of average processor MTTF, average

Table 6.1: Accuracy analysis (RMSE) of the feed-forward neural network (FNN) model

	Training Error		**Validation Error**	
	DNS	**WWW**	**DNS**	**WWW**
Tail latency	3.97%	6.53%	2.83%	9.37%
Avg. cluster power	2.64%	2.45%	3.02%	3.50%
Avg. proc. temp.	0.549%	2.91%	0.497%	2.92%
Avg. proc. MTTF	5.59%	6.78%	5.70%	7.40%
Energy per request	0.671%	0.738%	1.57%	1.20%

Table 6.2: Energy optimization for datacenter

	Energy per Request (J)	**Energy Saving (%)**
Max State (DNS)	67.63	
Case 1 (DNS)	18.76	72.25
Case 2 (DNS)	24.08	64.39
Case 3 (DNS)	35.04	48.18
Max State (WWW)	23.71	
Case 4 (WWW)	8.44	64.37
Case 5 (WWW)	8.44	64.37
Case 6 (WWW)	12.25	49.30

cluster power, and tail latency. Table. 6.2 and Fig. 6.3 shows the energy savings given constraint for average processor MTTF , average cluster power and tail latency, with DNS and WWW workload on the proposed datacenter framework. As we can see, energy savings for the different constraints have been evaluated in Fig. 6.3, case 1-3 is DNS workload and case 4-6 is WWW workload with tight MTTF constraints (case 1 and 4) and loose MTTF constraints (case 3 and 6). In Table. 6.2, our method finds relatively high energy savings for each case.

6.5 Summary

In this chapter, we developed a novel cross-layer approach to optimizing the energy of a datacenter subject to long-term reliability and performance constraints. We considered a recently proposed physics-based EM reliability model to predict the EM reliability of full-chip power grid networks for long-term failures. We showed how the new physics-based dynamic EM model at the physic level can be abstracted at the system level and even at in a datacenter level. To speed up the online optimization for energy for datacenter, we developed a new combined datacenter power and reliability model using a learning based approach in which a feed-forward neural network (FNN) was trained to predict energy and long term reliability for each processor under datacenter scheduling and workloads. To optimize the energy and reliability of a datacenter model, we applied the Q-learning based reinforcement learning method. Experimental results showed that the proposed compact models for the datacenter system trained with different workloads under different cluster power modes and scheduling policies are able to build accurate energy and lifetime.

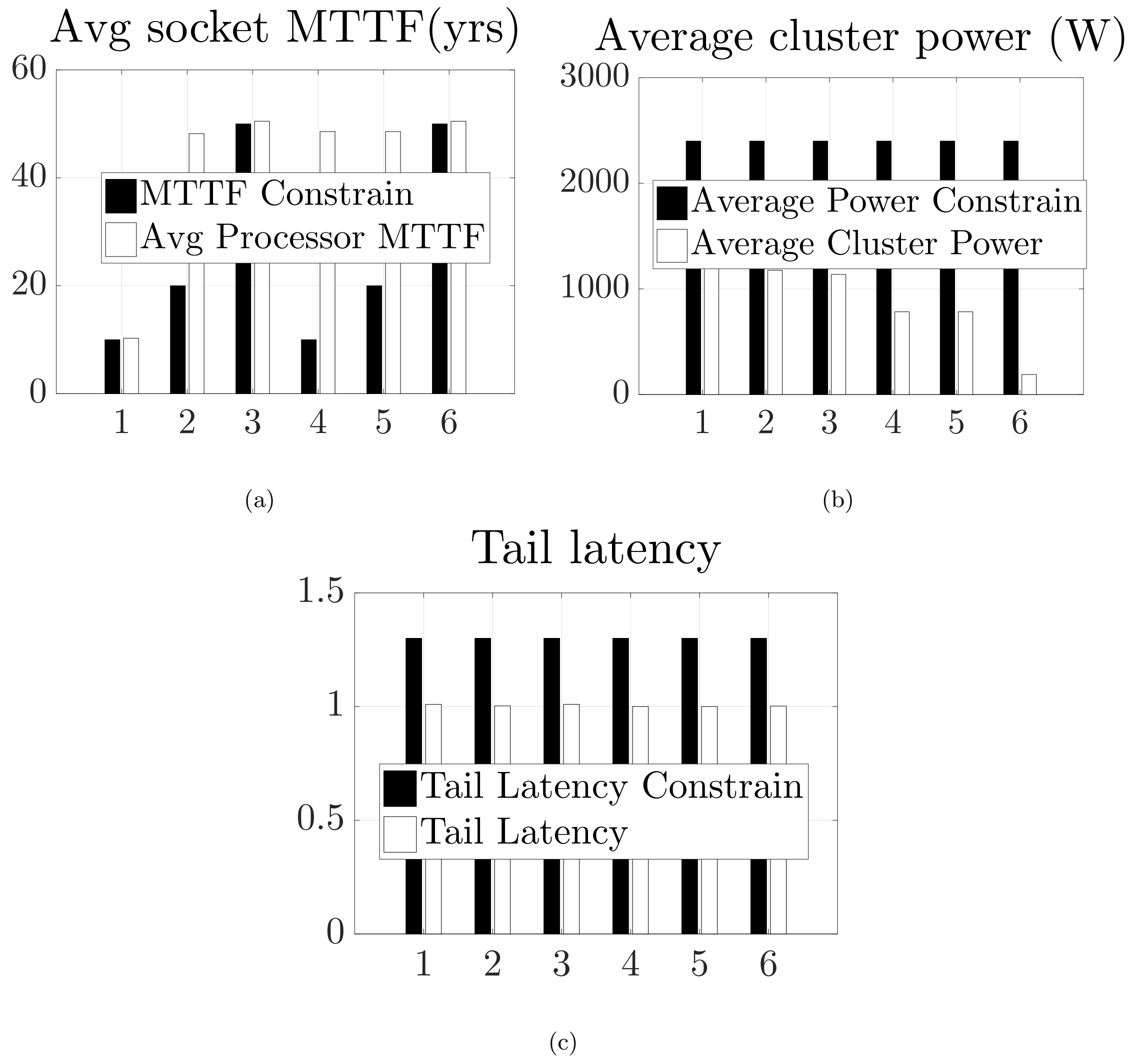


Figure 6.3: Validating violations with constraint limits (a) Average socket MTTF (b) Average cluster power (c) Tail latency

Chapter 7

Long-term reliability management for multitasking GPGPUs

This chapter presents a new long-term reliability management for GPU architectures using spatial multitasking. For GPU scheduling, we mainly focus on spatial multitasking, which allows GPU computing resources to be partitioned among multiple applications. We find that the existing reliability-agnostic thread block scheduler for spatial multitasking is effective in achieving high GPU utilization, but poor in reliability. We develop and implement a long-term reliability-aware thread block scheduler in GPGPU-sim, and compare it against the existing reliablity-agnostic scheduler.

7.1 System-level reliability resource consumption model

Given the new physics-based EM model, we now introduce our system level EM-reliability resource consumption model. Based on the EM model discussed in the previous

section, instead of using the EM-induced stress and resistance values, we can view the interconnect time to failure (TTF) as a resource. TTF occurs when the interconnect tree resistance has significant change due to the EM process so that the resulting voltage drops in the power delivery network exceed 12% [56] (or other predefined value). Once electrical current starts to flow through the wire, the EM process starts to spend the resource at a rate, which is a function of the temperature and current density. We notice that treating the EM as a resource was first introduced in [67]. But this work is still based on the traditional Black's equation.

Specifically, the reliability of a component is a probability function $R(t)$, defined on the interval $[0, \inf]$, that the component (system) operates correctly with no repair up to time t. The failure rate of a component, λ_f, is the conditional probability that the component (or system) fails in the interval $[t, t + \Delta t]$ while assuming correct operation up to time t. The *mean time to failure MTTF* is the expected time at which a component fails, i.e. $MTTF = \int R(t)dt$. If the failure rate λ_f is constant with time, then MTTF is $\frac{1}{\lambda_f}$ and the reliability is $R(t) = e^{-\lambda_f t}$. In general, failure rates are time dependent. In the following, the TTF is still used instead of MTTF as explained before.

Let us assume that we have a set of different time intervals Δp_k characterized by different workloads in terms of current density j_k and temperature T_k for a processor or a core. It means that $P = \sum_{k=1}^{n} \Delta p_k$ is the total execution time. Each kth workload, if it lasts till imaginary failure, provides time to failure TTF_k. Thus the failure rate at the kth workload, which last Δp_k is $\lambda_k = 1/TTF_k$. Then the average failure rate for the considered set of work loads can be expressed as follows.

$$\lambda_{avg} = \sum_{k=1}^{n} \frac{\Delta p_k}{\sum_{j=1}^{n} \Delta p_j} \lambda_k = \sum_{k=1}^{n} \frac{\Delta p_k}{P} \lambda_k \tag{7.1}$$

As a result, the expected time to failure or average lifetime of the whole processor, TTF is [67],

$$TTF = \frac{1}{\lambda_{avg}} = \frac{1}{(\sum_{k=1}^{n} (\Delta p_k \frac{1}{TTF_k}))/P} \tag{7.2}$$

Based on the (7.2), we can treat the lifetime of the processor specified by TTF as a resource that could be consumed as the SM works. We first define the specified TTF as a nominal value, denoted as TTF_N, which is the intended or required life of the SM under a typical temperature and power setting for a core or system. For example, one microprocessor has nominal TTF of 10 years under 70^oC and power of $20W$ as a specification. However, in reality, TTF varies under different temperature and power settings. For the kth workload, its time to failure is TTF_k and the overall time to failure for the whole set of workloads is given in (7.2).

In reality, depending on different workload settings, the consumption rate could be either higher or lower than its nominal rate, and we define *consumption rate* for workload k as

$$cr_k = \frac{TTF_N}{TTF_k} \tag{7.3}$$

in which the lifetime in real case (TTF_k under the kth workload) could be estimated by the new proposed reliability model in the previous sub-sections. In the nominal case, the SM is working under its specified temperature and power setting, and it has lifetime given by

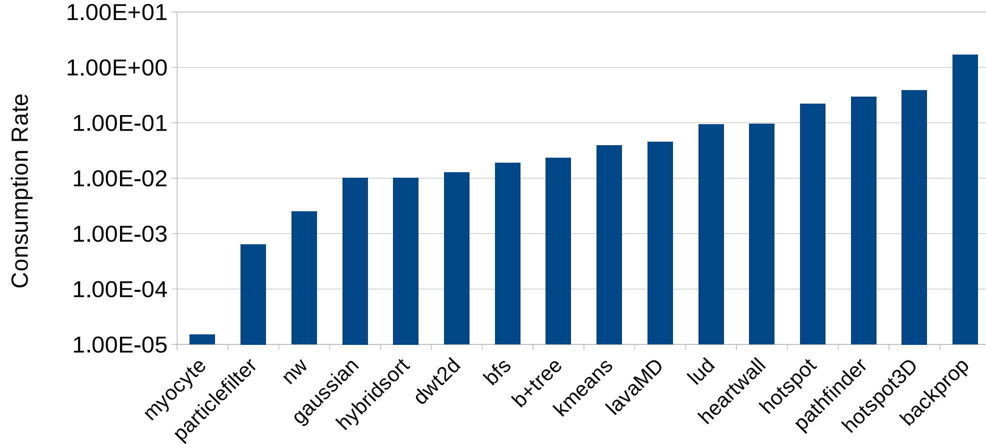


Figure 7.1: Consumption Rate of Rodina Benchmarks

TTF_N. Hence, the amount of lifetime consumed by the SM in each second is 1 EM second, that is to say, the nominal average consumption rate is $cr_N = 1$. If $TTF_k > TTF_N$, then $cr_k < cr_N$, which indicates that the SM is consuming its nominal lifetime at a lower rate, and thus the real lifetime is longer than the nominal one. Conversely, if $TTF_k < TTF_N$, then $cr_k > cr_N$, which indicates that the SM is consuming its nominal lifetime at the higher rate, and thus the real lifetime is shorter than the nominal one. Hence, instead of saying TTF changes, we perceive TTF as a constant resource, which is given by TTF_N, and (7.3) is the consumption rate (cr_k) of TTF that determines the real lifetime of the SM. If the time integration of EM slacks over a period is zero, then the lifetime or TTF of the core during that period will the TTF_N as predicted by (7.2).

In figure 7.1 we show the consumption rate of a range of common GPU applications. Clearly, there is a wide range of consumption rate behavior across benchmarks. As we will show later, when running applications concurrently, this imbalance in consumption rate can lead to premature failures of GPUs.

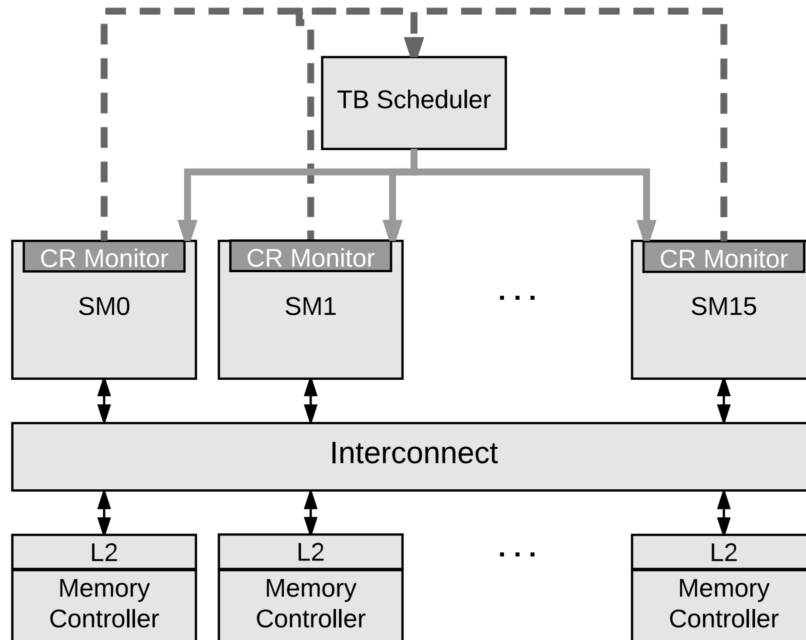


Figure 7.2: GPU Architecture

According to the definition of average TTF consumption rate defined by (7.3), if $cr_k > cr_N$ persistently, it will introduce excessive consumption of TTF, which would possibly lead to early failure of the core if no compensation is made. In real application, it is common that $cr_k > cr_N$ during the period when heavy tasks are assigned to the core, and the lifetime is excessively consumed during this period, while on the other hand, when light tasks are assigned to the core, $cr_k < cr_N$, and less lifetime is consumed during this period.

7.2 GPGPU Architecture and Stream Multiprocessor Scheduling

Figure 7.2 provides an overview of the GPU architecture in this work. This work uses a Nvidia GTX480 Fermi GPGPU as the baseline architecture. The GTX480 consists of a set of 15 Streaming multiprocessors (SMs). Each SM is comprised of a 128KB register file, 2 warp schedulers. The SM core clock is 700MHz, with each SM with two SP execution units, one SFU, and 16 LDST unit per SM. Each SP unit contains 16 double frequency CUDA cores, each with individual integer and floating point pipelines (total of 32 CUDA cores per SM).

Each SM has its own 64KB shared memory and L1 cache. Fermi supports up to 48 active warps per SM. Each warp comprises of 32 threads executing in a lockstep manner, also called Single Instruction Multiple Thread (SIMT) execution model. There are a total of 1,536 active threads per SM. The SMs are all connected to an interconnection network which leads to 6 memory partitions, containing a shared L2 cache and memory controller.

We evaluate the GPU using GPGPU-Sim [129] which models this GTX480-like architecture. The default warp scheduler is the greedy-than-oldest (GTO) warp scheduler with 48 warps per SM and capable of issuing a total of two warps per cycle per SM. The GTO scheduler greedily schedules from a single warp until that warp reaches a long-latency instruction, such as memory load. At which point, the warp scheduler will then issue from the next oldest warp. The goal of the GTO scheduler is to improve L1 cache locality by giving priority to a single warp during execution. To estimate power, we utilized GPUWattch [130]. In our experiments, we observed the total on-chip power of 95W (in-

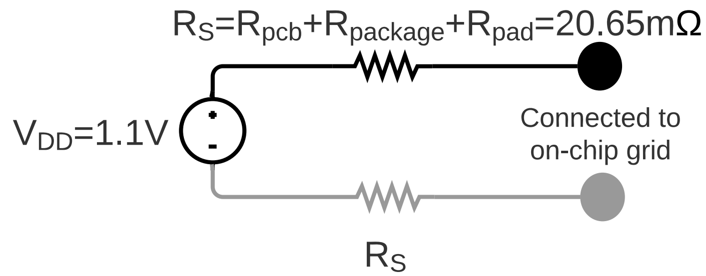
cluding SM, interconnect, and caches) with per SM peak power of 7W. We also simulate the temperature of the chip by utilizing HotSpot [131] using a GTX480 floorplan from [132].

GPU applications are processed on the GPU as CUDA contexts that consist of kernels. Kernels are further broken up into thread blocks, which are scheduled to individual SMs. These thread blocks are further broken up into warps, which are executed upon. GPUs are able to run multiple independent kernels from the same application simultaneously, or from multiple applications through multi-process service [133]. This enables spatial multitasking [54] where applications can run concurrently on different SMs to improve the utilization and efficiency of the GPU.

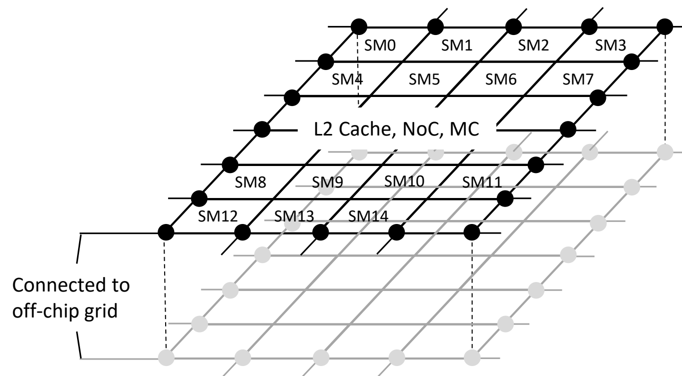
The goal of this work is to optimize the GPU for long-term reliability. In order to achieve this, we add a lightweight on-chip EM-aging sensor [101] to each SM. This EM-based aging sensor exploits the natural aging/failure mechanism of interconnect wires to time the aging of the chip. Compared with existing aging sensors, this sensor provides a more accurate prediction of the chip usage time at smaller area footprints due to its simple structure. By utilizing the EM-aging sensor, we estimate the reliability consumption rate (cr_k) of each SM in order to make fine-grain scheduling decisions.

7.3 Simulation Framework for EM Assessment on GPGPU

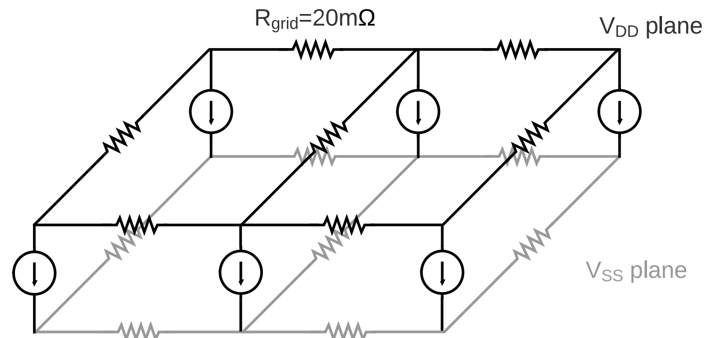
For the EM assessment of GPGPUs, PDN should be designed and simulated as discussed in 2.1.2. The PDN consists of two parts, off-chip network and on-chip network. The off-chip part is shown in Fig 7.3(a). It takes account of the resistors between on-chip PDN and the power source, which are the resistors of PCB, package, and pad. The on-chip



(a)



(b)



(c)

Figure 7.3: Off-chip PDN (a), On-chip PDN (b), and details of PDN of each SM (c)

part is built based on the dimensions of the publicly available specification and die photo for GTX480, same with the approach used in [134, 56]. The description explains that the

L2 cache, Network on Chip (NoC), and memory controllers (MC), which are located in the middle take approximately the area of 8 SMs. Thus, we split the PDN for the whole chip to 24 sections (4 horizontally and 6 vertically), which are shown in Fig 7.3(b). The 20 grid points on the edges, which are emphasized by dots, are connected to off-chip PDN respectively. Additionally, it is also shown in the die photo that the aspect ratio of each SM is about 2. Based on the fact that resistors of interconnects are proportional to their length, we model the PDN for each SM by a simple network consisting of 12 resistors shown in Fig 7.3(c). Besides the 12 resistors for VDD and GND networks, there are 6 current sources connecting two corresponding grid points in the two networks, which model the current that flows through the SM. We assume that the overall current of each SM is evenly distributed to these six current sources. For the current flowing through L2 cache, NoC and MC, it is modeled by 15 evenly distributed current sources as this part is modeled by 15 grid points. For the grid points at the border of two, or four SMs (or L2 cache, NoC, MC), the value of the current source is calculated by summing the two, or four SMs' currents, and then as stated above, dividing by 6. It is assumed in GPGPU-Sim that the operation voltage for the chip is $1V$, so we can easily get the current that flows through each part from the simulated power trace by GPGPU-Sim.

The EM reliability model is implemented in PDN simulator written in C++ with IT++ linear algebra library, which can parse RC power grid network. For EM model, the parameters used for the EM reliability model are listed in table 7.1 [4].

Figure 7.4 shows an overview of our long-term reliability simulation framework for GPUs. Benchmarks are run on GPGPU-Sim, in which activity traces are fed into

Table 7.1: Parameters used in EM analysis

Paras	**Value**	**Paras**	**Value**	**Paras**	**Value**
E_V	$0.674eV$	E_D	$0.65eV$	E_A	$0.86eV$
k_B	$1.38e-23J/K$	D_0	$7.56e-5m^2/s$	B	$1e11Pa$
ρ	$3e-8\Omega/m$	σ_{CR}	$5e8Pa$	Ω	$1.66e-29m^3$
f	0.6	Z	10		

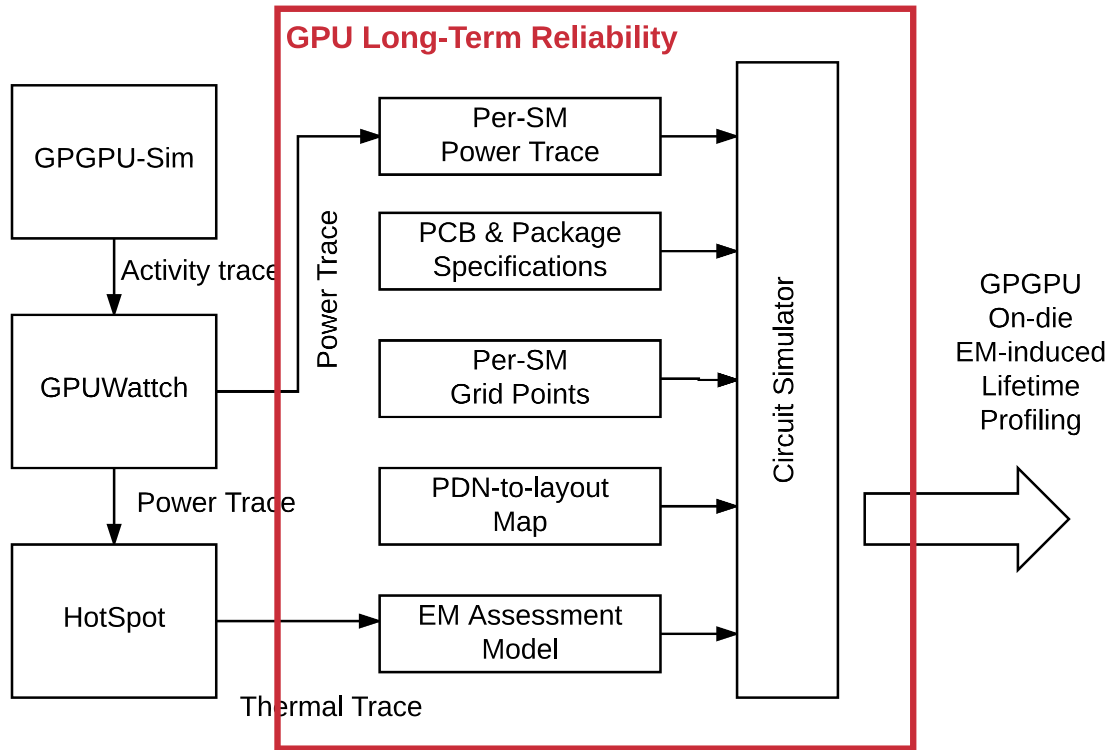


Figure 7.4: Simulation framework for long-term reliability assessment on GPGPU

GPUWattch. GPUWattch extracts a power trace with a sampling rate of 500 cycles. This power trace is then fed into HotSpot in order to derive a thermal trace of the GPU. Both the power trace and the thermal trace are fed into the long-term reliability framework,

which simulates the PDN along with the EM to output the expected lifetime and resource consumption rate of the GPU.

7.4 Resource consumption rate-aware thread block scheduler for long-term reliability

In spatial multitasking, we assume that it is up to the user to specify or identify the number of SMs allocated for each application. For example, prior work [54] utilizes compile-time application profiling or programmer input to partition applications among SMs. Finding the optimal SM partitioning is beyond the scope of this work. Instead, for a given application partitioning, our goal is to schedule applications to SMs in a way to improve long-term reliability.

As shown in figure 7.1, the consumption rate (cr_k) of benchmarks varies greatly. Under spatial multitasking, the difference among the consumption rates of co-scheduled applications can lead to significant wear imbalance across the SMs. As we consider the system-level EM reliability, we focus on the lifetime of the entire GPU, instead of individual SMs, and use the shortest lifetime among all the lifetimes of SMs as a metric [79]. Therefore, in a spatial multitasking environment with a mix of high and low consumption-rate applications, the goal is to balance the wear-out of all SMs. To do this, we place consumption rate monitors into each SM, and feed this monitoring information to the thread block scheduler.

We have made two key observations for the long-term reliability of GPUs: (i) the computational performance of applications depends only on the number of SMs allocated,

not on the exact indices of the SMs, and (ii) wear-out balancing can be easily achieved by rotating applications across SMs. Based on these observations, we find that shifting each application's designated SM subset by one is a simple but effective balancing method. For example, suppose that there are two GPU applications, `app1` and `app2`, and the GPU has 15 SMs. Assume that `app1` is assigned 5 SMs and `app2` is assigned 10 SMs. Initially, `app1` and `app2` are mapped to (SM0-4) and (SM5-15), respectively. When rebalancing is triggered, the SMs of each application are shifted by one, which remaps `app1` and `app2` to (SM1-5) and (SM0,SM6-15), respectively. When the thread block scheduler swaps the target SMs for each application's kernel, we ensure that the already-executing thread blocks in an SM will continue until completion and will not be preempted or migrated in the middle of thread execution. This is due to that many of GPU architectures as yet do not support preemption at an arbitrary point and doing so causes high overhead for GPU context switching. The latter is particularly important because such overhead can diminish the benefit of spatial multitasking.

The main challenge in designing our consumption rate-aware thread block scheduler is in determining the triggering condition. To this end, we will explore the following thread block mapping schemes.

Baseline: In the baseline, by default the thread block scheduler utilizes a loose round-robin scheme. Therefore, once given an application's SM partitioning, the thread blocks of an application are scheduled to a static subset of SMs. Due to this mapping, the GPU's long-term reliability is directly tied to the consumption rate of the high consumption rate workload.

Fixed scheduling: In the fixed scheduling case, we trigger rotation after a fixed number of cycles has elapsed. For fixed scheduling, we evaluated various fixed period lengths depending to the runtime. However as we will see in later section, fixed scheduling can lead to sub-optimal consumption rate balancing due to its fixed nature and applications' dynamic behavior.

Monitoring-aware scheduling: In the monitoring-aware scheduling scheme we utilize the embedded EM sensor to make triggering decisions. The EM sensors enable us to detect and measure consumption rate. The monitoring-aware scheduler is given in algorithm 2.
We measure the accumulated consumption of each SM i (denoted as $AccumulatedCR_i$ in the algorithm), until the difference between the maximum and minimum accumulated consumption of two SMs exceed a certain threshold, *rotation threshold*. The index of each SM starts from 1 and ends at N; hence, the total number of SMs is N. We specify the threshold as a unit normalized to nominal average consumption rate. For example, if the threshold is 4, then we trigger rotation when the maximum and minimum consumption rates (CR_{max} and CR_{min}, respectively) differ by 4 or more. Upon triggering rotation, the accumulated consumption of each SM ($AccumulatedCR_i$) will be cleared. This algorithm ensures that the difference of consumption rate between the SMs does not diverge too greatly from each other, leading to a more balanced long-term reliability. Unlike fixed scheduling, the EM sensors are able to capture the dynamic application behaviors and their time-varying consumption rate usage.

Optimal scheduling: We compare against an optimal scheduling scheme that uniformly spreads out the thread blocks of applications across all SMs. This scheduling

Monitoring-aware Scheduling;

while *Application is running* **do**

  foreach SM_i *in all SMs* **do**

    $CR_i \leftarrow$ Current EM sensor reading for SM_i ;

    $AccumulatedCR_i \leftarrow AccumulatedCR_i + CR_i$;

  end

  $CR_{max} \leftarrow \max_{1 \leq i \leq N}(AccumulatedCR_i)$;

  $CR_{min} \leftarrow \min_{1 \leq i \leq N}(AccumulatedCR_i)$;

  if $CR_{max} - CR_{min} \geq threshold$ **then**

    Rotate application's SM subset by 1 ;

    $\forall i : 1 \leq i \leq N$, clear $AccumulatedCR_i$;

  else

    Schedule round-robin to current subset ;

  end

end

Algorithm 2: Monitoring-aware Scheduling

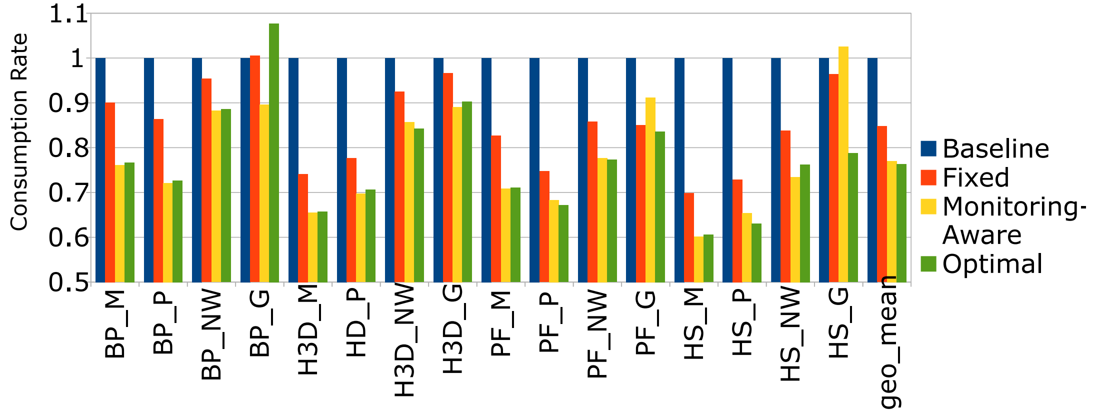


Figure 7.5: Consumption Rate of Rodina Benchmarks

policy does not take into account SM partitioning with the goal of maximizing consumption rate balance across SMs. This scheme essentially serves are our target goal for long-term reliability.

7.5 Numerical results and discussions

To evaluate our long-term reliability GPU approach, we utilize various benchmarks from the Rodinia [135] benchmark suite. The consumption rate of 16 Rodinia benchmarks was shown previously in figure 7.1. We group the Rodinia benchmarks into two categories; high consumption and low consumption. Our scheduler was tested using a subset of these groups which can be seen in Table 7.2. We selected 8 benchmarks consisting of the 4 highest and lowest consumption rates. We then mix the benchmarks from each group to evaluate a multitasked GPU running a high consumption and a low consumption application. This resulted in 16 mixes. Since each benchmark will run for different amounts of time, we repeat the short running job until the long running job completes, similar to [54]. As

Table 7.2: Benchmarks separated into Consumption Groups

High Consumption	**Low Consumption**
Backprop (BP)	Myocyte (M)
Hotspot3D (H3D)	Particlefilter (P)
Pathfinder (PF)	NW (NW)
Hotspot (HS)	Gaussian (G)

mentioned previously, our work does not perform SM partitioning among applications, rather, our goal is to minimize the consumption rate given a partition. For our mixes, we simply allocate 8 SMs for the higher consumption application and 7 SMs for the lower consumption application. In section 7.5.2, we explore how our proposed techniques fair under different partitioning.

7.5.1 Fixed Scheduling Performance

Figure 7.6 shows the results of our fixed scheduling policy. The x-axis shows the number of rotations triggered using fixed intervals, and the y-axis shows the consumption rate, normalized to the baseline. In this experiment, we choose the period to trigger as the total runtime of the longer application, divided by the x-axis. This figure demonstrates both the effectiveness of a fixed scheduling technique along with the sensitivity to the period length. The right-most data points show the optimal consumption rate with the given mix. Using our 16 mixed benchmarks, as shown in figure 7.6, when the frequency of rotations increase (and the rotation period decreases), the consumption rate of each benchmark gets

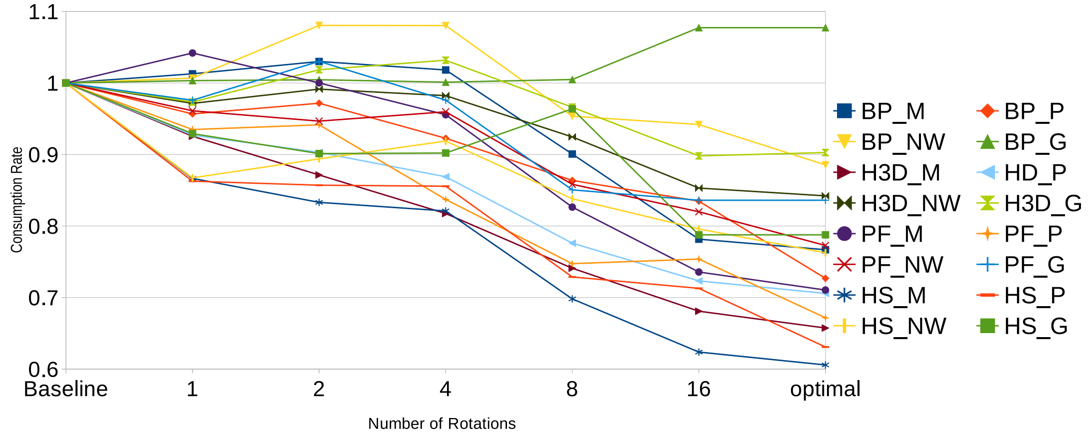


Figure 7.6: Normalized Consumption Rate with Fixed Scheduling Policy

closer to the optimal. Figure 7.5 shows that the fixed scheduler geometric mean has a 17.9% average improvement over the baseline.

7.5.2 Monitoring-Aware Scheduling Evaluation

In figure 7.5 we show the results of the baseline, fixed scheduling, monitor-aware scheduling, and optimal. Here we select a rotation threshold of 7 for our monitor-aware scheduler. We will later present a sensitivity analysis with our threshold value. Here, our monitoring-aware scheduler is able to reduce the consumption rate by 30%. Our scheme is able to come within 99% of the optimal scheme, which is able to reduce consumption rate by 31%.

7.5.3 Sensitivity to SM partitioning

Table 7.3 shows the consumption rate normalized to baseline of a Backprop-Myocyte mixed workload. The top row shows the ratio of SMs provisioned for Backprop

and Myocyte, respectively. This table shows several notable trends. First, reliability-aware scheduling is most effective when the number of SMs allocated to each application is relatively balanced, with a 11 to 4 ratio at most. Furthermore, it is also more effective if the low consumption rate application has more SMs allocated. This means that there are more lightly used SMs to balance out the heavier used application across. The trends highlighted here can further be included in SM partitioning optimizations to also account for reliability. In addition, this table utilized the fixed scheduling policy in order to demonstrate the variance in benefits due to the static policy. In many of the scenarios, the consumption rate impact is variable as the fixed rotation cycle length decrease. This further supports the need for a dynamic reliability monitoring aware scheduler.

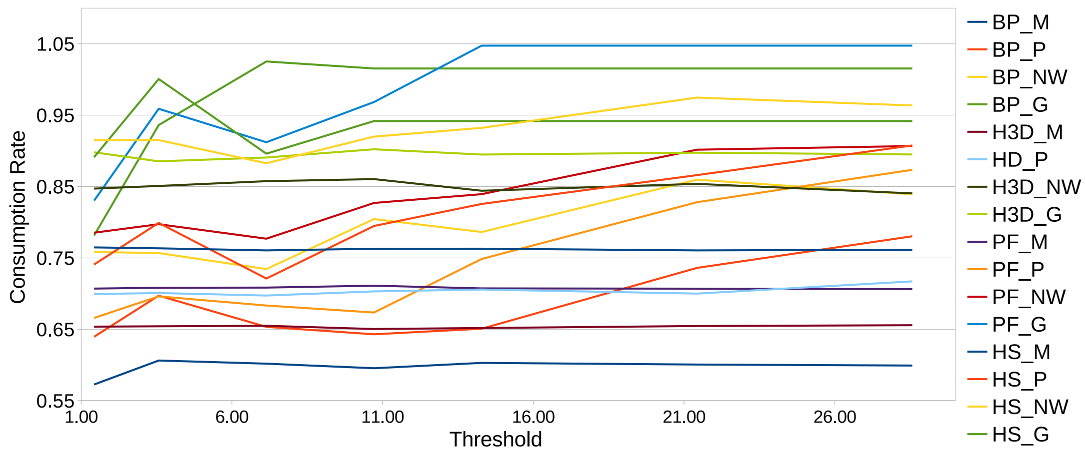


Figure 7.7: Sensitivity to rotation threshold for migration-aware scheduling

7.5.4 Threshold Exploration

In figure 7.7 we show the results of varying the rotation threshold with the migration-aware scheduler. The x-axis shows the threshold value and the y-axis show the consumption

Table 7.3: Sensitivity to SM partition for Backprop-Myocyte

	1:14	**2:13**	**3:12**	**4:11**	**5:10**	**6:9**	**7:8**
optimal	1.2079	1.1863	1.3825	1.3743	0.8122	0.6183	0.6796
16	1.2048	1.1587	1.2895	1.3066	0.8436	0.6796	0.6119
8	1.4017	1.4443	1.6696	1.6558	1.0514	0.8485	0.7456
4	1.2156	1.3135	1.5632	1.6358	1.0998	0.9383	0.8923
2	0.9252	0.845	1.1264	1.4427	1.1471	0.9082	0.7568
1	0.9982	0.9466	0.8854	1.2234	1.1695	1.0491	0.8823
baseline	1	1	1	1	1	1	1

	8:7	**9:6**	**10:5**	**11:4**	**12:3**	**13:2**	**14:1**
optimal	0.7669	0.6052	0.6121	0.5795	0.9831	1.0713	1.0238
16	0.7816	0.6497	0.6457	0.6443	0.9935	1.0884	1.0146
8	0.9007	0.7055	0.6987	0.6482	0.9895	1.1402	1.0661
4	1.0182	0.7943	0.8283	0.807	1.1017	1.1492	1.0587
2	1.03	0.8453	0.784	0.7849	1.0197	0.9904	0.9267
1	1.0128	0.9384	0.9416	0.8606	1.0728	1.0239	0.9087
baseline	1	1	1	1	1	1	1

rate normalized to the baseline. A smaller threshold leads to more rotations, therefore, leading to more balanced wear across the SMs. In general, as the threshold increases, the consumption rate also increases due to less rotation. The consumption rate typically begins increasing with a threshold value of 10. This value means that we will trigger rotation when the maximum and minimum consumption rate differs by 10 times the nominal consumption rate.

7.6 Summary

This chapter presented long-term reliability management for GPU architectures using spatial multitasking. We focused on electromigration (EM)-induced long-term failure of the GPU's power delivery network. A distributed power delivery network model at functional unit granularity were developed and used for our EM analysis in the GPU architecture. We considered a recently proposed physics-based EM reliability model and the EM-induced time-to-failure (TTF) at the GPU system level is modeled as a reliability resource. For GPU architecture, we majorly focused on spatial multitasking, which allows GPU resources to be partitioned among multiple applications simultaneously, which was recently introduced by Nvidia in their Pascal architecture. We found that the existing spatial multitasking scheduling can utilize GPU resources fully regarding performance, not reliability. We implemented long-term reliability-aware spatial multitasking in GPGPU-sim and compare it to the existing scheduling.

Chapter 8

Conclusion

Technology scaling has led to further processor integration, and reliability has become a more design challenge for the nanometer VLSI, especially more serious in 7nm technology. It is expected that future chips will show signs of reliability-induced aging much faster than the previous generations. Among of many reliability effects, electromigration (EM)-induced reliability has become a major design constraint due to the aggressive transistor and interconnect scaling and increasing power density. In this dissertation, I have developed many different new system-level EM-induced dynamic reliability managements from embedded system to high-performance systems. In this chapter, the main contributions of the thesis are summarized.

8.1 Summary of research contributions

8.1.1 Reliability-aware lifetime optimization for real-time embedded systems

Chapter 3 described new lifetime task optimization techniques for real-time embedded processors considering the electromigration-induced reliability. The new approach was based on a recently proposed physics-based electromigration (EM) model for more accurate EM assessment of a power grid network at the chip level. I applied the dynamic voltage and frequency scaling (DVFS) (by selecting the performance states or p-states of the tasks to manage the power) and thus the lifetime of the processor running different tasks over their periods. I considered both single-rate and multi-rate embedded systems with preemption. I explored to problem formulations and found the corresponding solutions with different solution qualities and computational costs. Experimental results have shown that for low utilization systems, significant reliability improvement can be achieved with even smaller power consumption than existing reliability-ignore scheduling method. I also compared the results from the two formulations and showed that the solutions given by the constrained nonlinear optimization method is close to the ones given by the MILP-based method, which is considered to be an optimal solution with regard to the proposed EM-induced reliability model and assumptions. The proposed methods can lead to near Pareto's front trade-off between the performance and the lifetime compared to the existing task scheduling method.

8.1.2 Learning-based reliability management and energy optimization for mary-core dark silicon processors

Chapter 4 presented a new energy and lifetime optimization technique for emerging dark silicon manycore microprocessors considering hard and soft errors. The new approach was based on a newly proposed physics-based electromigration (EM) reliability model to predict the EM reliability of full-chip power grid networks for hard error. DVFS-aware soft-error rate (SER) model and the Sum Of the Failure Rates (SOFR) method were employed for system-level soft-error model, which has been widely used to estimate microprocessor level soft errors. I employed both dynamic voltage and frequency scaling (DVFS) and dark silicon core state using On/Off pulsing action as the two control knobs. The impact on DVFS for hard and soft errors was investigated. I focused on two optimization techniques for improving lifetime and reducing energy. To optimize lifetime, we first applied the adaptive Q-learning based method, which was suitable for dynamic runtime operation as it was able to provide cost-effective yet good solutions. The second lifetime optimization approach was the mixed-integer linear programming (MILP) method, which typically yields better solutions but at higher computational costs. To optimize the energy of a dark silicon chip, we applied the Q-learning reinforcement learning method, which was suitable for our reliability management for the energy optimization considering hard and soft errors. Experimental results on a 64-core dark silicon chip showed that proposed methods work well for performance and lifetime optimizations considering the both soft and hard reliability constraints.

8.1.3 EM Recovery-aware dynamic reliability management for near-threshold dark silicon processors

In Chapter 5, I developed a new dynamic reliability management (DRM) techniques for emerging near-threshold dark silicon manycore microprocessors considering electromigraion (EM) reliability. To leverage the EM recovery effects, which was ignored in the past, at the system level, we developed a new equivalent DC current model to consider recovery effects for general time-varying current waveforms so that existing compact EM model can be applied. The new EM current model allows EM recovery effects to be effectively considered at the system level for the first time. To leverage the EM recovery effects, we considered the energy optimization problem for dark silicon manycore processors with Near-Threshold Voltage (NTV) capabilities considering EM reliability. I showed that the on-chip power consumptions have different impact on reliability. The resulting optimization problem was solved with State-Action-Reward-State-Action (SARSA) reinforcement learning algorithm to optimization the near-threshold dark silicon cores' voltage policy to minimize energy considering reliability. Experimental results on a 64-core near-threshold dark silicon processor showed that the new equivalent EM DC currents was able to fully exhibit the recovery effects at the system level so that trade-off between EM lifetime and energy/performance were easily made. I further showed that the proposed learning-based energy optimization can effectively manage and optimize energy subject to reliability, given power budget and performance limits. When the recovery effects were considered, the new optimization method was able to achieve $8.6X$ longer lifetime at the costs of $2.0X$ more energy and $3.3X$ more performance degradation.

8.1.4 Cross-layer modeling and optimization for EM-induced reliability in data center

In Chapter 6, I developed a novel cross-layer approach to optimizing the energy of a datacenter subject to long-term reliability and performance constraints. I considered a recently proposed physics-based electromigration (EM) reliability model to predict the EM reliability of full-chip power grid networks for long-term failures. I showed how the new physics-based dynamic EM model at the physic level can be abstracted at the system level and even at in a datacenter level. To speed up the online optimization for energy for datacenter, we investigated a new combined datacenter power and reliability model using a learning based approach in which a feed-forward neural network (FNN) was trained to predict energy and long term reliability for each processor under datacenter scheduling and workloads. To optimize the energy and reliability of a datacenter model, we applied the Q-learning based reinforcement learning method. Experimental results showed that the proposed compact models for the datacenter system trained with different workloads under different cluster power modes and scheduling policies were able to build accurate energy and lifetime. Moreover, the proposed optimization method effectively managed and optimized datacenter energy subject to reliability, given power budget and performance.

8.1.5 Long-term reliability management for multitasking GPGPUs

Lastly, in Chapter 7, I developed long-term reliability management for GPU architectures using spatial multitasking. I focused on electromigration (EM)-induced long-term failure of the GPU's power delivery network. A distributed power delivery network model

at functional unit granularity were developed and used for our EM analysis in the GPU architecture. I considered a recently proposed physics-based EM reliability model and the EM-induced time-to-failure (TTF) at the GPU system level was modeled as a reliability resource. For GPU architecture, we majorly focused on spatial multitasking, which allows GPU resources to be partitioned among multiple applications simultaneously, which was recently introduced by Nvidia in their Pascal architecture. I found that the existing spatial multitasking scheduling can utilize GPU resources fully regarding performance, not reliability. I implemented long-term reliability-aware spatial multitasking in GPGPU-sim and compare it to the existing scheduling. I evaluated several cases for partitioning GPU streaming multiprocessors (SMs) among parallel applications and find our proposed spatial multitasking scheduling shows a reliability improvement of up to 30%.

Bibliography

[1] "International technology roadmap for semiconductors (ITRS) interconnect, 2015 edition," 2015. http://public.itrs.net.

[2] X. Huang, V. Sukharev, J.-H. Choy, M. Chew, T. Kim, and S. X.-D. Tan, "Electromigration assessment for power grid networks considering temperature and thermal stress effects," *Integration, the VLSI Journal*, vol. 55, pp. 307 – 315, 2016.

[3] C. Cook, Z. Sun, T. Kim, and S. X.-D. Tan, "Finite difference time domain analysis of stress evolution and void growth for general interconnect wires," in *TECHCON*, Sept. 2016.

[4] X. Huang, A. Kteyan, X. Tan, and V. Sukharev, "Physics-based electromigration models and full-chip assessment for power grid networks," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 1848–1861, Nov. 2016.

[5] "Exascale computing initiative update, 2012 united states, department of energy," 2012. https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/aug12/2012-ECI-ASCAC-v4.pdf.

[6] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz, "Extending stability beyond cpu millennium: a micron-scale atomistic simulation of kelvin-helmholtz instability," in *Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on*, pp. 1–11, Nov 2007.

[7] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Scaling effects on neutron-induced soft error in SRAMs down to 22nm process," in *Third Workshop on Dependable and Secure Nanocomputing*, 2009.

[8] "Critical Reliability Challenges for The International Technology Roadmap for Semiconductors (ITRS)," 2003. In International Sematech Technology Transfer Document 03024377A-TR, 2003.

[9] "International technology roadmap for semiconductors (ITRS), 2015 edition," 2015. http://public.itrs.net.

[10] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Ramp: A model for Reliability Aware Microprocessor Design," *IBM Research Report*, 2003.

[11] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pp. 276–287, 2004.

[12] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *Computers, IEEE Transactions on*, vol. 53, pp. 584–600, May 2004.

[13] J.-J. Chen and T.-W. Kuo, "Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics," in *Parallel Processing, 2005. ICPP 2005. International Conference on*, pp. 13–20, June 2005.

[14] J.-J. Chen and T.-W. Kuo, "Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems.," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 289–294, Nov 2007.

[15] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, (New York, NY, USA), pp. 89–102, ACM, 2001.

[16] S. Saewong and R. Rajkumar, "Practical voltage-scaling for fixed-priority rt-systems," in *Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings. The 9th IEEE*, pp. 106–114, May 2003.

[17] C. Scordino and G. Lipari, "A resource reservation algorithm for power-aware scheduling of periodic and aperiodic real-time tasks," *Computers, IEEE Transactions on*, vol. 55, pp. 1509–1522, Dec 2006.

[18] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *Computers, IEEE Transactions on*, vol. 53, pp. 217–231, Feb 2004.

[19] O. Unsal, I. Koren, and C. Krishna, "Towards energy-aware software-based fault tolerance in real-time systems," in *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, pp. 124–129, 2002.

[20] L. Huang, F. Yuan, and Q. Xu, "On Task Allocation and Scheduling for Lifetime Extension of Platform-Based MPSoC Designs," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 2088–2099, 2011.

[21] T. Simunic, K. Mihic, and G. Micheli, *Optimization of Reliability and Power Consumption in Systems on a Chip*, vol. 3728 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[22] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," in *Proc. Design Automation Conf. (DAC)*, pp. 1057–1060, 2006.

[23] M. Basoglu, M. Orshansky, and M. Erez, "NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime," *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pp. 253–258, 2010.

[24] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini, "Workload and user experience-aware dynamic reliability management in multicore processors," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 2:1–2:6, ACM, 2013.

[25] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini, "A linux-governor based dynamic reliability manager for android mobile devices," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–4, March 2014.

[26] A. Calimera, E. Macii, and M. Poncino, "Energy-optimal SRAM supply voltage scheduling under lifetime and error constraints," in *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pp. 1–6, 2013.

[27] R. Dennard, F. Gaensslen, H. Yu, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, pp. 256–268, October 1974.

[28] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, (New York, NY, USA), pp. 365–376, ACM, 2011.

[29] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, pp. 33–38, July 2008.

[30] K. Chakraborty, *Over-provisioned Multicore Systems.* PhD thesis, University of Wisconsin-Madison, Madison, WI, USA, 2008. AAI3327881.

[31] S. Cho and R. Melhem, "Corollaries to amdahl's law for energy," *IEEE Comput. Archit. Lett.*, vol. 7, pp. 25–28, Jan. 2008.

[32] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, "Maestro: Orchestrating lifetime reliability in chip multiprocessors," in *Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers*, HiPEAC'10, (Berlin, Heidelberg), pp. 186–200, Springer-Verlag, 2010.

[33] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems," in *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, (New York, NY, USA), pp. 170:1–170:6, ACM, 2014.

[34] T. Kim, X. Huang, V. S. H.-B. CHen, and S. X.-D. Tan, "Learning-based dynamic reliability management for dark silicon processor considering EM effects," in *Proc. Design, Automation and Test In Europe. (DATE)*, Mar. 2016.

[35] Y. Tan, W. Liu, and Q. Qiu, "Adaptive power management using reinforcement learning," in *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, (New York, NY, USA), pp. 461–467, ACM, 2009.

[36] Y. Ge and Q. Qiu, "Dynamic thermal management for multimedia applications using machine learning," in *Proceedings of the 48th Design Automation Conference*, DAC '11, (New York, NY, USA), pp. 95–100, ACM, 2011.

[37] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, pp. 747–754, March 2012.

[38] R. Ye and Q. Xu, "Learning-based power management for multi-core processors via idle period manipulation," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp. 115–120, Jan 2012.

[39] "International technology roadmap for semiconductors (ITRS), 2014 update," 2014. http://public.itrs.net.

[40] C. Prasad, L. Jiang, D. Singh, M. Agostinelli, C. Auth, P. Bai, T. Eiles, J. Hicks, C. Jan, K. Mistry, S. Natarajan, B. Niu, P. Packan, D. Pantuso, I. Post, S. Ramey, A. Schmitz, B. Sell, S. Suthram, J. Thomas, C. Tsai, and P. Vandervoorn, "Self-heat reliability considerations on intel's 22nm tri-gate technology," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*, pp. 5D.1.1–5D.1.5, April 2013.

[41] H. Amrouch, V. M. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards interdependencies of aging mechanisms," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 478–485, Nov 2014.

[42] R. M. Swanson and J. D. Meindl, "Ion-implanted complementary mos transistors in low-voltage circuits," *IEEE Journal of Solid-State Circuits*, vol. 7, pp. 146–153, Apr 1972.

[43] M. Shafique, S. Garg, T. Mitra, S. Parameswaran, and J. Henkel, "Dark silicon as a challenge for hardware/software co-design: Invited special session paper," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, (New York, NY, USA), pp. 13:1–13:10, ACM, 2014.

[44] "2013 cost of data center outages," 2013. http://www.emersonnetworkpower.com.

[45] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, FAST '07, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2007.

[46] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: A large-scale field study," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, (New York, NY, USA), pp. 193–204, ACM, 2009.

[47] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, (New York, NY, USA), pp. 13–23, ACM, 2007.

[48] C. D. Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '14, (Washington, DC, USA), pp. 610–621, IEEE Computer Society, 2014.

[49] D. Tiwari, S. Gupta, G. Gallarno, J. Rogers, and D. Maxwell, "Reliability lessons learned from GPU experience with the Titan supercomputer at Oak Ridge leadership computing facility," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15*, pp. 1–12, 2015.

[50] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in GPGPU," in *CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*, pp. 691–696, IEEE, 2010.

[51] P. Rech, L. L. Pilla, P. O. A. Navaux, and L. Carro, "Impact of GPUs parallelism management on safety-critical and HPC Applications reliability," in *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 455–466, IEEE, jun 2014.

[52] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. Debardeleben, P. Navaux, L. Carro, and A. Bland, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture, HPCA 2015*, pp. 331–342, IEEE, feb 2015.

[53] P. Rech, L. Carro, N. Wang, T. Tsai, S. Kumar, S. Hari, and S. W. Keckler, "Measuring the Radiation Reliability of SRAM Structures in GPUs Designed for HPC," 2014.

[54] J. T. Adriaens, K. Compton, N. S. Kim, and M. J. Schulte, "The case for gpgpu spatial multitasking," in *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, HPCA '12, (Washington, DC, USA), pp. 1–12, IEEE Computer Society, 2012.

[55] S. P. Hau-Riege and C. V. Thompson, "Experimental characterization and modeling of the reliability of interconnect trees," *Journal of Applied Physics*, vol. 89, pp. 601–609, January 2001.

[56] J. Leng, Y. Zu, M. Rhu, M. Gupta, and V. J. Reddi, "Gpuvolt: Modeling and characterizing voltage noise in gpu architectures," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, (New York, NY, USA), pp. 141–146, ACM, 2014.

[57] X. Huang, T. Yu, V. Sukharev, and S. X.-D. Tan, "Physics-based electromigration assessment for power grid networks," in *Proc. Design Automation Conf. (DAC)*, June 2014.

[58] S. Biswas, M. Tiwari, T. Sherwood, L. Theogarajan, and F. T. Chong, "Fighting fire with fire: modeling the datacenter-scale effects of targeted superlattice thermal management," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pp. 331–340, IEEE, 2011.

[59] X. Huang, A. Kteyan, X. Tan, and V. Sukharev, "Physics-based Electromigration Models and Full-chip Assessment for Power Grid Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Feb. 2016. DOI.

[60] J. R. Black, "Electromigration-A Brief Survey and Some Recent Results," *IEEE Trans. on Electron Devices*, vol. 16, no. 4, pp. 338–347, 1969.

[61] I. A. Blech, "Electromigration in Thin Aluminum Films on Titanium Nitride," *Journal of Applied Physics*, vol. 47, no. 4, pp. 1203–1208, 1976.

[62] M. A. Korhonen, P. Borgesen, K. N. Tu, and C. Y. Li, "Stress Evolution Due to Electromigration in Confined Metal Lines," *Journal of Applied Physics*, vol. 73, no. 8, pp. 3790–3799, 1993.

[63] V. Sukharev, "Beyond Black's Equation: Full-Chip EM/SM Assessment in 3D IC Stack," *Microelectronic Engineering*, vol. 120, pp. 99–105, 2014.

[64] Z. Suo, *Reliability of Interconnect Structures*, vol. 8 of *Comprehensive Structural Integrity*. Amsterdam: Elsevier, 2003.

[65] J. He and Z. Suo, "Statistics of electromigration lifetime analyzed using a deterministic transient model," *AIP Conference Proceedings*, vol. 741, no. 1, pp. 15–26, 2004.

[66] S. Chatterjee, M. Fawaz, and N. F. Najm, "Redundancy-Aware Electromigration Checking for Mesh Power Grids," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, 2013.

[67] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, "Interconnect lifetime prediction under dynamic stress for reliability-aware design," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pp. 327–334, IEEE, November 2004.

[68] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer Publishing Company, Incorporated, 2nd ed., 2010.

[69] Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba, "Advanced configuration and power interface specification 5.0a," 2013. `http://www.acpi.info`.

[70] K.Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *International Symposium on Computer Architecture*, pp. 2–13, 2003.

[71] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley-Interscience, 2002.

[72] Intel, "Enhanced intel speedstep technology for the intel pentium m processor," 2004. http://download.intel.com/design/network/papers/30117401.pdf.

[73] L. Ingber, "Adaptive simulated annealing (asa): Lessons learned," *Control and cybernetics*, vol. 25, pp. 33–54, 1996.

[74] J. W. McPherson, "Time Dependent Dielectric Breakdown Physics - Models Revisited," *Microelectronics Reliability*, vol. 52, no. 9, pp. 1753–1760, 2012.

[75] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, (New York, NY, USA), pp. 169–180, ACM, 2009.

[76] IBM, "Ilog cplex optimizer," 2015. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer`.

[77] E. Klotz and A. M. Newman, "Practical guidelines for solving difficult mixed integer linear programs," *Surveys in Operations Research and Management Science*, vol. 18, no. 1, pp. 18–32, 2013.

[78] M. Salehi, M. K. Tavana, S. Rehman, F. Kriebel, M. Shafique, A. Ejlali, and J. Henkel, "Drvs: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations," in *Proceedings of the 2005 international symposium on Low power electronics and design*, (New York, NY, USA), ACM, 2015.

[79] A. Das, A. Kumar, and B. Veeravalli, "Reliability-driven task mapping for lifetime extension of networks-on-chip based multiprocessor systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, (San Jose, CA, USA), pp. 689–694, EDA Consortium, 2013.

[80] "Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C ." `http://www.jedec.org`.

[81] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '04, (Washington, DC, USA), pp. 35–40, IEEE Computer Society, 2004.

[82] L. Tan, S. Song, P. Wu, Z. Chen, R. Ge, and D. Kerbyson, "Investigating the interplay between energy efficiency and resilience in high performance computing," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pp. 786–796, May 2015.

[83] Y. Zhang, K. Chakrabarty, and V. Swaminathan, "Energy-aware fault tolerance in fixed-priority real-time embedded systems," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pp. 209–213, Nov 2003.

[84] J. Srinivasan, S. V. Adve, P. Bose, J. Rivers, and C.-K. Hu, "Ramp: A model for reliability aware microprocessor design," *IBM, Poughkeepsie, NY*, 2003.

[85] X. Li, S. Adve, P. Bose, and J. Rivers, "Architecture-level soft error analysis: Examining the limits of common assumptions," in *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pp. 266–275, June 2007.

[86] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, (Washington, DC, USA), pp. 29–, IEEE Computer Society, 2003.

[87] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *2011 International Reliability Physics Symposium*, pp. 5B.4.1–5B.4.7, April 2011.

[88] Y. Cao, *Predictive technology model for robust nanoelectronic design*. Springer Science & Business Media, 2011.

[89] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[90] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[91] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computation*, vol. 6, pp. 1185–1201, Nov. 1994.

[92] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 52:1–52:12, Nov. 2011.

[93] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug. 2011.

[94] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, pp. 92–99, Nov. 2005.

[95] K. Ghose and et al, "Marssx86: Micro architectural systems simulators," in *ISCA Tutorial Session*, 2012.

[96] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe, "Simflex: Statistical sampling of computer system simulation," *IEEE Micro*, vol. 26, pp. 18–31, July 2006.

[97] J. H. Ahn, S. Li, O. Seongil, and N. Jouppi, "Mcsima+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pp. 74–85, April 2013.

[98] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 72–81, ACM, 2008.

[99] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," in *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, pp. 24–36, June 1995.

[100] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, (San Francisco, CA, USA), pp. 394–402, Morgan Kaufmann Publishers Inc., 1995.

[101] K. He, X. Huang, and S. X.-D. Tan, "EM-Based on-chip aging sensor for detection and prevention of counterfeit and recycled ICs," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2015.

[102] M. Berktold and T. Tian, "Cpu monitoring with dts/peci," 2010. http://www.intel.com/content/www/us/en/embedded/testing-and-validation/cpu-monitoring-dts-peci-paper.html.

[103] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan, "A systematic method for functional unit power estimation in microprocessors," in *Proc. Design Automation Conf. (DAC)*, pp. 554–557, June 2006.

[104] G. Dhiman and T. S. Rosing, "Dynamic power management using machine learning," in *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '06, (New York, NY, USA), pp. 747–754, ACM, 2006.

[105] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1395–1408, Sept 2010.

[106] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, (San Jose, CA, USA), pp. 1521–1526, EDA Consortium, 2015.

[107] M. Sridharan and G. Tesauro, *Multi-agent Q-learning and Regression Trees for Automated Pricing Decisions*, pp. 217–234. Boston, MA: Springer US, 2002.

[108] G. Tesauro and J. O. Kephart, "Pricing in agent economies using multi-agent q-learning," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 3, pp. 289–304, 2002.

[109] T. Kolpe, A. Zhai, and S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Proc. Design, Automation and Test In Europe. (DATE)*, pp. 1–6, March 2011.

[110] V. Sukharev, X. Huang, and S. X.-D. Tan, "Electromigration Induced Stress Evolution Under Alternate Current and Pulse Current Loads," *Journal of Applied Physics*, vol. 118, pp. 034504–1–034504–10, 2015.

[111] X. Huang, V. Sukharev, T. Kim, and S. X.-D. Tan, "Electromigration recovery modeling and analysis under time-depdendent current and temperature stressing," in *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pp. 244–249, 2016.

[112] K.-D. Lee, "Electromigration Recovery and Short Lead Effect under Bipolar- and Unipolar-Pulse Current," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 6B.3.1–6B.3.4, April 2012.

[113] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, pp. 253–266, Feb 2010.

[114] C. Silvano, G. Palermo, S. Xydis, and I. Stamelakos, "Voltage island management in near threshold manycore architectures to mitigate dark silicon," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[115] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, May 2006.

[116] S. Li, J. H. Ahn, R. D. Strong, B. J. B, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 469–480, ACM, 2009.

[117] G. A. Rummery and M. Niranjan, *On-Line Q-learning using connectionist systems.* University of Cambridge, Department of Engineering, 1994.

[118] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (ntv) design: Opportunities and challenges," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 1153–1158, ACM, 2012.

[119] S. Wang and J.-J. Chen, “Thermal-aware lifetime reliability in multicore systems,” in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pp. 399–405, March 2010.

[120] W. Song, S. Mukhopadhyay, and S. Yalamanchili, “Architectural reliability: Lifetime reliability characterization and management of many-core processors,” *Computer Architecture Letters*, vol. PP, no. 99, pp. 1–1, 2014.

[121] D. Meisner, J. Wu, and T. F. Wenisch, “Bighouse: A simulation infrastructure for data center systems,” in *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, 2012.

[122] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, “Power management of online data-intensive services,” in *International Symposium on Computer Architecture*, 2011.

[123] D. Wong and M. Annavaram, “Implications of high energy proportional servers on cluster-wide energy proportionality,” in *Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture*, HPCA-19 ’14, 2014.

[124] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, “pfabric: Minimal near-optimal datacenter transport,” in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM ’13, (New York, NY, USA), pp. 435–446, ACM, 2013.

[125] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[126] M. T. Heath, *Scientific Computing: An Introductory Survey*. McGraw-Hill, 1997.

[127] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 593–605, IEEE, 1989.

[128] www.spec.org/power_ssj2008/, “Specpower_ssj2008,” 2012.

[129] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, “Analyzing cuda workloads using a detailed gpu simulator,” in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 163–174, April 2009.

[130] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, “Gpuwattch: Enabling energy optimizations in gpgpus,” in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA ’13, (New York, NY, USA), pp. 487–498, ACM, 2013.

[131] “HotSpot Program.” `http://lava.cs.virginia.edu/HotSpot/versions.htm`.

[132] NVIDIA, “GTX480 Architecture,” 2010. `https://www.bjorn3d.com/2010/03/nvidia-gtx-480-fermi-gf100/`.

[133] NVIDIA, “Multi-process service,” tech. rep., May 2015.

[134] R. Thomas, N. Sedaghati, and R. Teodorescu, "Emergpu: Understanding and mitigating resonance-induced voltage noise in gpu architectures," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 79–89, April 2016.

[135] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, (Washington, DC, USA), pp. 44–54, IEEE Computer Society, 2009.