This is to certify that the
dissertation entitled

ACTIVITY-AWARE MODELING AND DESIGN
OPTIMIZATION OF ON-CHIP SIGNAL INTERCONNECTS

presented by

KRISHNAN SUNDARESAN

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Electrical Engineering

Major Professor's Signature

12/7/2006

Date

ACTIVITY-AWARE MODELING AND DESIGN OPTIMIZATION OF ON-CHIP SIGNAL INTERCONNECTS

By

Krishnan Sundaresan

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

2006

ABSTRACT

ACTIVITY-AWARE MODELING AND DESIGN OPTIMIZATION OF ON-CHIP SIGNAL INTERCONNECTS

By

Krishnan Sundaresan

On-chip global signal bus energy dissipation, thermal reliability, and latency are all dependent upon transmitted word values. Real-world microprocessor workloads cause bus traffic that exhibit significant spatial, temporal, and value locality. However, existing signal interconnect modeling and optimization schemes are oblivious of the correlated nature of such traffic and were developed with random or worse-case (highly-changing) traffic conditions in mind, which limits their effectiveness. To address this, we present activity-aware methods to model and optimize bus energy dissipation, thermal reliability, and latency.

In the area of modeling, we present an activity-aware bus energy and thermal model that permits monitoring of energy dissipation and temperature, both spatially (horizontally across wires and longitudinally along individual wires) and temporally, during microarchitectural simulation of real programs. We find that final temperatures of wires in global signal buses carrying data (instruction) in the processor core increase by as much as 37 (58) degrees Celsius during a simulation run of only a billion instructions in 130-nm (45-nm) fabrication technology. We also find that highly-active wires in these buses attain absolute temperatures of up to 104 (123.7) degrees in 130-nm (45-nm) processors that are higher than the 100 degrees temperature typically assumed during interconnect design. In addition, wire temperature gradients across the sending and receiving ends, with magnitudes between 16-25 degrees, were also

detected. These conditions were found to degrade processor performance by at least 4% (11.92%) in 130-nm (45-nm) processors.

In bus design, we present a traffic-profile-guided approach to optimize bus energy subject to designer-specified thermal constraints and to reduce worst-case bus crosstalk and latency conditions. Our methodology performs these by evaluating several options for signaling individual bit values and all possible ways of mapping bits to bus lines (bit ordering), and then choosing, based on traffic value characteristics, an optimal encoding scheme (the combination of bit signaling and ordering) statically at design time to support in hardware. Our energy-optimal static encoding techniques provide bus energy reductions of 30.2% (52.1%) for processor core data (instruction) buses, respectively, compared to existing more-complex dynamic encoding schemes that yield only 4.19% (5.32%) reductions for the same buses. Our static encoding technique with thermal constraints added during optimization reduces peak wire temperatures by up to 12.26 (12.96) degrees for data (instruction) buses, while still providing significant energy savings. Finally, we also present a static encoding technique that reduces worst-case bus crosstalk conditions by at least 29.35% and a variable-cycle bus architecture that takes advantage of this reduced crosstalk to improve bus performance by 17.42%.

Our work represents a significant advancement over existing approaches that are activity-oblivious and/or consider worst-case traffic conditions. The microarchitecture-level activity-driven spatiotemporal bus energy and thermal model we present is the first of its kind. Our static value-aware bit reordering and signaling techniques are also highly-novel solutions that work remarkably well in real applications.

Dedicated to Mom, Dad, and Deepa,

for their unending love, support, and encouragement

ACKNOWLEDGEMENTS

The completion of this research and writing of this dissertation has been one of the most significant academic challenges that I have ever had to face. Without the support, guidance, and patience of many people this endeavor would not have been possible. I owe my thanks to all of them.

I have been fortunate to learn from many excellent teachers, from grade to graduate school, and I am indebted to all of them for helping me reach where I am today. In particular, I thank my advisor, Dr. Nihar Mahapatra, for his technical guidance and support, over the last five years. His mentorship has instilled in me the skill and confidence to identify, analyze, and efficiently solve research problems and present results in a clear and lucid manner. I have also learnt much from his classes and from our research meetings and discussions. I also thank my dissertation committee members, Dr. Anthony Wojcik, Dr. Andrew Mason, and Dr. Peixin Zhong, for their very insightful review and comments which have helped me improve this work.

I have also been fortunate to be in the company of a lot of good friends, many of them my lab-mates, and I thank them all for their support. Jiangjiang Liu helped me get my feet wet in research and was a great colleague during the early years. Kaushal Gandhi and Srivathsan Krishnamohan have been great friends and lab-mates and I have benefited greatly from many technical discussions I have had with them. I cherish their friendship, the good times we had together, and look forward to more Friday-night pizza-and-beer get-togethers in the Bay area where all three of us are starting our professional careers.

My family—Mom, Dad, and sister—has been a great source of encouragement through the years and their continuing love and affection has made me what I am today. I owe much more to them than what a few sentences can express. This dissertation is dedicated to them.

Last but not least, I thank all members of the Greater Lansing Bhagavad Gita group for their good thoughts and prayers. My association with them has helped me keep up my sanity during these years and taught me to live by the Bhagavad Gita's motto: *yogah karmasu kausalam*—"Efficiency in Action leads to (the Ultimate) Knowledge."

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xii

SELECTED LIST OF SYMBOLS xix

1 INTRODUCTION AND OVERVIEW 1

1.1 Interconnect Scaling Trends: Delay, Power, Temperature, and Reliability 1

1.2 Material, Process, and Architectural Advances 4

1.3 Impact of Interconnects on Architecture and VLSI 6

1.3.1 Wire Delay 7

1.3.2 Power and Temperature 8

1.3.3 Computer-Aided Design Tool Requirements 10

1.4 Drawbacks in Existing Techniques 11

1.5 The Need for Activity-Aware Design 12

1.6 Our Contributions 15

1.6.1 Activity-Aware Design Methodology 15

1.6.2 Accurate Energy, Temperature, and Delay Modeling 16

1.6.3 Profile-Guided Optimization Techniques 17

1.6.4 Novel Thermal Optimization Methodology 18

1.6.5 Performance-Oriented Adaptive Bus Design 19

1.7 Dissertation Outline 19

2 PRELIMINARIES 21

2.1 Interconnect Analysis Methods 21

2.1.1 Global, Semiglobal, and Local Wires 22

2.1.2 Interconnect Models: RC and RLC 22

2.1.3 Effect of Inductance on Global Signal Lines 23

2.1.4 Energy Estimation 24

2.1.5 Delay and Performance 26

2.2 Interconnect Optimization Techniques 28

2.2.1 Data Encoding 28

2.2.2 Wire Spacing and Shielding 31

2.3 Architecture-Level Simulators and Early-Stage Design 31

2.4 Our Experimental Methodology 33

2.4.1 Interconnect Geometry and Technology Data 33

2.4.2 Parasitic Capacitance Extraction 34

2.4.3 Simulation Infrastructure and Verification of its Correctness . 36
2.4.4 Target Systems and Benchmarks 41

3 ACTIVITY-DRIVEN ENERGY AND TEMPERATURE MODEL 43
3.1 Introduction 43
3.2 Related Work and Our Contributions 47
3.3 Bus Line Energy Dissipation Model 50
3.3.1 Energy Dissipated due to Line Self Capacitance 51
3.3.2 Energy Dissipated due to Inter-Wire Capacitance 52
3.3.3 Distributed-RC Line Energy Model 53
3.4 Thermal Model 57
3.4.1 Chip Thermal Structures and Heat Transfer 58
3.4.2 Detailed Thermal Model 59
3.4.3 Steady-State Thermal Model 64
3.5 Simulation Environment and Methodology 66
3.5.1 Benchmarks and Sample Sizes 66
3.5.2 Thermal Warmup and Initial Temperatures 67
3.5.3 Granularity of Thermal Simulation 68
3.6 Experiments and Results 69
3.6.1 Energy Dissipation in Processor Buses 69
3.6.2 Correlation between Energy and Temperature 75
3.6.3 Final and Peak Wire Temperatures 76
3.6.4 Wire Temperature Gradients 86
3.7 Summary 89

4 DATA- AND TEMPERATURE-DEPENDENT DELAY VARIABILITY MODEL 91
4.1 Introduction 91
4.2 Related Work and Our Contributions 92
4.3 Temperature Dependent Delay Variability Model 94
4.3.1 Wire Delay Considering Temperature Impact 95
4.3.2 Wire Delay Variability Considering Crosstalk and Temperature 96
4.4 Results and Discussion 97
4.4.1 Maximum Wire Temperatures and Gradients 98
4.4.2 Frequency of Timing Violations 100
4.4.3 Performance Impact 105
4.5 Summary 106

5 ACTIVITY-AWARE ENERGY AND TEMPERATURE OPTIMIZATION 109
5.1 Introduction 109

5.1.1 Need for Energy and Temperature Aware Bus Design 110
5.1.2 Key Contributions and Results 112
5.2 Related Work 114
5.3 Methodology 116
5.3.1 Target Scenarios 117
5.3.2 Bus Layout and Wire Geometry 120
5.4 Static Techniques for Bus Energy and Temperature Optimization . . 121
5.4.1 Choice of Signaling Modes 121
5.4.2 Minimum Energy Signaling (MES) 126
5.4.3 Minimum Energy Bit Ordering (MEBO) 127
5.4.4 Simultaneous Bit Ordering and Signaling (SBOS) 129
5.4.5 Thermal Optimization Methodology 130
5.4.6 Routing Overheads 134
5.5 Results and Discussion 136
5.5.1 Energy Dissipation in Processor Buses 138
5.5.2 Energy Reduction for General-Purpose Design 145
5.5.3 Energy Reduction for Workload-Specific Design 145
5.5.4 Energy Reduction for Program-Specific Design 147
5.5.5 Wire Temperature Reduction 154
5.6 Summary 161

6 ACTIVITY-AWARE PERFORMANCE OPTIMIZATION 163
6.1 Introduction 163
6.2 Related Work 164
6.3 Techniques for Performance Optimization 165
6.3.1 Variable Cycle Bus (VCB) Design 165
6.3.2 Minimum Crosstalk Bit Ordering (MCBO) 168
6.3.3 MCBO with Signaling (MCBOS) 171
6.4 Results and Discussion 172
6.4.1 Peak Crosstalk Reduction 172
6.4.2 Performance Improvement with VCB 173
6.5 Summary 177

7 CONCLUSION 178
7.1 Contributions and Key Results 178
7.2 Directions for Future Research 182

BIBLIOGRAPHY 183

LIST OF TABLES

2.1 Bus crosstalk conditions and models for a rising transition in the middle (victim) wire. 27

2.2 Technology, wire geometry, and equivalent circuit parameters for topmost layer interconnect. Values in top eight rows are from the international technology roadmap for semiconductors (ITRS) document [1]. Values listed in the next three rows are from Mui et al. [2]. The values for the self and coupling capacitances were extracted using the FastCap tool and the value for r_i was calculated using the formula $r_i = \rho_{Cu}/(w_i \cdot t_i)$, where $\rho_{Cu} = 2.2 \times 10^{-8}\Omega$-m. Values of h and k were found using expressions given in Section 2.1.5 and $r = c_{i,i \pm 1}/(c_{line} + h \times C_0)$. 35

2.3 Configuration of our target system and benchmarks. This processor-memory system configuration is based on the Alpha 21264 processor. 42

3.1 Comparison of normalized energy dissipated in wire subsegments obtained using our model and Cadence Spectre simulations for 10 subsegments. 57

3.2 Maximum wire temperatures in °C recorded during a simulation of one billion committed instructions for data and instruction buses using 130 nm and 45 nm parameters. 87

4.1 Maximum wire temperatures recorded for the ALU result bus. Ambient temperature is 318.15 K. 99

4.2 Performance impact expressed as percentage IPC degradation. 107

5.1 Optimization scenarios considered in this work. 118

5.2 Correlation coefficients r_{xy} between test and training set data for various signaling schemes discussed in Section 5.4.1. Since r_{xy} values are close to 1, our training and test sets are well correlated. 120

5.3 Number of iterations and running times for various problem types and sizes. 137

5.4 Optimal signaling and ordering obtained for workload-specific design of the data bus (0=LSB, 63=MSB). ♠ = org, ♡ = inv, ♢ =trs, ♣ =itr, and △ =mm. 148

5.5 Thermal Optimization Results. Peak wire temperatures (K) in data and instruction buses for SBOS scheme with and without thermal constraints (TC) applied during optimization. The methodology described in Section 5.4.5 was used to obtain the trade-off curves in Figures 5.16–5.20 and the wire permutations that resulted in bus energy reduction closest to $0.5(1 - \frac{E_{opt}}{E_{orig}})$ were chosen from each benchmark's tradeoff curve. Results shown here are for detailed thermal simulations with this permutation. 153

LIST OF FIGURES

1.1 Gate and interconnect delay scaling for current and future nanometer-scale technologies. Local interconnects scale with gate delay whereas global interconnect delays do not [3]. 2

1.2 Interconnect power dissipation due to global and local wires. Global lines are responsible for 21% of total dynamic power dissipation at 130 nm [4]. 3

1.3 Projected wire temperature rise in multi-layer interconnects for various technologies under worst-case conditions. Global metal lines will be the hottest, with temperatures expected to reach as much as 209°C in 45 nm technology [5]. 4

1.4 Pipeline stages and loops in a typical out-of-order processor. More frequently used loops like fetch, LSQ, and bypass are affected strongly by wire delay. 7

1.5 Power dissipation in Intel processors showing an exponential trend [6]. Since 2001, low-power and power management techniques that have been used widely in microprocessors and have helped slow down the trend somewhat. 9

2.1 Layout of wires routed in the top-most layer metal. Self and coupling capacitances are shown. The bottom plate represents the V_{DD}/GND plane. 33

2.2 Wire segment of length l_{opt} between two repeaters. 34

2.3 Distribution of self and coupling capacitance values for the middle wire of a 32-bit bus extracted using the FastCap tool [7]. Cgnd = self capacitance of the wire; Cc1 = coupling capacitance between the wire and its adjacent neighbor; Cc2 = coupling capacitance between the wire and a non-adjacent wire with 1 wire between them; Cc3 = coupling capacitance between the wire and a non-adjacent wire with 2 wires between them; Cc_rest = sum of coupling capacitances between the wire and other wires with 3 or more wires between them. For current and near-future ITRS technology nodes (up to 45 nm), non-adjacent coupling capacitances are somewhat non-negligible—they contribute approximately 8–10%. 37

3.1 Distributed-RC model of the wire segment divided into n subsegments. 53

3.2 Figure shows the view of different thermal structures of a C4/CBGA chip and the primary and secondary heat transfer paths. 58

3.3 Thermal model. (a) Complete equivalent thermal-RC network for a 5-wire bus. $P'_{1,k} = P'_{2,k} = \ldots = P'_{5,k}$, $R_{1,k} = R_{2,k} = \ldots = R_{5,k}$, $C_{1,k} = C_{2,k} = \ldots = C_{5,k}$, and $P_{1,k}, P_{2,k}, \ldots, P_{5,k}$ are bus-activity dependent in the model shown. (b) Geometry for calculating equivalent thermal resistances for a wire based on previous work of Chiang et al. The lightly shaded regions and arrows represent heat flow between the conductors or between layers (from a hotter to a cooler one). 60

3.4 Steady state thermal equivalent circuit for three wires. Heat transfer between wires is modeled by R_{inter} and heat loss to surroundings by R_{th}. P_i represents power dissipated in each wire due to switching activity and it can found using a microarchitecture-level simulator. . 65

3.5 Total energy dissipated in a 64-bit data bus for various benchmarks. 'Cc1 only' represents the existing energy models which consider only self and adjacent coupling capacitances. 'Cc1+Cc2+Cc3' represents our model that considers self capacitances, adjacent coupling capacitances (Cc1), and two non-adjacent capacitances (Cc2 and Cc3) on each side. The % energy mismatch shown by the line is plotted with respect to the right-hand side Y-axis. 71

3.6 Total energy dissipated in a 128-bit instruction bus for various benchmarks. The % energy mismatch shown by the line is plotted with respect to the right-hand side Y-axis. 72

3.7 Total energy dissipated in a 64-bit data bus with various encoding schemes. 'Self' denotes self energy, 'C/D' denotes the coupling charge/discharge energy and 'Toggle' denotes the coupling toggle energy dissipation. 'Cc1 only' refers to existing energy models that consider self and adjacent coupling capacitance only and 'Cc1+Cc2+Cc3' refers to our energy model that considers self, adjacent coupling, and two non-adjacent coupling capacitances. 73

3.8 This plot shows average energy dissipation and wire temperature of the bus for a simulation interval of 10 billion cycles. The continuing temperature rise can be clearly observed. 77

3.9 Plots show the wire temperature rise recorded for benchmarks gcc and gzip for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 79

3.10 Plots show the wire temperature rise recorded for benchmarks mcf and lucas for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 80

3.11 Plots show the wire temperature rise recorded for benchmarks ammp and applu for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 81

3.12 Plots show the wire temperature rise recorded for integer benchmarks gcc and gzip for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 82

3.13 Plots show the wire temperature rise recorded for integer benchmarks mcf and lucas for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 83

3.14 Plots show the wire temperature rise recorded for integer benchmarks ammp and applu for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark. 84

3.15 A three-dimensional plot showing spatial and temporal variations in wire temperature for the lower-order 32 bits of the load/store data bus for the *gcc* benchmark. 85

3.16 Frequency distribution of maximum wire temperature gradients for 130 nm and 45 nm processor wires. 88

4.1 Distribution of maximum wire temperature gradients in result bus wires for the 130 nm processor. 100

4.2 The number of temperature-induced violations per hundred bus references occurring across ten benchmark programs in a 130 nm processor. 101

4.3 The number of temperature-induced violations per hundred bus references occurring across ten benchmark programs in a 45 nm processor. 102

4.4 This plot shows the frequency of occurrence of five different crosstalk conditions on the bus. See Section 4.3.2 and Table 2.1 for an explanation of these crosstalk conditions. The crosstalk condition determines the actual propagation delay without considering thermal effects. . . 103

4.5 Figure shows the percentage of temperature-induced delay violations that correspond to a given crosstalk condition. 104

5.1 Markov model-based signaling technique. (a) A 4-bit prediction table for the Markov model for bits 0–7 of the data bus obtained by analyzing training set benchmarks. Depending on which bits are selected for Markov model signaling, the corresponding row of the table can be translated to hardware using logic minimization tools. (b) Examples of sending end hardware that would be required for 2 bits (0 and 7) assuming these are chosen to be signaled using the mm scheme. As can be seen, the logic overhead required for mm signaling is very minimal. . 124

5.2 Sample peak wire temperature versus bus energy trade-off curve. The thermal optimization steps can be used to obtain curves similar to the one shown here. 133

5.3 Routing strategy and overheads for re-ordering. (a) Definition of the routing channel. (b) Matching diagram showing ten crossing points. (c) Two-layer routing strategy using eight horizontal tracks and ten vias. 135

5.4 Transition Densities for the 13 integer SPEC CPU2000 Benchmarks for 64-bit Data Bus. 139

5.5 Transition Densities for the 13 floating-point SPEC CPU2000 Benchmarks for 64-bit Data Bus. 140

5.6 Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit data address bus in the Alpha 21264 target system while running SPEC CPU2000 programs. 141

5.7 Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit instruction address bus in the Alpha 21264 target system while running SPEC CPU2000 programs. 142

5.8 Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 64-bit data bus in the Alpha 21264 target system while running SPEC CPU2000 programs. 143

5.9 Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 128-bit instruction bus in the Alpha 21264 target system while running SPEC CPU2000 programs. 144

5.10 Energy dissipation results for general-purpose design for the 64-bit data bus. Statistics collected on 13 training set benchmarks were used to obtain the optimal static encoding schemes. These were tested on 13 other (test set) benchmarks. Average energy reductions are MES: 7.81%, MEBO: 11.91%, and SBOS: 20.04%. 146

5.11 Energy dissipation results for general-purpose design for the instruction bus. Average energy reductions are MES: 10.96%, MEBO: 19.85%, and SBOS: 38.78%. 147

5.12 Energy dissipation results for workload-specific design of the 64-bit data bus. Statistics collected for SimPoint samples from 13 training set benchmarks were aggregated and used to obtain the optimal static encoding schemes. These were then tested on a non-overlapping sample from the same set of benchmarks. The average energy reductions are MES: 9.73%, MEBO: 15.97%, and SBOS: 22.79%. 149

5.13 Energy dissipation results for workload-specific design for the 128-bit instruction bus. The average energy reductions are MES: 10.43%, MEBO: 21.25%, and SBOS: 40.77%. 150

5.14 Energy reduction results for program-specific design. Statistics collected for SimPoint samples of each benchmarks was used to obtain the optimal static encoding schemes specific to that benchmark for our schemes, MES, MEBO, and SBOS. These were then tested on the same sample. Results for dynamic encoding schemes BI and OEBI proposed in previous work are also shown. The average energy reductions for the data bus are BI: 4.19%, OEBI: 1.58%, MES: 19.7%, MEBO: 23.25%, and SBOS: 30.2%. 151

5.15 Energy reduction results for program-specific design. Statistics collected for SimPoint samples of each benchmarks was used to obtain the optimal static encoding schemes specific to that benchmark for our schemes, MES, MEBO, and SBOS. These were then tested on the same sample. Results for dynamic encoding schemes BI and OEBI proposed in previous work are also shown. The average results for the instruction bus are BI: 2.63%, OEBI: 5.32%, MES: 21.7%, MEBO: 32.1%, and SBOS: 52.1%. 152

5.16 Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for ammp and crafty. The permutation selected for each benchmark was the one that resulted in bus energy reduction closest to $0.5(1 - \frac{E_{opt}}{E_{orig}})$ compared to the original bus. 155

5.17 Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for eon and gcc. 156

5.18 Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for gzip and lucas. 157

5.19 Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for mesa and mgrid. 158

5.20 Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for swim and twolf. 159

6.1 Three-bit crosstalk analyzer truth table and circuit. (a) Truth table showing only the ON-set. "-" indicates a don't care input. (b) Logic circuit implementing the truth table. 166

6.2 Variable cycle bus. (a) Complete bus crosstalk analyzer for an n-bit bus. (b) Sender and receiver logic for VCB. 168

6.3 Crosstalk reduction results for workload-specific design of the 64-bit ALU result bus. (a) Average reductions in number of 1+4r delay cycles. For MCBO: 24.89% and MCBOS: 30.61%. (b) Average reductions in number of 1+3r cycles. For MCBO: 19.21% and MCBOS: 23.42%. . 174

6.4 Crosstalk reduction results for general purpose design of the 64-bit ALU result bus. (a) Average reductions in number of 1+4r delay cycles. For MCBO: 21.22% and MCBOS: 29.35%. (b) Average reductions in number of 1+3r cycles. For MCBO: 16.77% and MCBOS: 20.29%. . 175

6.5 Reduction in the number of cycles taken to transmit the information with MCBO and MCBOS applied to the result bus. (a) Workload-specific optimization. (b) General-purpose optimization. 176

SELECTED LIST OF SYMBOLS

$\mathcal{C}_{i,k}$	Thermal capacitance of the k^{th} subsegment of the i^{th} wire
$\mathcal{R}_{i,k}$	Thermal resistance along the heat transfer path of the k^{th} subsegment of the i^{th} wire
θ_0	Ambient temperature inside the computer box (45°C)
ε_r	Relative permittivity of dielectric
C_0	Capacitance of minimum size inverter in fF
$c_{i,i\pm 1}$	Adjacent coupling capacitance per unit length in pF/m
$c_{i,j}$	Coupling capacitance between line i and any other line j, $i \neq j$
c_{line}	Self/Area capacitance of wire per unit length in pF/m
f_{clk}	Clock frequency
k_{ild}	Thermal conductivity of dielectric
R_0	Resistance of minimum size inverter in kΩ
r_{line}	Resistance of wire per unit length in kΩ/m
t_{ild}	Thickness of the inter-layer dielectric
t_i	Thickness of wire i
t_p	End-to-end propagation delay of a wire
V_{DD}	Supply voltage
w_i	Width of wire i

CHAPTER 1
INTRODUCTION AND OVERVIEW

High-speed systems and circuits are increasingly facing the limitations posed by shrinking physical dimensions of transistors and their interconnections [8]. As circuits become denser, smaller transistors naturally speed up. But interconnects, in general, do the reverse and introduce delays that reduce or even cancel the speed gains due to smaller transistors. The problems due to interconnects are exacerbated by the fact that parasitic resistance, inductance, and capacitance (RLC) effects increase as wires scale to smaller dimensions, which in turn aggravates delay, power consumption, and cause signal integrity/reliability problems. Thus, on-chip interconnect design has been recognized as one of the most important challenge to address in nanometer-scale integrated circuits [9, 10].

1.1 Interconnect Scaling Trends: Delay, Power, Temperature, and Reliability

According to the data available from the international technology roadmap for semiconductors (ITRS) documents, the intrinsic gate delay has improved ten times, from 10 ps to 1 ps in the 20 years between 1980 and 2000. However, in the same period of time, the interconnect delay in a 1 mm line degraded 100 times, from 1 ps to 100 ps [1]. This growing disparity between gate and interconnect delays is also highlighted in Figure 1.1 for current and future technologies [3]. The figure shows that while local interconnect delays scale with gate delays, global interconnect delays do not. Such trends have forced costly performance compromises, like the allocation of two out of twenty pipeline stages for communication in the Pentium-4 microprocessor [11].

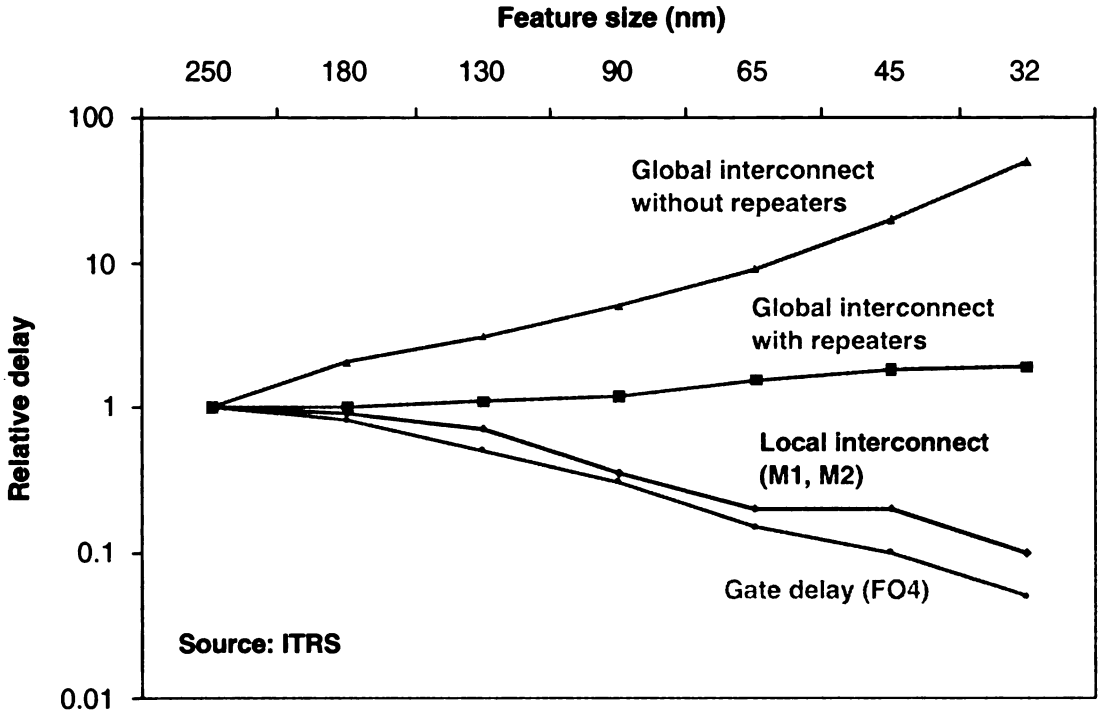


Figure 1.1. Gate and interconnect delay scaling for current and future nanometer-scale technologies. Local interconnects scale with gate delay whereas global interconnect delays do not [3].

Interconnects are also responsible for about 50% of the power dissipation, as shown by results from studies on a 130 nm Intel microprocessor [4]. Figure 1.2 shows the distribution of power dissipation by the type of the net/wire. As can be seen, global signal lines account for 34% of the total interconnect power dissipated and hence 21% of the total dynamic (switching-related) power dissipation at 130 nm.

Due to increased Joule heating in the global wires, their temperatures are also increasing alarmingly. The spatial temperature distributions along the vertical direction

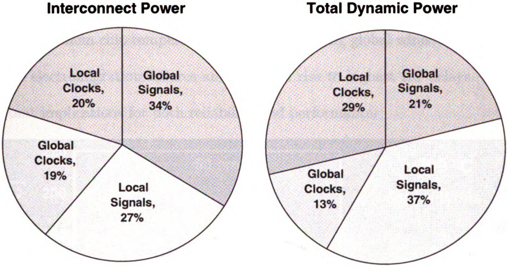


Figure 1.2. Interconnect power dissipation due to global and local wires. Global lines are responsible for 21% of total dynamic power dissipation at 130 nm [4].

from the Silicon (Si) substrate obtained using finite element models and simulations are shown in Figure 1.3 [5]. This analysis assumed that all wires in the interconnect stack carried currents with maximum rated current density for that technology which represents an extreme worst case. Nevertheless, the results show how temperatures will be distributed across interconnect layers. It can be observed that as technology scales down, the temperature gradient between the top metal lines and the substrate becomes larger. Global metal lines were found to be the hottest in all technologies using this worst-case analysis, with temperatures reaching as much as 209°C in 45 nm technology [5]. For the 35 nm node, the temperature gradient is smaller than that for the 50 nm node due to the larger fraction of metalization (Cu) layers compared to inter-layer dielectric (ILD) layers, an artifact of the ITRS scaling scenario that was used for this analysis. It should be noted that the total height of the (Cu+ILD) layers decreases as scaling continues, due to the smaller vertical dimensions of wires and insulators despite increase in number of metal layers. It can also be observed that the maximum chip temperature occurs for the long global wires, which are most prone to electromigration failures and also give rise to highest RC delays. This has important implications for both reliability and performance.

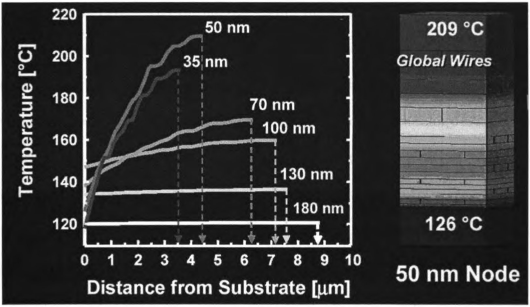


Figure 1.3. Projected wire temperature rise in multi-layer interconnects for various technologies under worst-case conditions. Global metal lines will be the hottest, with temperatures expected to reach as much as 209°C in 45 nm technology [5].

1.2 Material, Process, and Architectural Advances

Many methods, such as utilizing Cu and low-k insulators [12–14], short-wire architectures [15, 16], on-chip networks [17], optical interconnects [18], and three-dimensional interconnect structures [19, 20] have been suggested to help alleviate the impact of interconnect scaling on current and future nanometer-scale fabrication. The pros and cons of these techniques are discussed next.

Material and process enhancements: Copper interconnects in high speed microprocessors were introduced by IBM in its 400 MHz PowerPC750 processor. Although the resistivity of Copper is 40% less than that of Aluminum, the percentage of performance improvement from using the former is limited to about 15% [14]. The thickness and resistivity of the Tantalum (Ta) liner, used in the dual-damascene process for Copper electrodeposition, also limit the performance advantage of Copper interconnects. Low-k dielectrics also help improve chip performance. For example, the performance of a metal wire improves 25% for 0.25 μm technology using a k=2.5 dielectric material compared to conventional silicon dioxide, which has k=3.9. However, the use of Copper metal and low-k dielectrics are known to aggravate thermal issues in interconnects and cause reliability problems, during both fabrication and chip lifetime [21].

Novel architectures: *Short-wire architectures* such as systolic arrays can be employed to overcome some of the problems imposed by long global interconnects [16]. Although these architectures are not applicable to all microprocessors, they can be useful in specific applications, such as pattern recognition, multiprocessor systems, and arithmetic computation. *On-chip networks* can be used instead of global interconnects to reduce the global interconnect congestion [22]. Since most of the global wires are not utilized in every clock cycle, it is more efficient to send packets over a global network rather than signals in global wires. However, this requires a completely new architecture, tools, and design methodology different from conventional microprocessors.

Optical interconnections and 3D integration: It has been shown that the optical interconnections have higher bandwidth and consume lesser power for long-distance communication compared to electrical interconnections [23]. However, because of incompatibility with standard CMOS technology, optical interconnects have not been widely deployed in current microprocessors. The primary application has been restricted to clock distribution networks in some designs [24]. Three-dimensional interconnection schemes are also expected to significantly reduce global wiring requirements and have a significant impact on reducing interconnect delay and power [25]. However, vertical pitch limitations resulting from alignment tolerances in the bonding of wafers [26] and heat removal capacity limitations [27] are some of the problems limiting the use of three-dimensional architectures.

1.3 Impact of Interconnects on Architecture and VLSI

Interconnect-related problems have affected chip design to such an extent that product roadmaps of almost all chip design companies have been drastically re-drawn as it is becoming evident that high-speed processors—with clock frequencies exceeding 10 GHz—are no longer economically viable, due to restrictions imposed by power, temperature, and reliability [28–30]. The impact of interconnect scaling and power and performance issues affects the very first architectural design decisions of today's processors [31,32].

1.3.1 Wire Delay

Processor clock speeds have increased continuously, due to faster transistors and also due to deeper pipelines. However, since global wire delays—for example, delay of register bypass wires—scale much slower than transistor delays, deeper superscalar pipelines have experienced increased latencies and a significant degradation in instruction throughput. Several studies have pointed out rising wire delays dictate that deeper pipelines will not perform better than shallower ones in future technologies and also conclude that superscalars do not have sufficient parallelism to tolerate the relative rise in wire delays [33, 34]. Hence the industry trend toward multi-core and multithreaded architectures [35].

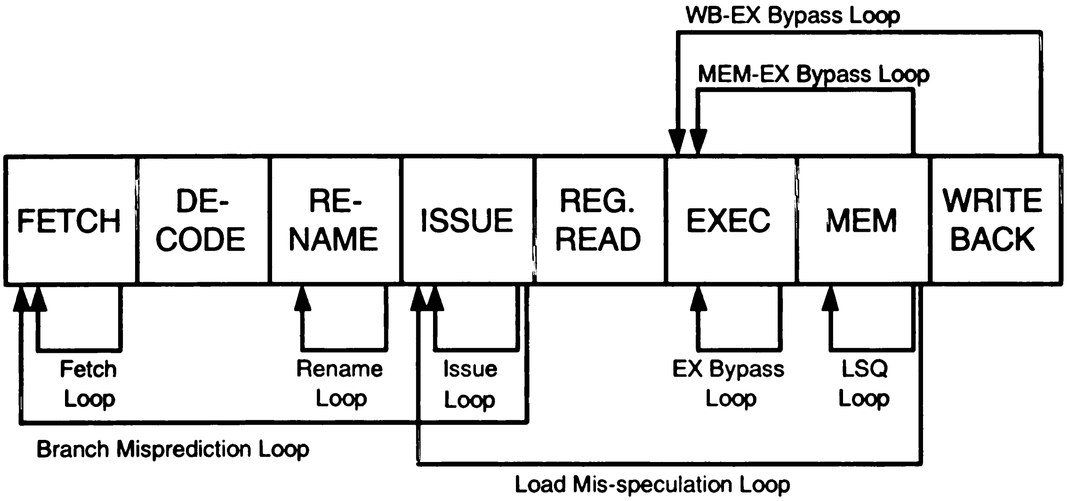


Figure 1.4. Pipeline stages and loops in a typical out-of-order processor. More frequently used loops like fetch, LSQ, and bypass are affected strongly by wire delay.

We briefly examine next why wire delay trends affect architectural decisions. An out-of-order superscalar pipeline is composed of two in-order half-pipelines, called the *front-end* and *back-end*, connected by the issue queue. Figure 1.4 shows this configuration and the various loops in the pipeline [35]. Wire delay affects many of these loops significantly as discussed next. The *fetch loop* is due to the fact that the current program counter (PC) is used to predict the next PC. The delay of this loop includes the instruction bus and cache delays. The *rename loop* is due to the dependence between a previous instruction assigning a rename tag and a later instruction reading the tag and the *issue loop* is due to the dependence between the producer and wakeup of a consumer instruction. The rename and issue loop delays are sensitive to the delay on the tag lines. The *load misspeculation loop* is due to use of speculation and the need for load-miss replay. The *load/store queue (LSQ) loop* is due the dependence between a previous store and a later load to the same address and includes the load/store bus and data cache delays. The various *bypass loops*—EX/EX, EX-MEM, and Writeback-EX—are all affected by the wire delays on the ALU result bus. Also, the more frequently a loop is used, the higher its impact on performance. The fetch, rename, issue, and bypass loops are all fairly frequent and hence have the highest impact. The load misspeculation and branch misprediction loops that are used only upon load misses and branch mispredictions, respectively, are relatively less frequent and have lesser impact.

1.3.2 Power and Temperature

Power has become a first-class constraint in the design of nanometer-scale ICs. Figure 1.5 shows the trend observed in the power dissipation of Intel microprocessors [6]. In 2001, it was predicted that with the scaling rates at that time, the power density in microprocessors will reach that of the Sun by 2015, following an almost exponential trend [6]. Since then various steps have been taken to reduce power dissipation in logic and memories with techniques at various levels of abstraction. These have resulted in reducing the trend to a linear one, as shown by the dotted line in Figure 1.5.

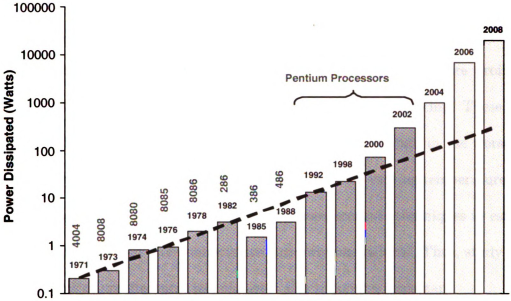


Figure 1.5. Power dissipation in Intel processors showing an exponential trend [6]. Since 2001, low-power and power management techniques that have been used widely in microprocessors and have helped slow down the trend somewhat.

Among the three different sub-systems of a high-performance processor—*computation*, *storage*, and *communication*—the communication or interconnect subsystem, which carries address, instruction, data, and control signals, is still responsible for a bulk of the on-chip power dissipation as discussed earlier, in part due to interconnect scaling trends. With increasing interconnect power dissipation, wire temperatures rise as a result of the Joule effect, wire resistance increases due to temperature-dependent resistivity forcing performance to degrade further, and wire reliability decreases sharply due to electromigration-induced breakage. Even with the advent of multi-core processors, clock frequencies and datapath widths have continued to increase and hence, all of the above effects are bound to worsen further. Hence, interconnect power dissipation and temperature remains one of the primary issues facing microarchitects and VLSI designers.

Popular low-power and power-management techniques like fine-grained clock gating and power gating can also significantly affect on-chip temperature profiles by creating localized hot spots and/or temperature gradients on the chip. These gradients cause delay variabilities, setup and hold time violations and, in the worst case, failure of interconnects that are routed across regions with varying temperatures. Designing for these issues is almost impossible because accurate techniques to estimate temperature gradients in interconnects are currently unavailable. Thus, study of the thermal impact of architectural techniques is also becoming important.

1.3.3 Computer-Aided Design Tool Requirements

In conventional ASIC design, signal and power integrity were checked in later stages of the design cycle and the design was modified if these checks were found to be unsatisfactory. However, with explosion in the number of transistors and highly-complex designs in nanometer-scale technologies, iterating between upstream (architecture or high-level) design changes and layout to achieve design closure has becoming increasingly futile, leading to longer time-to-market schedules and higher design costs [9]. The design-productivity gap, exemplified by the lack of proper CAD tools to identify and correct issues at an early stage, exacerbates this problem. While the push toward ever-higher performance still drives the semiconductor industry, there is growing awareness now that winning designs need to balance multiple objectives: high performance, low power, low cost, robustness (noise immunity), and reliability. As such, it is becoming imperative to: (1) model interconnect-related effects accurately and efficiently for different system architectures (superscalar, multi-core, and network-on-chip) and fabrication technologies (130 nm, 90 nm, etc.) and (2) design the interconnect system, at an early stage, to alleviate or mitigate these effects without incurring unsustainable performance, energy, and/or area/cost overheads.

1.4 Drawbacks in Existing Techniques

Next, we discuss some drawbacks of existing models and design techniques for signal interconnects. First, almost all existing work addressing signal interconnect analysis, design, and optimization is *not activity-aware*, i.e., such interconnect models and design techniques are not developed with an accurate knowledge of the characteristics of data that is transmitted on these interconnects. An average wire switching factor—such as 0.15 suggested in [36]—is used to estimate energy dissipation, wire temperature, delay impact, and/or reliability impact [37, 38]. As such, these average estimates lead to over-design because switching activity in interconnects is actually information and time dependent. It depends on the type of information (address, instruction, data, or control) being transmitted because the information type influences switching activity factors; for example, the activity factor is expected to be higher for data and instruction streams since they are more random in nature than addresses. It varies with time too because, during execution of most typical programs, there are substantial periods when a bus may remain idle; for example, when there are no level-one (L1) cache misses, the bus connecting to level-two (L2) cache will remain idle. These idle cycles help bring down wire temperatures and hence, may reduce wire delay and electromigration impact. Hence, to facilitate interconnect design that can be tuned to the requirements of different architectures, activity-aware modeling and design optimization techniques are necessary.

Second, as mentioned earlier, increasing the number of iterations between high-level design and physical layout to achieve design closure, has become exorbitantly costly, time-consuming, and impractical in nanometer designs. Hence, growing emphasis is being placed on making accurate early-stage design decisions obtained using microarchitecture-level simulations on benchmark programs. Interconnect models that have been built into existing execution-driven simulators lack the detail needed to accurately estimate the impact of interconnect power dissipation, temperature, and related effects, since many not consider the influence of wire coupling and thermal heat dissipation paths. For example, the amount of energy dissipated due to the parasitic coupling capacitance between wires is much greater than energy dissipated due to the area capacitance. Similarly, thermal coupling or heat transfer through the inter-metal dielectric occurs between adjacent wires, affecting temperatures in both wires.

1.5 The Need for Activity-Aware Design

Existing techniques that target bus energy and crosstalk reductions, perform well only when patterns that are transmitted on the bus are randomly distributed in time. However, this is rarely the case in actual microprocessor buses. Information transmitted on these buses show high degrees of correlation across programs as well as across sections of the same program, due to the presence of *temporal*, *spatial*, and *value localities*. Temporal locality describes the likelihood that a recently-referenced item will be referenced again soon, while spatial locality describes the likelihood that a close neighbor of a recently-referenced item will be referenced soon. Value locality refers to the likelihood of a previously-seen value recurring repeatedly in the information stream.

Address, instruction, and data streams in microprocessor buses exhibit substantial amounts of temporal and spatial locality due to the reasons discussed next. Instruction addresses issued by the processor to the L1 cache are typically sequential, except when branches or jumps occur and even then the target addresses are not typically very far away from the last address. This is the reason why many instruction sets use PC-relative addressing with shorter-than-full-word-size offsets for branch and jump instructions. Data addresses issued by the processor are also exhibit these localities primarily because of scanning of data arrays in loops that are placed in contiguous memory locations.

The dynamic instruction stream executed by a processor corresponds to instruction addresses issued by fetch unit, and hence instructions exhibit the same temporal and spatial locality as instruction addresses. Also, not all instructions, instruction sequences, opcodes, register operands, and immediate constants are present equally frequently in the dynamic instruction mix, leading to more predictability in the instruction stream. The reasons for the presence of such redundancies are that all programs share certain basic characteristics: procedures and procedure calls, branches every few instructions—typically every six instructions [39], and loops and if-then-else clauses that lead to repetitive instruction sequences.

Data buses in the processor, such as load/store and ALU result buses, also exhibit temporal and spatial locality, although to a lesser extent than addresses and instruction buses. There is an additional element of redundancy present in the magnitude of values communicated by these buses and stored in registers, data caches, and/or CAM structures in the processor core. This redundancy is due to the fact that for any given type of data—character, integer, floating-point, etc.—not all values are equally likely. For instance, many programs do not tend to use the entire range of integer values possible, but rather the values used tend to be concentrated around certain values, especially, zero. For such small magnitude two's complement numbers, most high order bits of the data bus are likely to be either all zero (positive) or all one (negative) due to sign extension. The concept of value locality also adds to the redundancies present in data buses. For example, the number of times each static load (or store) instruction retrieves a value from (or writes to) memory that matches a previously seen value, is quite high. Studies have shown that this value is around 50% for most superscalar processors running standard benchmark applications [40].

The presence of temporal, spatial, and value localities in information streams opens opportunities for *activity-aware design* of high-performance buses, i.e., design that is tailored to the unique characteristics of different types of data that are transmitted on these buses, as well as to the typical applications that are executed on the processor. Such design can be achieved with the following steps: (1) profile the information transmitted on target buses using cycle-accurate microarchitecture-level simulators for a representative workload, (2) identify opportunities by correlating, for example, the number of self and coupling transitions with objective function (bus energy, temperature, crosstalk, etc.), (3) and design techniques that minimize the value of the objective function. Although, the technique is designed using a representative workload, it is likely to work well for any real application in the same domain due to the similarities in program characteristics. In fact, computer architecture continue to use similar methodologies to design efficient branch, load-value, and other prediction-based techniques to improve instruction-level parallelism in modern superscalar processors.

1.6 Our Contributions

As presented earlier, accurate modeling and cost-effective design of global signal interconnects is a critical issue in current and future nanometer-scale design. Since interconnect performance (wire delay) and energy dissipation depend closely on switching characteristics of the data stream, *activity-aware* modeling and design approaches are important. Furthermore, the introduction of Cu and low-k dielectrics exacerbate problems such as wire self-heating which need to be modeled, along with the impact of temperature on wire delay variability. Finally, newer design techniques are needed to deal with rising interconnect power dissipation and temperature since existing techniques are not effective in most real architectures, workloads, and applications.

The objective of this research is to provide a methodology to model and design signal interconnects in nanometer-scale ICs and address power, temperature, and performance concerns during early-stage design. To accomplish this goal, four research tasks were identified and novel contributions are made in each.

1.6.1 Activity-Aware Design Methodology

Our research is perhaps the first attempt that proposes and examines activity-aware design techniques for global signal buses. Existing techniques rely on worst-case estimates to design high-performance buses, resulting in overly-pessimistic energy, temperature, and clock cycle time estimates. Due to lack of accurate models suitable for early stage design exploration, interconnect design is done late in the design cycle, offering very limited opportunities to optimize the architecture for performance, power, and cost. In contrast, the methodology we propose examines typical applications, collects statistics for different types of data, and optimizes the design of target buses, all using early stage simulation. Thus interconnect design can be completed early in the design cycle and it can be used as a parameter in design space exploration.

1.6.2 Accurate Energy, Temperature, and Delay Modeling

We introduce accurate modeling techniques to help estimate the impact of activity-dependent interconnect energy dissipation, wire temperature rise due to Joule heating and delay variation due to temperature, using a microarchitecture-level simulator. In addition to self capacitance, our model incorporates the effects of capacitive coupling between adjacent as well as non-adjacent pairs of wires and repeater insertion on switching energy, the effect of lateral heat transfer between adjacent wires to estimate wire temperatures, and also estimates wire temperature gradients and its impact on wire delay, all of which were not available in earlier models. We estimate from simulations using our model for 130 nm technology node that, during the time interval taken to commit one billion instructions in the pipeline, high performance bus wire temperatures rise by 10-37°C for various SPEC CPU2000 benchmarks. This is solely due to Joule heat dissipated due to wire switching activities. In a future 45 nm technology node, wire temperature rise for the same set of benchmarks and simulation sample was found to be between 20-58°C. We observed that instruction and data bus wires attained absolute temperature in the range 80.3-104°C and 97.6-123.7°C, in 130 nm and 45 nm processors, respectively, during the course of our simulation, showing that signal lines attain significant temperatures too. Significant wire temperature gradients of magnitude between 16-25°C were found to be most common between the sending and receiving ends of the wires during the course of simulation. Notable correlation was found to exist between energy dissipation behavior and wire temperature rise in buses across time; short, intermittent cycles of high energy-dissipating switching activity trigger steep changes in temperature.

We also developed models that track the impact of changing wire temperature on timing/delay violations occurring in global signal buses during microarchitecture-level exploration. Results show that for a 130 nm processor with no power and thermal management the temperature-induced clock cycle time violations in an ALU result bus—which is on the critical path—is 2.27 per hundred bus references, averaged over ten programs in the SPEC CPU2000 workload. It increases to an average of 6.20 per hundred bus references for the same processor at the 45 nm technology node. Our analysis also shows that conventional techniques like bus encoding that seek to reduce energy dissipation and potentially wire temperatures have limited impact on alleviating temperature-induced delay violations.

1.6.3 Profile-Guided Optimization Techniques

Efforts to reduce bus energy dissipation, particularly in long global signal buses, are becoming increasingly important in nanometer-scale technologies as interconnects continue to aggravate performance, power, and cost. While dynamic encoding schemes have been proposed to reduce bus switching energy, they do not work well for correlated traffic such as those found in typical workloads like SPEC CPU2000 benchmarks. Hence, we develop static bus encoding techniques and present a methodology to design such schemes in an optimal manner. Being completely static, such schemes can be designed during early stage microarchitectural exploration and incur minimal run-time hardware area/cost, power, and latency compared to dynamic encoding logic. We use a microarchitecture-level simulator, profile representative samples of SPEC CPU2000 benchmarks to collect data, and use integer linear programming to design our encoding scheme. Results show that, for the SPEC CPU2000 workload, i.e., workstation/PC class processors, total bus energy dissipation reduced by as much as 22.79%/40.77% for data/instruction buses when our best static encoding scheme was applied. In contrast, existing dynamic bus encoding techniques yield only 4.19%/5.32% reductions for the same type of bus traffic.

1.6.4 Novel Thermal Optimization Methodology

Apart from bus energy, rising wire temperatures are also becoming an important issue to address in high performance buses since they affect wire delay and reliability. We propose a first-of-its-kind methodology to design temperature-aware encoding schemes by trading off some of the energy gains we obtain with static encoding techniques to achieve wire temperature reduction. In this methodology we add temperature constraints during energy optimization, and our ILP produces a static encoding scheme that reduces maximum/hottest wire temperatures by up to 15.23 K/16.17 K for data/instruction buses while still producing significant total bus energy reductions.

1.6.5 Performance-Oriented Adaptive Bus Design

The rate at which signals can be transmitted in a high-speed processor bus is decided based on the worst-case crosstalk pattern. This pessimistic estimation gives rise to significant performance penalties since the worst case never occurs or occurs with very low frequency in actual applications. Hence, we propose an adaptive bus design, called variable cycle bus (VCB) architecture, that examines incoming data patterns and transmits them using variable number of clock cycles, improving bus performance significantly. To maximize effectiveness of our adaptive bus architecture, we propose a profile-guided optimization approach—like the one described earlier in Section 1.6.3—to reorder and signal bits to minimize bus crosstalk. Results on SPEC CPU 2000 benchmarks, in a general-purpose optimization scenario, show a 29.35% reduction in 1+4r cycles, a 20.29% reduction in 1+3r cycles, and a bus performance improvement of 17.42% for a VCB with static reordering and signaling technique targeting bus crosstalk minimization.

1.7 Dissertation Outline

This remainder of this dissertation is organized as follows. Next, Chapter 2 presents a background on interconnect analysis and optimization for delay and power and provides a general overview of our experimental methodology and simulation infrastructure. Following that, in Chapter 3 we present the model for estimating activity-driven energy and temperature in processor buses and study the energy and temperature characteristics of data and instruction buses. Then, in Chapter 4, we present the model for estimating data and temperature-dependent delay variability and examine the impact of delay variability on the performance of a processor in current and future fabrication technologies. Next in Chapter 5, we discuss novel interconnect optimization techniques to reduce processor bus energy and temperatures. Then, we discuss delay optimization techniques in Chapter 6. Finally, we conclude and present directions for future work in Chapter 7.

CHAPTER 2

PRELIMINARIES

Integrated circuits (ICs) consist of two basic components: transistors and their interconnections. As more and more devices are integrated on a single die, wires or interconnections gain importance and play an important role in determining the speed, area, reliability, and yield of VLSI circuits [41]. In this chapter, we provide a brief introduction to some terminology used in the context of interconnect design and discuss interconnect analysis and optimization methods. We also discuss the role of architecture-level simulators in interconnect analysis and design. Finally, we outline the general experimental methodology followed in our experiments.

2.1 Interconnect Analysis Methods

Interconnect analysis as it applied to power and timing seeks to answer three questions: (1) what is the effective loading due to the interconnect? – this is necessary for driver/repeater sizing to minimize delay and to estimate power dissipation, (2) what is delay and slew at the receivers? and (3) what is the effect of switching of this and other neighboring nets on power dissipation and propagation delay? This analysis can be performed with dynamic circuit simulation, in which specific stimuli are applied to the circuits and interconnect in question. Unfortunately, this technique cannot be practically applied to the millions of transistors on a digital integrated circuit. Hence interconnect analysis is performed using simpler models. Interconnects in a VLSI chip can be grouped into three categories, based on their length, as discussed next.

2.1.1 Global, Semiglobal, and Local Wires

Since it is not possible to connect millions of transistors on the die using only one level of interconnect, multi-layer interconnect structures are commonly used. The metal layers closest to the Silicon (Si) substrate are called *local* interconnects/wires. The next few layers are called *semiglobal* or *intermediate* interconnects, and the top layers are called *global* interconnects. The wires in the global layers are wider and thicker and this yields shorter propagation (or RC) delays since wire resistance and hence delay is inversely proportional to the area of cross section. Consequently, these layers are used to route high performance buses in the core of the microprocessor. Wider and thicker wires at higher layers are also used to provide low-resistance power/clock distribution lines to different regions of the chip. Layer assignment, i.e., the decision to route a wire/net in the local, semiglobal, or global layer, is performed based on stochastic wire length estimates [42]. In our research, we are interested in power, temperature, performance, and reliability optimization of longer wires, i.e., semiglobal and global wires that are used to route high performance buses. These interconnects are analyzed using the models discussed next.

2.1.2 Interconnect Models: RC and RLC

Interconnects, in general, have three important electric characteristics: resistance (R), capacitance (C), and inductance (L). All three depends on the interconnect geometry and its position relative to the other surrounding structures. These parasitics affect circuit performance; capacitance adds load to driving gates, resistance, inductance, and capacitance all add signal delay, and inductive and capacitive coupling between interconnects add signal noise.

The circuit parasitics of a wire are distributed along its length and are not lumped into a single position. As long as the resistive component of the wire is small, and the switching frequencies are in the low to medium range, it is meaningful to consider only the capacitance component of the wire, and to lump the distributed capacitance as a single capacitor. This is the *simple capacitive model* and is not very accurate. On-chip metal interconnects of over a few millimeters in length have a significant resistance. The π-model lumps the total wire resistance of each wire segment into a single resistor R and represents the total capacitance as two capacitances of $\frac{C}{2}$ each. This model, called the *lumped-RC* model is, however, pessimistic and inaccurate for long interconnects, which are more adequately represented by a *distributed-RC* model. In practice, this model is represented as a π-ladder network. Similar to resistance and capacitance of interconnect, the inductance is also distributed over the wire. Thus, a *distributed RLC model* of interconnects, also known as the transmission line model, is the most accurate approximation of the actual behavior of interconnects.

2.1.3 Effect of Inductance on Global Signal Lines

In spite of shrinking dimensions and increasing clock frequencies in nanometer-scale technologies, it has been shown that inductance can be safely ignored for global signal lines that are longer than 10 mm [2,43]. This is due to various factors discussed next. First, it has been shown that, for long global signal lines, the signal response to a step input is over-damped when the line is modeled using the complex distributed RLC model. This response can be approximated using a distributed-RC model, without significant error [43]. Second, inductance is not a significant problem in minimum-width global lines as much as it is in clock and power/ground lines that are several times minimum width. It has been estimated that inductance becomes an issue in a global line only if its width is at least eight times the minimum width [2]. Third, in high-performance buses that we consider in this research, designers ensure that inductive effects are minimized by ensuring that current return paths for worst-case input patterns are kept within limits. This is normally achieved by placing power/ground planes above and/or below the layer in which the high-performance bus is routed and also by routing shield wires in the same layer as the bus [44]. Finally, in the recent times, architectural trends have shifted toward improving power/performance (or Watt/MIPS) efficiency by using shorter pipelines and multi-core architectures, compared to just improving performance by increasing clock speed. Thus, in current and future generation microprocessors, clock frequencies are not expected to increase exponentially as predicted until a few years ago. This trend also contributes to keeping inductive effects in check for global lines.

Due to the reasons outlined above, we do not consider inductive effects in our work. Using an RC-model, interconnect energy can be estimated as discussed next.

2.1.4 Energy Estimation

Self transitions are defined as transitions on the *self* or *area capacitance* which is the parasitic capacitance between a bus line and the ground/V_{DD} plane. *Coupling transitions* are defined as transitions that occur on the *coupling capacitance* which is the parasitic capacitance between two wires on the same plane. Figure 2.1 shows self and coupling capacitances for a 5-bit bus. Note that there can be two types of coupling capacitances for a wire of length l_{wire}: adjacent coupling capacitance $C_{c1} = l_{wire} \times c_{i,i\pm 1}$ and non-adjacent coupling capacitance $C_{cx} = l_{wire} \times c_{i,i\pm x}$, where $x \geq 2$. The adjacent coupling capacitance is the most dominant. Hence it is most often considered in energy and delay estimation and other (non-adjacent) capacitances are ignored.

Self transitions in a wire are of two types: charge ($0 \rightarrow 1$) and discharge ($1 \rightarrow 0$), and coupling transitions in a pair adjacent wires are of three types: *coupling charge transitions* ($00 \rightarrow 01$,[1] $00 \rightarrow 10$, $10 \rightarrow 11$, and $01 \rightarrow 11$), *coupling discharge transitions* ($01 \rightarrow 00$, $10 \rightarrow 00$, $11 \rightarrow 10$, and $11 \rightarrow 01$), and *toggle* transitions ($01 \rightarrow 10$ and $10 \rightarrow 01$). Note that if the total number of self and coupling (charge, discharge, and toggle) transitions is reduced, bus energy dissipation will reduce significantly.

The *energy consumption* and *energy dissipation* of a bus in a given time interval t are given by:

$$E_{cons,avg} = [N_s \cdot C_w + C_{c1} \cdot (N_c + 2 \cdot N_t)] \cdot V_{DD}^2 \cdot f_{clk} \cdot t, \tag{2.1}$$

$$E_{diss,avg} = [N_s \cdot C_w + C_{c1} \cdot (\frac{N_c}{2} + \frac{N_d}{2} + N_t)] \cdot V_{DD}^2 \cdot f_{clk} \cdot t, \tag{2.2}$$

where $C_w = C_{line} + C_{rep} = l_{wire} \times c_{line} + C_{rep}$ is the self capacitance of the wire including the contribution of repeaters, N_s is the total number of self-charge transitions recorded on the bus in time interval t, N_c, N_d, and ,$\mathrm{N_t}$ are the number of coupling-charge, coupling-discharge, and coupling-toggle transitions, respectively, recorded in the same interval. Thus, only charging transitions that require current flow from the power supply to charge the parasitic capacitances are used to determine energy consumption, whereas current flow from the power supply (during charging)

[1] For two lines i and j, this notation represents the transition: $V_i^{in} V_j^{in} \rightarrow V_i^{fin} V_j^{fin}$.

and current flow into the ground (during discharging) of the parasitic capacitances account for energy dissipation. Energy consumption and dissipation are equal on the average, though their instantaneous values may be different.

2.1.5 Delay and Performance

When designing circuits it is necessary to ensure that a signal is fully transmitted across a wire in a given time. This time should be at least the *propagation delay* of the wire which depends on wire and driver sizes and also on the interaction with neighboring wires, which is referred to as *inter-wire crosstalk*. Due to crosstalk, the propagation delay t_P of a wire (called the victim), which is a function of transitions in its neighboring wires $k-1$ and $k+1$, can be expressed as follows, including the effect of load (receiver) capacitance [45]:

$$t_P = 0.69(R_D + R_w) \cdot C_r + g_0 \cdot C_{line} \cdot (0.38R_w + 0.69R_D), \qquad (2.3)$$

where R_w and R_D are the wire and driver resistances, respectively, C_r is the input (gate) capacitance of the receiver, g_0 is the *delay correction factor* due to inter-wire coupling between wires separated by the minimum spacing and is a function of the *capacitance ratio* $r = \frac{C_{c1}}{C_w}$. The wire resistance R_w is estimated using the resistivity at a design temperature of 100°C. The various crosstalk conditions occurring when the victim wire k experiences a rising ($0 \rightarrow 1$) transition (denoted as $\uparrow$) are listed in Table 2.1. A corresponding table of delay factors can be constructed for a victim wire experiencing a falling ($1 \rightarrow 0$) transition ($\downarrow$).

In the worst case—toggle or oppositely switching transitions on both sides of the

Crosstalk mode	$k-1$, k, $k+1$	Delay factor (g_0)
mode-0	↑, ↑, ↑	1+0r
mode-1	↑, ↑, -	1+1r
mode-2	↑, ↑, ↓	1+2r
mode-2	-, ↑, -	1+2r
mode-3	-, ↑, ↓	1+3r
mode-4	↓, ↑, ↓	1+4r

Table 2.1. Bus crosstalk conditions and models for a rising transition in the middle (victim) wire.

victim—the delay is:

$$t_{wc} = 0.69(R_D + R_w) \cdot C_r + (1 + 4r) \cdot C_w \cdot (0.38R_w + 0.69R_D). \qquad (2.4)$$

It is clear that width of the clock pulse to the circuit should be more than t_{wc} to ensure that the signal propagates completely to the destination, i.e., $t_{bus\_clk} \geq t_{wc}$. To ensure that this does not impact performance, repeaters/buffers are used to divide long wires into several sections and hence reduce propagation delay. Assuming that the size of each repeater is h times the size of a minimum-sized inverter (which is technology-dependent) and k is the number of repeaters needed to achieve optimum delay on the interconnect, these can be calculated using:

$$h = \sqrt{\frac{R_0 \cdot C_{int}}{C_0 \cdot R_{int}}} \quad \text{and} \qquad (2.5)$$

$$k = \sqrt{\frac{0.4(R_{int} \cdot C_{int})}{0.7(C_0 \cdot R_0)}}, \qquad (2.6)$$

where $C_{int} = c_{line} + 4 \cdot c_{i,i\pm 1}$ is the total per-unit length capacitance of a wire leading to the worst-case delay impact, C_0 are R_0 are the capacitance and resistance of a minimum sized inverter, and $R_{int} = r_{line}$ is the per-unit length wire resistance [46].

2.2 Interconnect Optimization Techniques

Several techniques have been proposed to ensure that interconnect power and performance are not affected due to technology scaling. We discuss these next.

2.2.1 Data Encoding

In general, system-level encoding techniques fall under three categories, based on whether they use redundancy in space (extra number of bus lines), time (extra number of cycles) and voltage (number of distinct voltage levels) [47]. In particular, use of time redundancy has been demonstrated to be as effective as the space redundancy for decreasing the average switching activity and issues due to extra cycle overheads have been addressed by using compression [48–50]. Different modes of signaling—*level* and *transition* signaling—can also be used to reduce bus switching activity.

The bus-invert (BI) code is a low-power encoding scheme designed to limit the average power of the bus [51]. It performs well when patterns to be transmitted are randomly distributed in time and no information about pattern correlation is available. Therefore, this method is most appropriate for encoding the information on data buses. A redundant control line INV is needed to signal to the receiving end of the bus the encoding mode in the current cycle. The encoding depends on the Hamming distance (i.e., the number of bit differences) between the value of the encoded bus lines at time $t-1$ (also counting the redundant line at time $t-1$) and the corresponding value at time t. The Hamming distance is compared to $\frac{n}{2}$, where n is the bus width (assuming n is even without loss of generality). If the Hamming distance between two successive patterns is larger than $\frac{n}{2}$, the current value is transmitted with inverted polarity and the control line is asserted; otherwise, the current value is transmitted as is, and the INV line is de-asserted.

If the words transmitted on the bus are independent and uniformly distributed, the average number of transitions per clock cycle is lowered by less than 25% of the original value, due to the binomial distribution of the distance between consecutive patterns [52]. Major drawbacks of the BI technique are related to the required redundant bus line and the overheads due to the logic to implement the encoder to decide whether the Hamming distance exceeds $\frac{n}{2}$. The encoding latency, in particular, is quite significant as discussed next.

In BI, encoding consists of three sequential steps. First, the Hamming distance is computed. To do this, the current n-bit pattern and the previous n-bit pattern that was transmitted on the bus in the previous cycle are bitwise XOR-ed and the number of "1"s in the result is counted. This step requires a constant time operation for bitwise XOR and $O(n)$ to $O(\log_2 n)$ time for counting, depending upon the counter structure used. In the second step, the Hamming distance is compared with $\frac{n}{2}$ to check which is greater; this can be completed in $O(n)$ to $O(\log_2 n)$ time, again depending on the hardware structure used. Finally, the current pattern is inverted or sent as-is and this takes constant time. Thus, BI encoding takes at least $O(\log_2 n)$ time.

More recently, odd/even bus invert (OEBI) [53] and coupling-driven bus invert (CBI) [54] encoding schemes, designed to reduce transitions on the coupling capacitance between adjacent bus lines, were proposed. In OEBI, even and odd bit positions can be encoded (with bus inversion) independently and two invert lines are used to indicate one of four modes of transmission: 00–none of the bits are inverted, 01–only the even-numbered bits are inverted, 10–only odd-numbered bits are inverted, and 11–all bits are inverted. This is based on the observation that by inverting only the odd or even bits, a coupling toggle transition can be reduced to a coupling charge/discharge transition [53]. The scheme assigns weights of 1 and 4 to coupling charge/discharge and toggle transitions, respectively, to estimate coupling energy dissipation. Based on the current and previous input patterns, the total coupling energy dissipation for each of the four modes is estimated. Then the mode that will result in the least coupling energy dissipation is chosen and data is transmitted on the bus in that form. In a similar manner, the CBI encoding technique examines pairs of adjacent bits in the same position for the current and previous input patterns and estimates coupling activity. The differences here are: (1) only one invert line is used to indicate whether the transmitted data is in inverted or non-inverted form; and (2) it uses weights of 1 and 2 for coupling charge/discharge and toggle transitions, respectively. Note that neither OEBI nor CBI considers self transitions to decide the inversion mode while BI considers only self transitions.

Bus encoding is also often used to reduce crosstalk. Crosstalk-aware encoding schemes can be one of two types: those that have memory or those that are memoryless. If an encoding scheme has memory then each codeword is dependent on the word that came before it. Thus, each codeword has its own codebook of valid words that can come after it. On the other hand, if an encoding is memoryless then any codeword can follow any other codeword. The minimum number of wires needed to encode 32 bits with memory is 40 and without memory is 46 [55]. Thus the extra wiring overhead for an encoding scheme with memory is 25% and 44% for optimal encoding without memory.

2.2.2 Wire Spacing and Shielding

Inserting V_{DD}/GND wires known as *shields* is a popular method to avoid crosstalk in high-performance buses. Signal isolation due to the presence of shields prevents both noise and increase in delay due to coupled lines switching. A dense fabric interconnect architecture with shield lines inserted after every signal wire was proposed in [56]. Shield insertion also reduces inductive effects because it creates a shorter return path to ground for the current flowing through signal wires. However, inserting shield wires between every pair of signal wires results in large area/costs, increases wire congestion and may end up requiring more metal layers leading to higher production costs. Alternatively, wires can be simply spaced apart to produce a similar solution. Though spacing does not eliminate coupling noise, it reduces the value of the coupling capacitance—since capacitance is inversely proportional to the spacing—and at the same time reduces power dissipation since the total capacitance load of the line also decreases. In many cases, this is a significant gain compared to shielding which eliminates the noise at the cost of extra power dissipation [57].

2.3 Architecture-Level Simulators and Early-Stage Design

At the very early stages of design definition, microarchitects start with analytical cycles-per-instruction (CPI) performance models that lead to trace or execution-driven, cycle-by-cycle simulators. Full or sampled benchmark traces are processed through such simulators, driven by a microarchitecture parameter file. The goal of this design space exploration phase is to optimize the choice of microarchitectural parameters for CPI performance under design constraints known at that stage. The performance model is typically written in a standard systems programming language such as C or C++ and is designed to project execution times (in cycles) for input application traces; it typically does not model the actual execution of the instructions, but only the execution timing. More recently, power dissipation models that are based on counting the number of transitions occurring in microarchitecture blocks have also been added to these simulators.

Several architecture-level simulators have been developed and used in the academia and industry: Wattch [58], SimplePower [59], TEM^{2}P^{2}EST [60], WArPE [61], Sim-Panalyzer [62], IBM Turandot/PowerTimer [31], AccuPower [63], and HotSpot [64]. Interconnect/bus models used in these simulators suffer from many drawbacks. First, none of the existing simulators have models for estimating inter-wire coupling activity dependent power consumption and delay. For example, the SimplePower tool, which models only memory system buses (between different levels of caches and/or main memory), uses an interconnect model that considers only the self-capacitance of bus lines calculated based on an empirical formula [65]. The Wattch simulator which models only the result bus in the microarchitecture also does not take into account inter-wire coupling activities when estimating power dissipation. Thermal models for buses are not available in most current simulators. The HotSpot tool addresses this need to some extent, but it contains a temperature model for the interconnect system as a whole rather than for each bus and hence cannot track activity-dependent temperature changes in key processor buses [66]. Temperature gradients and delay variations cannot be estimated using this tool.

2.4 Our Experimental Methodology

2.4.1 Interconnect Geometry and Technology Data

For all the interconnects considered in this work, we assumed that it was routed in the top-most layer metal. A representation of wires in this layer is shown in Figure 2.1.

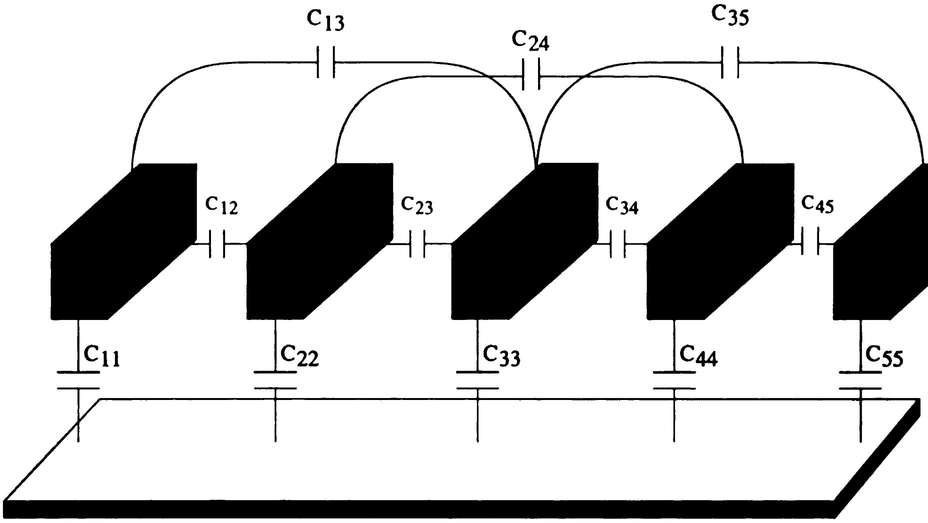


Figure 2.1. Layout of wires routed in the top-most layer metal. Self and coupling capacitances are shown. The bottom plate represents the V_{DD}/GND plane.

Values for wire geometry (wire width, spacing, etc.) and technology and equivalent circuit parameters, like capacitance and resistance of a global line for various nanometer-scale technologies were obtained from the ITRS document and are listed in Table 2.2. Note that wire spacing is assumed to be equal to wire width per ITRS [1]. In this work, we use 130 nm and 45 nm as the representative technologies for a current generation and a future-generation microprocessor and compared our results for these designs.

In current generation microprocessors, a global signal bus is typically a few millimeters long; we consider a bus of length 6 mm using the numbers reported in [44]

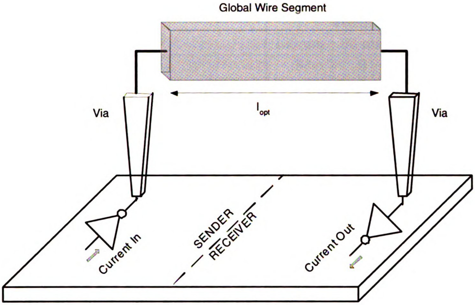


Figure 2.2. Wire segment of length l_{opt} between two repeaters.

for a Pentium-4 microprocessor. Using this length (l_{wire}), we estimate the number of repeaters (k) that need to be inserted to enable non-inverting transmission using Equation 2.6, and then we find the inter-repeater segment length $l_{opt} = \frac{6 \times 10^{-3}}{k}$. In the remainder of this work, all experiments and analysis focus on a single wire segment of length l_{opt}, driven by a sending end repeater of size h and connected to a receiving end repeater of the same size, as shown in Fig 2.2. In addition to its self capacitance, this wire segment has a capacitance, due to its sending and receiving end repeaters, that can be calculated as: $C_{rep} = h \times C_0$, where C_0 is the sum of the input and output capacitances of a minimum sized inverter.

2.4.2 Parasitic Capacitance Extraction

The ITRS roadmap provides values only for self and adjacent-wire coupling capacitance for current and future technology nodes. Hence, to estimate the coupling

	Technology node			
Parameter	130 nm	90 nm	65 nm	45 nm
Number of metal layers	8	9	10	10
Wire width, w_i (nm)	335	230	145	103
Wire thickness, t_i (nm)	670	482	319	236
Relative permittivity of dielectric, ε_r	3.3	2.8	2.5	2.1
Thermal conductivity of dielectric, k_{ild} (W/mK)	0.6	0.19	0.12	0.07
Clock frequency, f_{clk} (GHz)	1.68	3.99	6.73	11.51
Supply voltage, V_{DD} (V)	1.1	1.0	0.7	0.6
Maximum current density in a wire, j_{max} (MA/cm^2)	0.96	1.5	2.1	2.7
Height of inter-layer dielectric, t_{ild} (nm)	724	498	329	243
Resistance of minimum size inverter, R_0 (kΩ)	6.23	9.04	9.6	13.2
Capacitance of minimum size inverter, C_0 (fF)	4.65	3.14	2.25	1.5
Self capacitance of wire, c_{line} (pF/m)	44.06	32.77	25.07	19.05
Adjacent coupling capacitance, $c_{i,i\pm 1}$ (pF/mm)	91.72	76.84	68.42	58.12
Non-adjacent coupling capacitance, $c_{i,i\pm 2}$ (pF/mm)	6.49	4.65	3.56	2.76
Non-adjacent coupling capacitance, $c_{i,i\pm 3}$ (pF/mm)	2.53	1.76	1.29	0.98
Resistance of wire, r_{line} (kΩ/m)	98.02	198.45	475.62	905.05
Optimal repeater size, h	74.95	70.25	51.77	49.45
Optimal # of repeaters for non-inverting bus, k	6	8	12	16
Coupling ratio including effect of repeaters, r	2.065	2.329	2.716	3.039

Table 2.2. Technology, wire geometry, and equivalent circuit parameters for topmost layer interconnect. Values in top eight rows are from the international technology roadmap for semiconductors (ITRS) document [1]. Values listed in the next three rows are from Mui et al. [2]. The values for the self and coupling capacitances were extracted using the FastCap tool and the value for r_i was calculated using the formula $r_i = \rho_{Cu}/(w_i \cdot t_i)$, where $\rho_{Cu} = 2.2 \times 10^{-8}\Omega$-m. Values of h and k were found using expressions given in Section 2.1.5 and $r = c_{i,i\pm 1}/(c_{line} + h \times C_0)$.

capacitances between all pairs of wires (adjacent as well as non-adjacent wire pairs), we employed the publicly available three-dimensional capacitance extraction program called FastCap [7]. Using the wire geometry parameters from ITRS (see Table 2.2 for values) to model a coplanar global bus layout, similar to the one shown in Figure 2.1, we extracted values of self and all coupling capacitances for the middle wire of a 32-bit bus. Figure 2.3 shows the percentage distribution of these capacitances for various technologies. From the figure, we observe that, for current 130 nm and 90 nm technologies, non-adjacent coupling capacitances are somewhat non-negligible (they contribute ≈10%), while even in a future 45 nm node, non-adjacent capacitances account for about 8% of the total capacitance. Our energy model which is described in a later chapter considers the effect of two non-adjacent coupling capacitances, $Cc2$ and $Cc3$, for better accuracy.

2.4.3 Simulation Infrastructure and Verification of its Correctness

Computer simulators have been used for a long time to study both hardware and software behavior. They allow the collection of information and statistics during the execution of programs. Various types of information, such as memory profiles, instruction profiles, and timing statistics, can be gathered from these simulators. For this research, we use the `sim-outorder` out-of-order processor simulator from the SimpleScalar microarchitecture tool set, which is very widely used in academia [67]. Many microarchitectural simulators used in the industry also closely resemble and/or are derived from SimpleScalar or its derivatives [31, 58–64]. We added several enhancements to the `sim-outorder` simulator to facilitate our analysis and optimization

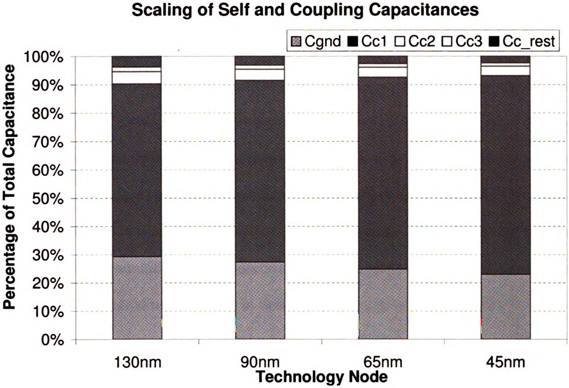


Figure 2.3. Distribution of self and coupling capacitance values for the middle wire of a 32-bit bus extracted using the FastCap tool [7]. Cgnd = self capacitance of the wire; Cc1 = coupling capacitance between the wire and its adjacent neighbor; Cc2 = coupling capacitance between the wire and a non-adjacent wire with 1 wire between them; Cc3 = coupling capacitance between the wire and a non-adjacent wire with 2 wires between them; Cc_rest = sum of coupling capacitances between the wire and other wires with 3 or more wires between them. For current and near-future ITRS technology nodes (up to 45 nm), non-adjacent coupling capacitances are somewhat non-negligible—they contribute approximately 8–10%.

efforts. These are described next.

Support for analyzing bus data: We added support for tracing and analyzing the data transmitted on high performance processor core buses. The original `sim-outorder` contains only a functional model of a superscalar processor and does not have the ability to track the data that is transmitted between the microarchitectural blocks in the pipeline. We modified the simulator to track and analyze, on a cycle-accurate basis, the data transmitted on load/store address, load/store data, instruction, and result buses in the processor core.

Wire energy, temperature, and delay models: We also added our wire energy, temperature, and delay models to the simulator. While energy dissipation and delay of our target buses—including the temperature impact on delay—can be estimated on a per-cycle basis, temperature estimates can be obtained at a coarser granularity, i.e., after every 100K cycles or so. This is because temperature is a slow-changing effect that does not warrant per-cycle estimation. More details on how we determine the granularity of temperature simulation depending on the fabrication technology used are discussed later in Section 3.5.3.

Integration with other thermal analysis tools: Recently, a tool called HotSpot [64], also based on SimpleScalar, was developed to estimate substrate (active layer) temperatures using the Wattch model for energy estimation [58]. Even though the on-chip interconnect system is a major contributor to the power budget, it was not modeled accurately in HotSpot. We have integrated our models with HotSpot, thus creating a microarchitecture-level simulation tool for full-chip energy and thermal analysis.

As a result of our enhancements to the simulator, the running time is somewhat longer. The original `sim-outorder` without enhancements executes ~200K instructions per second [67] whereas our modified simulator executes ~110K instructions per second while running detailed energy and temperature simulations at the granularities described earlier in this subsection. To reduce simulation time for analyzing a large number of programs on our simulator, we used a shared Linux cluster for our experiments [68].

We verified the correctness of our modified simulator with regard to four aspects, as discussed next.

Functional correctness: All the changes we made to the simulator add to its instrumentation capabilities and do not change it functionally, with regard to the microarchitectural model it seeks to implement. We verified this in two ways, as discussed next. First, we executed and compared the outputs for a suite of six microbenchmarks, supplied along with the SimpleScalar toolset, using the original (unmodified) simulator and our modified version. As expected, the program outputs from both versions matched exactly. Second, we compared several performance metrics recorded by the simulator—number of instructions executed, L1/L2 cache misses, branch misprediction rate, etc.—and found that these matched in the original simulator and our modified version, for the six microbenchmarks we tested. These tests show that the functional correctness of our modified simulator has not changed compared to the original one.

Instrumentation correctness: The original `sim-outorder` simulator contains a detailed-enough microarchitectural model that enabled us to gather data transmitted on our target buses, in each cycle. Thus, instruction addresses and instructions were gathered from the program counter and the fetch stage of the simulator, respectively, data addresses by computing the target address for load/store instructions, load/store data by monitoring L1 cache reads/writes, and ALU result bus data by monitoring the outputs of the functional units in the execute stage. As such, the instrumentation capabilities we added to the simulator are correct by design.

Model correctness: We tested if the models we constructed represent actual energy/thermal behavior of buses consistent with previously-known data and/or estimates. For our energy model, discussed later in Section 3.3, results were compared with circuit simulation of a distributed-RC wire using the Cadence Spectre simulator. Our model yielded energy results that were only about 4.53% different compared to those from Spectre, faster and with much less complexity. Our thermal model, discussed in Section 3.4, is based on the well-known analogy between electrical and thermal quantities that has been used widely in earlier work to model chip thermal structures [66, 69–71] and verified using finite element modeling (FEM) simulations [72, 73]. The average and maximum temperatures obtained using our model, while running SPEC CPU 2000 benchmarks on the simulator, were consistent with previously published data in [66], although our model estimated bus energies more accurately considering actual bus traffic values, interconnect temperatures at a finer granularity, and tracked spatiotemporal variation of temperature, all of which were absent in earlier models. The worst-case temperatures that global signal lines may potentially attain, assuming they carry currents at maximum density all the time, were estimated using FEM-based techniques in [72, 73]. Signal lines, which are the focus of our work, do not carry currents at maximum density all the time and hence their temperatures are likely to be somewhat less than estimates obtained using worst-case FEM analysis. We verified that results using our model were consistently lower than worst-case estimates and remained so for the different technology nodes we tested: 130 nm, 90 nm, and 45 nm.

Implementation correctness: We also tested that modifications were implemented correctly in the simulator and that desired outputs were obtained. For all the six microbenchmarks, we collected tracedumps of various buses using our simulator and verified manually that the data in the tracedump matched the expected value for that type of data. For example, each entry in the instruction address tracedump should match the program counter value which is in a known range of memory addresses and each entry in the instruction tracedump should correspond to known instructions in the processor's instruction set architecture. We found these to be true in all the tracedumps we tested. We also prepared several small synthetic traces of data streams and verified that results obtained from hand calculations matched those using equations from our energy model implemented in the simulator.

2.4.4 Target Systems and Benchmarks

The SimpleScalar platform can simulate various RISC microarchitectures. For our work, we use the Alpha 21264 microarchitecture representing general-purpose superscalar processors. The Alpha 21264 architecture is modeled as a 4-issue, superscalar processor with out-of-order execution and with 32-bit address, 64-bit data, and 128-bit (fetch width=4) instruction bus between the processor and L1 cache [74]. Other details of the microarchitecture and memory system for our target system is presented in Table 2.3.

For evaluation on the Alpha target system, we use the SPEC CPU2000 benchmark suite which consists of 26 programs drawn from real user CPU-intensive applications [75]. The little-endian SPEC benchmark executables we used were downloaded from the SimpleScalar Website [76]. These programs were compiled for the Alpha 21264 instruction set using a Compaq Alpha compiler with SPEC peak settings and included all linked libraries. We ran our experiments using the `ref` input set from the SPEC CPU2000 suite.

Since the time taken to simulate an entire SPEC CPU2000 benchmark is very long—typically several days on a cycle-accurate simulator—we used the 100 million

Processor Core	
Clock rate	1.68 GHz (130 nm), 11.51 GHz (45 nm) [1]
Fetch/Issue width	4 each
LSQ	8 entries
Memory System	
P↔L1 bus	Non-pipelined; 64-bit data and 128-bit instruction
L1 D-cache	Virtually-indexed physically-tagged (VIPT), 64KB, 2-way set associative, 64B block size, LRU policy, 3 cycle hit latency, write-through cache.
L1 I-cache	Virtually-indexed virtually-tagged (VIVT), 64KB, 2-way set associative, 64B block size, LRU policy, 1 cycle hit latency
L1 MAF	8 entries
L1↔L2 bus	Non-pipelined; 128-bit data/instruction lines and 38-bit address lines (21 bits for block index and 17 bits for tag)
L2 cache	Physically-indexed physically-tagged (PIPT), 2MB, direct-mapped, 64B block size, LRU policy, 12 CPU cycles hit latency, write-back policy, operating at 2x CPU clock cycle
L2↔M bus	Non-pipelined; 64-bit data/instruction lines and 38-bit address lines

Table 2.3. Configuration of our target system and benchmarks. This processor-memory system configuration is based on the Alpha 21264 processor.

single simulation points recommended by the SimPoint toolset to collect results only a representative slice of the program [77,78]. Although the accuracy of representative samples from SimPoint has not been explicitly validated using energy/temperature metrics, its use in design/evaluation of microarchitecture-level energy reduction techniques is widespread in literature. Several works that use phase classification techniques like SimPoint for microprocessor energy evaluation have been surveyed in [79].

CHAPTER 3

ACTIVITY-DRIVEN ENERGY AND TEMPERATURE MODEL

Accurate early stage modeling techniques for signal interconnect energy dissipation and temperature are becoming necesary for current designs. This chapter describes a detailed energy model and a first-of-its-kind thermal model for interconnects [80,81].

3.1 Introduction

As fabrication technologies scale down, interconnects are becoming the dominant factor in determining performance, power, cost, and reliability characteristics of a system. Interconnect scaling impacts performance because wire delay has continued to increase relative to that of logic. In recent years, power density in microprocessors has doubled every three years, primarily because feature sizes and clock frequencies have scaled faster than operating voltages [82]; this rate is expected to increase further in future technology generations [64]. The on-chip interconnect system is already the most important contributor to dynamic power; in current microprocessors (130 nm technology), interconnects are reported to contribute about 51% of the total on-chip dynamic power dissipation and global signal lines—address, instruction, data, and control buses routed in the top-most layer metal—about 21% [4]. As technology scales down, dynamic power dissipation will still remain important even as leakage power increases. It has been estimated that even in the 45 nm technology node, dynamic power will contribute to about 46% of the total power dissipation [2]. Supply voltage scaling and smaller sizes will reduce dynamic power dissipation due to logic in future technologies at a faster rate than in interconnects and hence, interconnect dissipation will contribute a larger share to total dynamic power. Rising interconnect power dissipation will lead to localized Joule heating and temperature rise in wire metal that can affect wire delay due to temperature-dependence of resistivity and/or cause wire breakage due to thermal stresses and electromigration.

As power densities continue to increase, thermal effects in wires are becoming important due to the reasons outlined next. Signal transmission over a line/wire i is associated with current flow, which results in I^2R power dissipation, where I is the magnitude of current and R is the resistance of the wire. This dynamic switching power depends on: (1) the *self capacitance* (capacitance between the line to ground) of the wire c_{line}, (2) the *coupling capacitance* $c_{i,j}$ between line i and any other line j, (3) the self and coupling activity factors (which in turn depend on self transitions on line i and coupling transitions between line i and any other line j, respectively), (4) the supply voltage, and (5) the bus clock frequency. Advances in technology have resulted in ever-higher values of $\frac{c_{i,j}}{c_{line}}$ due to higher wire aspect ratios and smaller inter-wire spacings; among all $\mathrm{c}_{i,j}$, the adjacent coupling capacitance ($\mathrm{c}_{i,i\pm 1}$) dominates the other (non-adjacent) coupling capacitances. With newer technologies, bus clock frequency has also continued to increase. The supply voltage is scaling down but at a rate not enough to offset the rate of increase in the other two. Thus, the net effect is that the I^2R power is continuing to increase as technology scales down, and consequently local heating in wires is becoming a concern. Further, since global signal wires are separated by multiple layers of low-K dielectrics from the substrate that is connected to the heat sink, and since these dielectrics have poor thermal conductivities, heat cannot be removed from the wire efficiently. Energy dissipation and/or thermal effects in global signal lines are further aggravated due to the following reasons: (1) increasing use of repeaters in long signal lines to reduce delay leads to higher energy dissipation [46]; (2) a steady increase in the number of metal layers, particularly the number of global metal layers, also increases overall energy dissipation; and (3) long via separations in upper metal layers contribute to higher average wire temperatures—vias are normally better thermal conductors than surrounding low-K dielectrics [83].

By virtue of their carrying smaller currents than power supply lines, energy dissipation and thermal characteristics of signal (both clock and data) lines have not been the subject of serious study. But this will need to change as clock frequencies increase with technology scaling. Higher frequency also means that the large fluctuating line currents drawn by the bus driving circuitry can influence resistive and inductive voltage drop in power supply lines, since long global signal lines present a high load capacitance. In this work, we develop a model for activity-dependent bus line energy dissipation and temperature rise, and apply it to different types of microprocessor core buses. While we do not study clock lines in this work, our model can be easily applied to thermal analysis of clock networks and estimate temperature impact on signal delay, skew, and reliability.

The dynamic power dissipated in a bus wire, which ultimately determines its temperature as discussed earlier, is both time and information dependent. It depends on the type of information (address, instruction, data, or control) being carried by the bus because the information type influences the self and coupling activity factors;

for example, the number of coupling transitions are expected to be higher for data streams that are more random in nature than for others. The type of information also directly influences the temperature characteristics of the wire because of the presence of unequal numbers of idle cycles between successive transfers; address and instruction buses typically carry new information every cycle as opposed to data buses where more idle cycles are likely to be present between data accesses. These idle cycles, during which no power is dissipated in the bus lines (assuming they hold the last value that was transmitted), present opportunities for cooling. Hence, interconnect thermal models that estimate temperature and reliability based on the assumption that all bus lines carry the maximum RMS current density (worst-case scenario) [83, 84], and models that use switching activity factors to estimate average self-heating power and determine temperature rise [66], may result in inaccuracies. This may, in turn, lead to incorrect interconnect lifetime prediction, since dynamic heating and cooling effects are not taken into account. Also, designers will be forced to allow higher-than-required safety margins and, as a result, the system will incur higher packaging costs. Hence, energy dissipation and thermal effects in buses are best studied using microarchitectural simulators and real workloads; in this work, we present models to facilitate this.

Detailed thermal models and workload-based studies for estimating temperature distributions in substrate [64] and interconnects are essential for facilitating early-stage design of future high-performance processors. For such designs, a pessimistic temperature assumption will lead to costly and perhaps unrealistic guard bands and high cooling system costs. On the other hand, an optimistic assumption will lead to underestimation of the chip power and leakage, and may lead to shorter lifetime and lesser reliability. Higher wire temperatures can have a dramatic impact on performance since temperature directly affects wire delay. Typically, the Elmore delay of an on-chip wire increases approximately 5% for every 20°C rise in temperature [37]. In addition to its absolute temperature, wire delay also depends on the temperature gradient between the sending and receiving ends. The growing popularity of chip multiprocessing (CMP) and simultaneous multi-threading (SMT) will increase bus switching activities, since, potentially, uncorrelated data from different streams are transferred on the same bus, resulting in higher per-wire energy dissipation and temperatures. Thus, realistic temperature models and early-stage estimates are essential for meeting design goals and avoiding temperature-induced problems in silicon.

The organization of the rest of this chapter is as follows. Section 3.2 briefly reviews related work. Next, in Section 3.3 and 3.4, we present our energy dissipation and thermal models for global signal lines. Following that, in Section 3.5, we discuss our simulation environment and methodology. Then, in Section 3.6, we present results from simulations by applying our models in an execution-driven simulator. Finally, we summarize in Section 3.7.

3.2 Related Work and Our Contributions

Some methods for architecture-level interconnect power analysis have been proposed [59, 85]. Earlier modeling methods estimated bus energies based on self transitions only [59], whereas recent models also consider adjacent inter-wire capacitances for energy calculations [85]. Thermal effects in interconnects and their implications for performance, current density, and reliability have been studied in [21, 37]. Recently, interconnect thermal models have been proposed in [66, 83]. But these models either perform a worst-case analysis using maximum current metrics suitable only for power supply lines [83] or consider average switching activities [66]. Such approaches are not suitable for analyzing signal lines since: (1) signal lines carry much less current than power supply lines, and (2) their energy dissipation and thermal characteristics are tied to actual traffic patterns (with intermittent idling) carried on the bus. A large body of work exists on low-power bus encoding, many of which also use bus energy models similar to ones described in [59] or [85]. Some of the older bus encoding schemes have been surveyed in [86]. Newer schemes include odd/even bus-invert [53], coupling-driven bus-invert [54], transition pattern coding [87], and leakage-aware bus encoding [88]. The contributions of this work are outlined next.

First, we present an accurate model to estimate bus line energy dissipation that can be used in a trace-driven setup or in an execution-driven simulator. Existing bus energy models, like the one proposed in [85], only estimate energy dissipation considering the bus as a whole, not in each line, whereas our model is capable of estimating energy dissipated in each bus line. Also, these models do not account for the non-uniform dissipation of energy across the wire length, which we do in our model. As we shall see later, these factors are necessary to model dynamic temperature effects in buses, both temporally and spatially, across wires. Our bus model is also more accurate because it considers the effect of capacitive coupling between adjacent and non-adjacent wire pairs on switching energy in addition to energy dissipated in the self capacitance. Our work is the first to show that switching transitions in parasitic capacitances between non-adjacent wire pairs account for a significant (7-8%) portion of the total energy dissipation and hence this contribution should not be neglected in bus energy models. Further, we model the effect of repeaters, which increase the self capacitance and hence self energy dissipation. This is so because the output capacitance of a repeater adds to the self capacitance of the line segment that it drives, and the input gate capacitance of a repeater adds to that of its input line.

Second, using our bus line energy dissipation model, we study the effectiveness of some existing low-power bus encoding techniques when used for data and instruction bus encoding. To our knowledge, no previous work has studied these bus encoding techniques using realistic traffic from SPEC CPU2000 benchmark programs; most of them have used random traffic patterns that do not behave like real-world instruction and data streams. In this context too, we use realistic technology parameters from the ITRS roadmap for current and future nanometer technology nodes.

Finally, we present a thermal model and a methodology to estimate the temperatures of individual wires of a global signal bus during dynamic simulation. Our model incorporates the effect of inter-layer heat transfer (heat conduction from the substrate and lower metal layers through the inter-layer dielectric) and intra-layer heat transfer between adjacent bus lines through the inter-metal dielectric. It can also estimate the temperature gradient between the sending and receiving ends of the bus and hence, it can be used to estimate any dynamic delay variations due to Joule heating. Our model can also be used to estimate the effect of varying substrate temperatures on wire self heating, although in this work, we assume a constant substrate temperature for simplicity. Specific results we obtained are listed next.

- We estimate from simulations using our model for 130 nm technology node that, during the time interval taken to commit one billion instructions in the pipeline, high performance bus wire temperatures rise by 10-37°C for various

SPEC CPU2000 benchmarks. This is solely due to Joule heat dissipated due to wire switching activities.

- In future 45 nm technology node, wire temperature rise for the same set of benchmarks and simulation sample was found to be between 20-58°C.
- We observed that instruction and data bus wires attained absolute temperature in the range 80.3-104°C and 97.6-123.7°C, in 130 nm and 45 nm processors, respectively, during the course of our simulation, showing that signal lines attain significant temperatures too.
- Significant wire temperature gradients of magnitude between 16-25°C were found to be most common between the sending and receiving ends of the wires during the course of simulation.
- Some significant correlation was found to exist between energy dissipation behavior and wire temperature rise in buses across time; short, intermittent cycles of high energy-dissipating switching activity trigger step changes in temperature.

3.3 Bus Line Energy Dissipation Model

In this section, we develop our bus line energy dissipation model that calculates energy dissipated as a result of a switching (both self and coupling) transition. This energy model is then used to determine change in wire temperature that occurs due to the combined effect of self-heating in the wire and heat conduction into the surrounding medium. Values for wire geometry (wire width, spacing, etc.) and technology and equivalent circuit parameters, like capacitance and resistance of a global line, that we used for various nanometer-scale technologies were listed in Table 2.2.

As described earlier, the energy drawn from the supply rails by the driving gates of a bus line is dissipated as I^2R losses in the bus line. This results in temperature rise in wires due to the self-heating effect. Existing bus energy models, like that in [85], only provide expressions for total energy dissipated in the bus. From the thermal design point of view, the energy dissipated in each bus line is important since it helps determine the temperature rise in each individual wire separately. This can be estimated using our model described below. First, we describe how the energy dissipated due to line self capacitance can be found; the procedure to estimate the contribution of repeaters to this self energy is also explained. Next, we explain how energy dissipated due to inter-wire coupling capacitances, including adjacent coupling and non-adjacent coupling capacitances, can be estimated.

3.3.1 Energy Dissipated due to Line Self Capacitance

Define $V_i = V_i^{fin} - V_i^{in}$, i.e., the difference between the final and initial voltages on line i. Note that V_i^{in} and V_i^{fin} can take either one of two values: 0 or V_{DD}. Thus, $V_i = V_{\mathrm{DD}}$ implies that the self capacitance of line i charges due to a rising transition ($0 \rightarrow 1$), whereas $V_i = -V_{\mathrm{DD}}$ means that it discharges due to a falling transition ($1 \rightarrow 0$). For each transition, energy that is dissipated in wire i due to charging or discharging of the self capacitance of the wire can be calculated as: $E_i^s = 0.5 \times (C_{line} + C_{rep}) \cdot {V_i}^2$, where C_{line} is the self capacitance of the wire and C_{rep} is the total capacitance of repeaters on the line. The energy E_i^s is called *self energy* since it involves only the self or line capacitance (including the contribution of repeaters).

Values for C_{line} are obtained by multiplying the per-unit length capacitances given in Table 2.2 with wire length and values for C_{rep} are computed using Equation 2.6.

3.3.2 Energy Dissipated due to Inter-Wire Capacitance

The second component of energy dissipation is *coupling energy*, which is influenced by the charging, discharging, or toggling of the coupling capacitance $c_{i,j}$ between two lines i and j. A *coupling charge transition* occurs between the two lines when $V_i = 0$ or $V_j = 0$, and $V_i + V_j = V_{\mathrm{DD}}$; 00 → 01, 00 → 10, 10 → 11, and 01 → 11 are the possible cases. A *coupling discharge transition* occurs when $V_i = 0$ or $V_j = 0$, and $V_i + V_j = -V_{\mathrm{DD}}$; 01 → 00, 10 → 00, 11 → 10, or 11 → 01 are the possible cases. A *coupling toggle transition* occurs when $V_i, V_j \neq 0$ and $V_i = -V_j$, i.e., when 01 → 10 or 10 → 01 transition occurs. In all three cases, the coupling energy dissipated in line i due to $c_{i,j}$ is obtained as: $E^c_{i,j} = 0.5 \times c_{i,j}(V_i{}^2 - V_i \cdot V_j), i \neq j$. Values of $c_{i,i\pm 1}$ are given in Table 2.2. It can be seen that the toggle case dissipates an equal amount of energy ($E^c_{i,j} = E^c_{j,i} = 2 \times c_{i,j} \cdot V_{\mathrm{DD}}{}^2$) in both coupled lines, but the charge and discharge transitions result in coupling energy dissipation equal to $0.5 \times c_{i,j} \cdot V_{DD}{}^2$ in the line that charges/discharges.

Thus, the total energy dissipated in a segment of the wire between two repeaters of bus line i is the sum of the self energy and coupling energies and is given by the following equation.

$$E_i = E^s_i + \sum_{\forall j, j \neq i} E^c_{i,j} \tag{3.1}$$

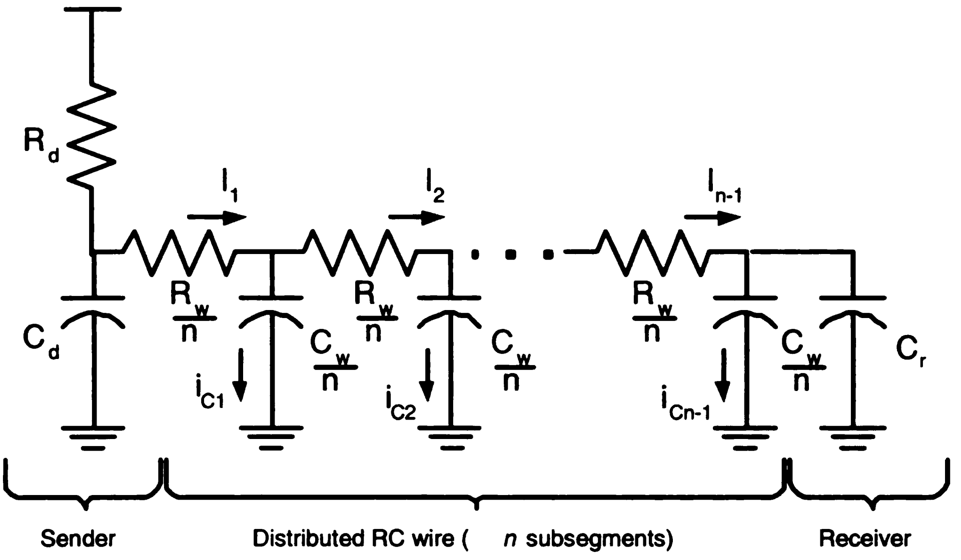


Figure 3.1. Distributed-RC model of the wire segment divided into n subsegments.

3.3.3 Distributed-RC Line Energy Model

This energy is dissipated non-uniformly across the length of the segment, as we show next. Consider the schematic of the segment of a distributed RC-wire shown in Figure 3.1. For this segment of length l_{opt}, the total wire electrical resistance R_w and the parasitic capacitance C_w which includes the self and coupling capacitance, can be divided equally across n *subsegments*. Thus, each subsegment has a resistance $\frac{R_w}{n}$ and capacitance $\frac{C_w}{n}$. The driving repeater is represented by its resistance R_d. At the end of the wire is the receiving repeater, contributing a gate capacitance C_r to the load. Let the energy dissipated in the k^{th} subsegment of wire i be represented by $E_{i,\,k}$. Consider the 4-stage RC network corresponding to shown in Figure 3.1; this represents a distributed RC line. For a unit input signal $v(t)$, the s-domain voltages at the four nodes will be:

$$
\begin{aligned}
v_1(s) &= V_{DD}(s^{-1} - m_0^1 + m_1^1 s + m_2^1 s^2 + \cdots) \\
v_2(s) &= V_{DD}(s^{-1} - m_0^2 + m_1^2 s + m_2^2 s^2 + \cdots) \\
v_3(s) &= V_{DD}(s^{-1} - m_0^3 + m_1^3 s + m_2^3 s^2 + \cdots) \\
v_4(s) &= V_{DD}(s^{-1} - m_0^4 + m_1^4 s + m_2^4 s^2 + \cdots),
\end{aligned}
$$

where m_0^i, m_1^i, m_2^i, etc. represent the first, second moment and so on. The corresponding currents through the capacitors are $i_{c1}(s) = Y_1(s) \cdot v_1(s)$, $i_{c2}(s) = Y_2(s) \cdot v_2(s)$, $i_{c3}(s) = Y_3(s) \cdot v_3(s)$, and $i_{c4}(s) = Y_4(s) \cdot v_4(s)$, where $Y_i(s) = sC_i$ is the admittance of each subsegment [89]. This gives:

$$
\begin{aligned}
i_{c1}(s) &= sC_1(s^{-1} - m_0^1 + m_1^1 s + m_2^1 s^2 + \cdots)V_{DD} \\
i_{c2}(s) &= sC_2(s^{-1} - m_0^2 + m_1^2 s + m_2^2 s^2 + \cdots)V_{DD} \\
i_{c3}(s) &= sC_3(s^{-1} - m_0^3 + m_1^3 s + m_2^3 s^2 + \cdots)V_{DD} \\
i_{c4}(s) &= sC_4(s^{-1} - m_0^4 + m_1^4 s + m_2^4 s^2 + \cdots)V_{DD}.
\end{aligned}
$$

From the circuit, it is clear that $I_1 = i_{c1} + i_{c2} + i_{c3} + i_{c4}$, $I_2 = i_{c2} + i_{c3} + i_{c4}$, $I_3 = i_{c3} + i_{c4}$, and $I_4 = i_{c4}$. In general, we can write the following equation for current through a resistor i after discarding higher order moments:

$$
I_i(s) = V_{DD} \sum_{j \in D_i} C_j \left(1 - s \sum_{j \in D_i} C_j m_0^j + s^2 \sum_{j \in D_i} C_j m_1^j \right), \qquad (3.2)
$$

where the set D_i represents all the downstream nodes of node i, V_j is the voltage at the j-th node, and $C_j = C_w/n$ is the capacitance of j-th subsegment. The downstream capacitance of node i which is the sum of the capacitance of subsegments i

through n can be evaluated as:

$$\sum_{j \in D_i} C_j = (n-i)\frac{C_w}{n}. \tag{3.3}$$

We can express the power series in Equation 3.2 in transfer function form with poles (p_1^i, p_2^i) and zeros (z_1^i, z_2^i). However, it has been shown in [90] that for interconnect lines, the transfer function using two-pole analysis has a special form in which the numerator polynomial is a constant as shown in the following equation.

$$H_i(s) \approx \frac{1}{(1 + b_1^i s + b_2^i s^2)}. \tag{3.4}$$

Expanding this transfer function about $s = 0$, we have $H_i(s) = 1 - b_1^i s + ((b_1^i)^2 - b_2^i)s^2$ [89]. Comparing with Equation 3.2, we get:

$$b_1^i = \sum_{j \in D_i} C_j m_0^j = (n-i)\frac{C_w}{n} \sum_{j \in D_i} t_{ED}^j, \tag{3.5}$$

since the Elmore delay t_{ED}^j of the line until the j-th subsegment is given by the first moment m_0^j [90]. Thus we have:

$$\begin{aligned} I_i(s) &= \frac{V_{DD}(n-i)\frac{C_w}{n}}{(1 + \frac{s}{p_1^i})(1 + \frac{s}{p_2^i})} \\ &= V_{DD}(n-i)\frac{C_w}{n} \frac{p_1^i p_2^i}{p_1^i p_2^i + (p_1^i + p_2^i)s + s^2} && (3.6) \\ I_i(t) &= \mathcal{L}^{-1}[I_i(s)] \\ &= V_{DD}(n-i)\frac{C_w}{n} \frac{p_1^i p_2^i}{p_1^i - p_2^i}(e^{-p_1^i t} - e^{-p_2^i t}), && (3.7) \end{aligned}$$

where $\mathcal{L}^{-1}[\cdot]$ is the inverse Laplace transform operator. In Equation 3.6, we have used the transfer function of the form:

$$G_i(s) = \frac{p_1^i p_2^i}{p_1^i p_2^i + (p_1^i + p_2^i)s + s^2}.$$

By equating $H_i(s)$ and $G_i(s)$, we obtain $b_1^i = \frac{p_1^i + p_2^i}{p_1^i p_2^i}$. Now the amount of Joule heat dissipated in the i-th subsegment can be estimated as:

$$\begin{aligned} E_i &= \int_0^\infty [I_i(t)]^2 \cdot \frac{R_w}{n} dt \\ &= \frac{(n-i)^2 C_w^2 V_{DD}^2 R_w}{n^3} \times \frac{p_1^i p_2^i}{2(p_1^i + p_2^i)} \\ &= \frac{(n-i)^2 C_w^2 V_{DD}^2 R_w}{n^3} \times \frac{1}{2b_1^i}. \end{aligned}$$

Substituting for b_i^i from Equation 3.5, and rearranging, we get:

$$E_i = \frac{\frac{C_w}{n} V_{DD}^2}{2} \times \frac{\frac{n-i}{n} R_w C_w}{\sum\limits_{j \in D_i} t_{ED}^j}. \tag{3.8}$$

We observe that the first term in Equation 3.8, i.e., $0.5 \times \frac{C_w}{n} V_{DD}^2$ corresponds to the Joule heat dissipated in a subsegment assuming that energy is dissipated uniformly across the wire length. The second term can be regarded as a *correction factor* indicating that the energy dissipation is non-uniform across the length, i.e., higher energy is dissipated at the subsegments near the sending then than those near the receiving end. This is because for increasing i ($0 \leq i \leq n-1$), the numerator reduces and the denominator increases in value and hence the correction factor reduces overall.

We validated our model by comparing with energy distribution obtained using the Cadence Spectre simulator for different number of subsegments (n = 10, 50, and 100). The normalized energy dissipated in each wire subsegment for the $n = 10$ case, obtained using our model and Cadence Spectre simulations with 130 nm ITRS parameters, is shown in Table. 3.1. The average error of our model is 4.53% and the maximum error is 7.75% compared to Spectre results. Note that this difference arises because the derivation of Equation 3.8 ignored higher-order moments of the node voltages. For n = 50, and 100 subsegments too, we found that energy values from our model are very close to those from Spectre; the average errors in these cases were 3.94% and 3.51% respectively. As a trade-off between model complexity, in terms of its simulation time and its accuracy, we use n = 10.

Sub-segment #	Normalized energy		%Error
	Equation 3.8	Spectre	
0	0.132565	0.123033	7.75
1	0.125624	0.117550	6.87
2	0.118420	0.112207	5.34
3	0.112894	0.107001	5.50
4	0.106988	0.101931	4.96
5	0.101150	0.096996	4.28
6	0.095827	0.092194	3.94
7	0.089974	0.087525	2.79
8	0.084548	0.082986	1.88
9	0.080009	0.078578	1.82
Average			4.53

Table 3.1. Comparison of normalized energy dissipated in wire subsegments obtained using our model and Cadence Spectre simulations for 10 subsegments.

3.4 Thermal Model

In this section, we present our thermal model. This enhanced model can also estimate the distribution of wire temperatures across the length of the wire segment, compared to our earlier model [81]. Next, we briefly introduce chip thermal structures and discuss the heat transfer mechanism in modern chip packages.

3.4.1 Chip Thermal Structures and Heat Transfer

Figure 3.2 shows the cross sectional view of various layers in a chip package that influence the way heat is transferred away from the active areas. The figure shows a C4/CBGA (flip-chip) package with an attached heat sink and no forced air cooling. For this type of packaging and cooling system, it has been found that there are two heat transfer paths: a primary path that conducts away heat generated at the active layer (substrate) through the heat spreader, attach material, and the heat sink, and a secondary path that transfers heat from the substrate through the dielectric layers—heat flows from the bottommost to the topmost interconnect layer—and finally flows through C4 bumps, ceramic substrate, CBGA joints, and the printed circuit board to the ambient air [66]. As mentioned earlier, models for estimating substrate temperatures are available in tools like HotSpot [64, 66] but detailed activity-dependent models for estimating global signal wire temperatures are not. Next, we present the model that will help estimate spatially-distributed wire temperatures in a wire segment.

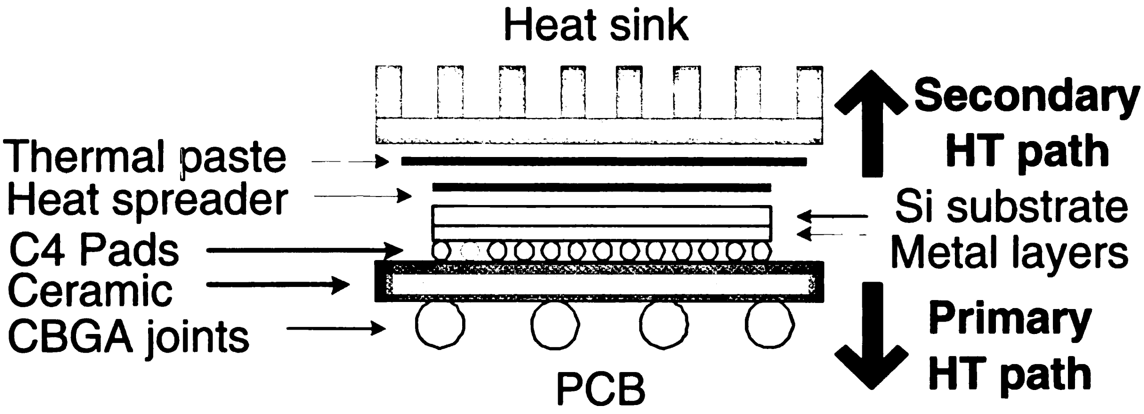


Figure 3.2. Figure shows the view of different thermal structures of a C4/CBGA chip and the primary and secondary heat transfer paths.

3.4.2 Detailed Thermal Model

In the thermal model presented next, we consider any subsegment k as a point source of Joule heat, called a *thermal node.* Using the well-known analogy between thermal and electrical quantities, we can consider that, the temperature difference between two nodes, corresponds to a voltage difference and the heat transfer rate to current. The ability of the wire segment to hold heat is modeled by its *thermal capacitance* and the ability of the surrounding dielectric to conduct heat away from the wire segment is modeled as the *thermal resistance.* These thermal circuit parameters are brought together to form a thermal-RC network, shown in Figure 3.3(a) for a 5-wire bus, across the same subsegment k in all wires.

By equating the rate of heat flowing into a node in the thermal equivalent circuit to the rate of heat flowing out (analogous to Kirchoff's current law in electrical circuits), we obtain the following.

For the two edge wires:

$$P_{i,\,k} + P'_{i,\,k} = \mathcal{C}_{i,\,k} \cdot \frac{d\theta_{i,\,k}}{dt} + \frac{(\theta_{i,\,k} - \theta_0)}{\mathcal{R}_{i,\,k}} + \frac{(\theta_{i,\,k} - \theta_{i\,\pm\,1,\,k})}{\mathcal{R}_{inter}} \tag{3.9}$$

and for the middle wires:

$$P_{i,\,k} + P'_{i,\,k} = \mathcal{C}_{i,\,k} \cdot \frac{d\theta_{i,\,k}}{dt} + \frac{(\theta_{i,\,k} - \theta_0)}{\mathcal{R}_{i.k}} + \frac{(2\theta_{i,\,k} - \theta_{i\,-\,1,\,k} - \theta_{i\,+\,1,\,k})}{\mathcal{R}_{inter}}, \tag{3.10}$$

where $P_{i,\,k}$ is the instantaneous power dissipated in the k^{th} subsegment of the i^{th} wire, $P'_{i,\,k}$ is the equivalent power due to the effect of switching activity in lower metal layers and the substrate, and θ_0 is the ambient temperature (45 °C or 318.15 K) inside the computer box. Note that these equations do not include heat that may potentially flow through the vias. The reason for neglecting the via effect is given

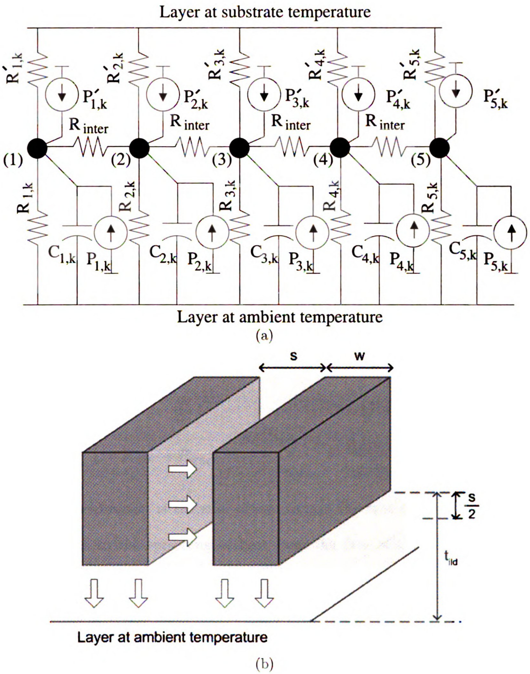


Figure 3.3. Thermal model. (a) Complete equivalent thermal-RC network for a 5-wire bus. $P'_{1,\,k} = P'_{2,\,k} = \ldots = P'_{5,\,k}$, $R_{1,\,k} = R_{2,\,k} = \ldots = R_{5,\,k}$, $C_{1,\,k} = C_{2,\,k} = \ldots = C_{5,\,k}$, and $P_{1,\,k}, P_{2,\,k}, \ldots, P_{5,\,k}$ are bus-activity dependent in the model shown. (b) Geometry for calculating equivalent thermal resistances for a wire based on previous work of Chiang et al. The lightly shaded regions and arrows represent heat flow between the conductors or between layers (from a hotter to a cooler one).

in Section 3.4.2. The instantaneous or cycle-by-cycle power $P_{i,k}$ can be obtained by dividing the energy $E_{i,k}$ obtained using Equation 3.8 by the clock cycle time. However, in our microarchitectural simulations, we record the energy $E_{i,k}$ for a finite interval and then divide it by the duration of the time interval to obtain the power dissipated. This time duration is set as explained later in Section 3.5.3.

In the above equations, $\mathcal{C}_{i,k}$, the thermal capacitance of the wire segment, is given by: $\mathcal{C}_{i,k} = \mathcal{C}_S \cdot (t_i \cdot w_i)$, where $\mathcal{C}_s$ is the specific heat per unit volume of the wire metal, and w_i and t_i are wire dimensions as shown in Figure 3.3(b) and with values given in Table 2.2. $\mathcal{R}_{i,k}$ is the thermal resistance of the wire segment along the heat transfer path as shown in Figure 3.3(b) and it can be calculated from the following expression using wire geometry and thermal conductivity k_{ild} of the inter-layer dielectric (ILD) as described in [83]:

$$\mathcal{R}_{i,k} = \mathcal{R}_{spr} + \mathcal{R}_{rect} = \frac{\ln(\frac{w_i + s_i}{w_i})}{2 \cdot k_{ild}} + \frac{t_{ild} - 0.5s_i}{k_{ild}(w_i + s_i)}. \tag{3.11}$$

The above expression is the sum of two terms: the first is the spreading resistance $\mathcal{R}_{spr}$ due to the spreading of heat from the face of the wire exposed to a cooler layer (away from the substrate) in a trapezoidal manner, and the second is the thermal resistance $\mathcal{R}_{rect}$ due to rectangular heat flow as depicted in Figure 3.3(b). Equations 3.9 and 3.10 can be solved to determine the wire temperature $\theta_{i,k}$.

Heat transfer from lower layers through the dielectric

Next we consider the temperature rise in global signal lines due to heat transfer from underlying layers. This is needed because, in current C4/CBGA packages, a secondary heat transfer path exists from the substrate through the interconnect layers.

Thus, some heat flows from the substrate through the metal layers—bottommost to the topmost interconnect—and finally flows through C4 bumps, ceramic substrate, CBGA joints, and the printed circuit board to the ambient air [66]. The temperature increase due to this effect to each global wire can be estimated using the following closed-form expression [83]:

$$\Delta\theta = \sum_{i=1}^{N-1} \frac{t_{ild,i}}{k_{ild,i} s_i \alpha_i} \cdot [\sum_{j=i}^{N-1} (jmax)^2 \rho_j \alpha_j t_j], \tag{3.12}$$

where N is the number of layers of metal and ρ_j is the resistivity of the metal line (Copper). The values for $t_{ild,i}$, $k_{ild,i}$, s_i, and t_i, corresponding to different layers of metal, were obtained from the ITRS roadmap.

Note that Equation 3.12 neglects the thermal capacitance of wire segments in the lower layers. This because wires at lower layers are usually thinner and shorter (smaller w and t) and also have smaller lengths. Thus, $\mathcal{R}_{inter}$, which depends on $t \cdot l$, and $\mathcal{C}_{th}$, which depends on $w \cdot t \cdot l$, both have negligible values, and the dominant $\mathcal{R}_{th}$ terms are only considered in this equation. The above equation also assumes that all wiring tracks underneath the global bus are populated with power supply wires that carry current at their maximum density ($jmax$).

The net effect of the secondary heat transfer path (from the substrate and lower metal layers) is depicted as the constant current source $P'_{i,k}$ in the network shown in Figure 3.3(a). Note that the $P'_{i,k}$s are all equal since spatial variation in substrate temperature across the width of the bus is neglected. This is valid, because in almost all cases, the area footprint of the buses we study is well within the dimensions of the underlying circuit block for which we know the substrate temperature.

Heat transfer from lower layers through vias

Joule heat generated in the lower metal layers can flow to the global metal layer through the ILD (as described in the previous subsubsection) and also, in parallel, through the vias. However, heat transfer through vias occurs only within the range of the thermal characteristic length L_H of the wire [37,83]:

$$L_H = \sqrt{\frac{t_i \cdot t_{ild} \cdot k_m}{k_{ild}(1 + 0.88\frac{t_{ild}}{w_m})}}, \tag{3.13}$$

where $k_m = 401\text{W/mK}$ is the thermal conductivity of Copper metal. If a wire is longer than L_H, the via heat transfer is negligible. Using parameters in Table 2.2, L_H was found to be 10.56 μm for 130 nm and 10.33 μm for 45 nm, which are much smaller compared to our inter-repeater segment length l_{opt}. Hence, the heat transfer through vias will always be negligible in the global buses we consider.

Lateral thermal coupling between wires

The lateral heat transfer between adjacent wires can be a significant amount due to the large exposed sidewall area in high aspect-ratio global lines and due to the difference in activity rates of the neighboring lines (which creates a temperature difference and hence lateral heat flow). It has been shown using FEM simulations that thermal coupling is a significant phenomenon in global lines, particularly when high activity wires are placed next to low activity ones [73]. In our model, this effect is captured with a *lateral inter-wire thermal resistance* whose value depends on wire geometry parameters, as shown in Figure 3.3(a), and the inter-metal dielectric (IMD) thermal conductivity, k_{imd}, and is given by the expression:

$$\mathcal{R}_{inter} = l_{opt} \times \frac{s_i}{k_{imd} t_i}. \tag{3.14}$$

Previous work on interconnect thermal modeling did not consider the effect of inter-wire heat transfer [66]; our model incorporates this for better accuracy. For simplicity, we assume that the ILD and IMD are the same material. Hence $k_{imd} = k_{ild}$.

Thus, the temperature $\theta_{i,\,k}$ of the k-th subsegment of wire i is affected by the rate of heat $P_{i,\,k}$ generated in it as a result of activity-dependent current flow, the thermal capacitance $\mathcal{C}_i$ of the wire metal, thermal resistances of surrounding inter-layer and intra-layer dielectric $\mathcal{R}_i$ and $\mathcal{R}_{inter}$, respectively, and the temperature $\theta_{i\,\pm\,1,\,k}$ of the k-th subsegments of its adjacent wires, all of which are considered in our model. A distribution of wire temperatures across the wire length can be obtained by solving Eqs. 3.9 and 3.10 for a number of subsegments $k = 0, 1, \ldots, n$. The *temperature gradient* $\Delta\theta_i$ or difference between the sending and receiving end temperatures can be estimated using: $\Delta\theta_i = \theta_{i,\,0} - \theta_{i,\,n}$, where n is the number of subsegments.

3.4.3 Steady-State Thermal Model

The detailed thermal model discussed above is used to track activity-dependent temperature variations in bus wires across time. However, due to its complexity, it is somewhat difficult to use in the temperature optimization methodologies that we propose later in our research. Hence we develop an approximate version of this model, known as the *steady-state thermal model.* This model is also used to estimate the initial temperatures for the bus wires before starting detailed thermal simulations.

The steady-state model for three wires is discussed next. Consider three consecutive wires w_i, w_j, and w_k on a bus. When there is no bit reordering, data bits b_i, b_j, and b_k are carried on these lines. Let the corresponding power dissipation on these wires be P_i, P_j, and P_{tsk}, respectively. We assume a steady state temperature model for thermal analysis of this wire set. In this model, the final temperature T^{fin} of a structure with initial temperature T^{ini} is: $T^{fin} = T^{ini} + P \times R_t$, where P is the power dissipated by the structure and R_t is its thermal resistance. Thermal resistances of global signal wires can be estimated based on their geometry using equations given in [66,81] and wire power dissipation can be obtained using a microarchitecture-level simulator. For three adjoining wires, the steady state thermal equivalent circuit is shown in Figure 3.4.

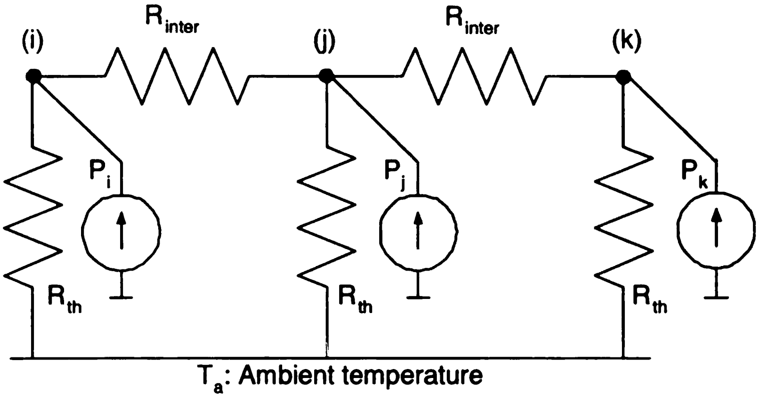


Figure 3.4. Steady state thermal equivalent circuit for three wires. Heat transfer between wires is modeled by R_{inter} and heat loss to surroundings by R_{th}. P_i represents power dissipated in each wire due to switching activity and it can found using a microarchitecture-level simulator.

Using Kirchoff's law on the three nodes, we get the following equations:

$$\begin{aligned} P_i &= \frac{T_i - T_a}{R_{th}} + \frac{T_i - T_j}{R_{inter}}, \\ P_j &= \frac{T_j - T_a}{R_{th}} - \frac{T_i - T_j}{R_{inter}} - \frac{T_k - T_j}{R_{inter}}, \\ P_k &= \frac{T_k - T_a}{R_{th}} + \frac{T_k - T_j}{R_{inter}}, \end{aligned}$$

In these equations, R_{th} is the inter-layer thermal resistance, R_{inter} the intra-layer thermal resistance, and T_a is the ambient temperature, assumed to be 45°C inside the computer box. Solving this set of simultaneous equations using *Mathematica*, the expression for the temperature of the middle wire is found to be:

$$T_j = (P_i + P_k) \cdot \alpha + P_j \cdot (\alpha + \beta) + T_a, \tag{3.15}$$

$$\text{where } \alpha = \frac{R_{th}^2}{3R_{th} + R_{inter}} \text{ and } \beta = \frac{R_{th} R_{inter}}{3R_{th} + R_{inter}}. \tag{3.16}$$

Thus, we find that the temperature rise ($\Delta T_j = T_j - T_a$) in the middle wire is proportional to a weighted sum of the power dissipated in itself and in its neighboring wires.

3.5 Simulation Environment and Methodology

We used the Alpha 21264 platform for this work. Details of the simulation infrastructure for this platform were described earlier in Chapter 2.4.4.

3.5.1 Benchmarks and Sample Sizes

Previous work on temperature-aware microarchitecture design has characterized benchmarks, mostly in the SPECint suite, as *hot, medium,* or *cold* benchmarks based on the percentage number of cycles that they are in violation of a 81.8°C threshold [64]. From the benchmarks used in that work, we chose three benchmarks that were reported to result in extreme thermal stress (*gcc, crafty,* and *vortex*), and two from the medium (*gzip* and *mesa*) thermal stress group. We randomly chose seven benchmarks, that have not been characterized previously, to complete the 12 benchmarks in our set. Thus, our workload represent a mix of benchmarks that have been shown to result in severe to moderate thermal violations (those listed above) and those which operate well below the threshold of 81.8°C. Hence, with this workload, we can also analyze the extent to which high silicon die temperatures and thermal stress, which [64] studied, correlate with global interconnect temperatures.

We collected energy and temperature results for a simulation sample of one billion committed instructions after a fast-forward phase of five billion instructions that skips over the program startup phase. We did not use techniques like SimPoint [77] to choose representative samples because our thermal simulations needed a single, large sampling window covering possibly, multiple phases of benchmark execution, and to capture the effects of idling of processor units and buses that provide dynamic opportunities for wire temperatures to cool down.

3.5.2 Thermal Warmup and Initial Temperatures

As reported in earlier work, it is computationally impractical to simulate long enough for the heat sink temperature to reach steady state, since its thermal RC time constant is significantly larger than that of any on-chip structure [64, 91]. Hence, we followed the methodology suggested in [64] to obtain accurate results from our thermal simulations. First, we used the Wattch power/performance simulator to obtain average power consumption values for various on-chip structures [58]. Then, we fed these values to the HotSpot tool to obtain the steady state heat sink temperature, and used this value to initialize the heat sink when running our simulations. Also, to avoid "cold start" effects during the initial period of our wire temperature simulation, we ran all simulations using our wire model twice. In the first pass, we obtained an approximate steady state temperature value for each wire by estimating the power dissipated in each wire for one billion cycles using the model discussed in Section 3.4.3.

We initialized the temperature of each wire of our target bus using its steady state temperature (Equation 3.15) and performed the temperature simulation as described in the next subsection. Note that, using this approach, the initial temperatures of the bus wires will not be the necessarily equal since it will depend on the distribution of energy across the wires.

3.5.3 Granularity of Thermal Simulation

After the fast-forward phase which skips through the unrepresentative initial section of the benchmark program, wire temperatures were set to the steady state temperatures estimated as described in the previous subsection. Then, for the next one billion instructions—our simulation window—we recorded energy and temperature results every 100K cycles. For thermal simulations, the energy dissipated per wire was divided by the time taken for each window ($f_{clk} \times 10^5$), and a fourth-order Runge-Kutta (RK4) method was used to solve the differential equations for the thermal-RC network (Eqs. 3.9 and 3.10) to obtain the individual wire temperatures at the end of the interval. The RK4 simulation loop, which was implemented using the method described in [92], iterates for a number of times which depends on the interval size (100K cycles) and the thermal RC time constant of the wire. This ensures that each RK4 simulation advances the solution by a small enough time interval *dt* that is substantially less than the thermal RC time constant. In this way, each step of the temperature simulation will yield sufficiently accurate temperature estimates without the rigor of cycle-by-cycle simulation which will require huge computation time and memory resources.

Using experimentation, we found that setting the value of *dt* to three (130 nm) and two (45 nm) gave the best tradeoff between simulation time and the nature of temperature characteristics we obtained. For example, with the clock frequency in the 130 nm process (1.68 GHz), time taken by the processor to execute 100K cycles is $t_{window} = 59.52$ μs and the thermal RC time constant of the wire, calculated using wire geometry parameters in Table 2.2, is $t_{RC} = 3.6171$ μs. For these values, the RK4 simulation should iterate $dt \times \frac{t_{window}}{t_{RC}} = 3 \times \frac{59.52}{3.6171} \approx 50$ times to ensure the best granularity of temperature simulation.

3.6 Experiments and Results

In this section, we present results from simulations using our bus-line energy dissipation and thermal models and discuss their implications.

3.6.1 Energy Dissipation in Processor Buses

In this subsection we show that, in addition to adjacent wire coupling capacitances, energy dissipated in switching transitions between non-adjacent wires also affects bus energy dissipation significantly for current and future technologies. It is a well-known fact that, in global signal lines, the wire-aspect ratio—the ratio of wire thickness to wire width—is increasing faster than wire-spacing ratio, the ratio of inter-wire spacing to inter-layer spacing. This causes the sidewall (inter-wire) coupling capacitance to dominate the area capacitance. In sub-100 nanometer bus lines, the reduced inter-wire distance further causes increased fringing effects with adjacent as well as non-adjacent neighbors of a wire. With capacitance values we extracted using FastCap for the 130 nm technology node—values are given in Table 2.2—and using our model from Section 3.3 to estimate the coupling energy dissipation in each line, we found that the energy dissipation is underestimated by up to 7.8% in data buses and 7.6% in instruction buses, when non-adjacent coupling capacitances are neglected, for data bus traffic in the nine benchmarks we analyzed. Results for this experiment are shown in Figures 3.5 and 3.6. Also, we found that, although the non-adjacent coupling capacitance values are decreasing with technology scaling, this energy estimation error remains more or less constant in future technologies. Thus, we conclude that accurate bus energy dissipation models must consider the influence of non-adjacent coupling capacitances also. Previous work did not consider the effect of non-adjacent coupling capacitances and its influence on energy; ours is the first to do so.

Non-adjacent coupling capacitances are especially important to consider when evaluating the benefits of microarchitectural techniques for low-power buses. In current literature, only schemes that aim to reduce energy dissipation due self and adjacent inter-wire coupling transitions exist. Such schemes can potentially increase the relative contribution of energy dissipated in transitions involving non-adjacent coupling capacitances.

Effectiveness of Low-Power Bus Encoding Schemes

We evaluated the effectiveness of some popular bus encoding schemes like bus-invert (BI) [51], odd/even bus-invert (OEBI) [53], and coupling-driven bus-invert (CBI) [54] on wide data and instruction buses. To our knowledge, this is the first study to report energy dissipation results for microprocessor buses using SPEC benchmarks that represent real-world programs; most previous studies, including the ones cited above, reported energies for random traffic patterns. Additionally, we also implemented a

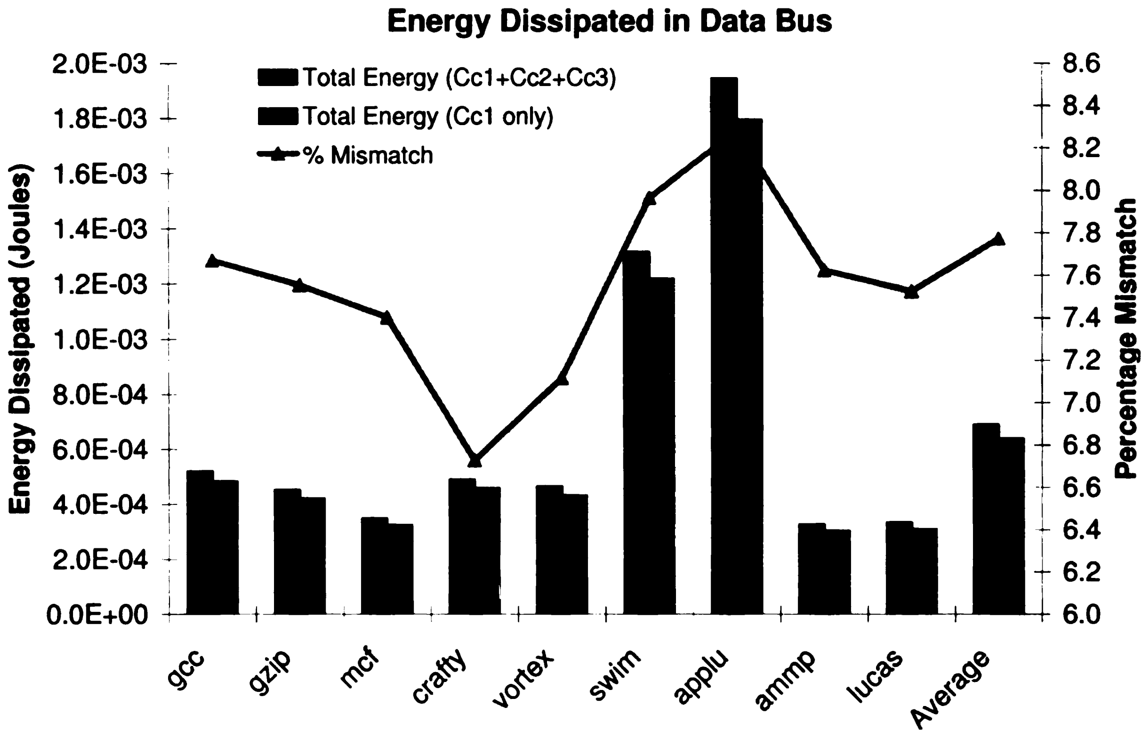


Figure 3.5. Total energy dissipated in a 64-bit data bus for various benchmarks. 'Ccl only' represents the existing energy models which consider only self and adjacent coupling capacitances. 'Ccl+Cc2+Cc3' represents our model that considers self capacitances, adjacent coupling capacitances (Ccl), and two non-adjacent capacitances (Cc2 and Cc3) on each side. The % energy mismatch shown by the line is plotted with respect to the right-hand side Y-axis.

variant of the BI scheme called *segmented bus invert* where the bus is divided into four groups and BI encoding is applied to each group separately. This arrangement requires four extra invert lines that are placed in the four higher order bit positions. In our experiments, BI was implemented with the one invert line at the MSB position—we found this to result in less energy dissipation compared to the case when the invert line is at the LSB position—and CBI was implemented with the invert line in the LSB position as mentioned in [54]. OEBI was implemented with two invert lines (LSB as the odd-invert line and MSB as the even-invert line) as described in [53].

The total bus energy dissipated for unencoded and encoded data is shown in

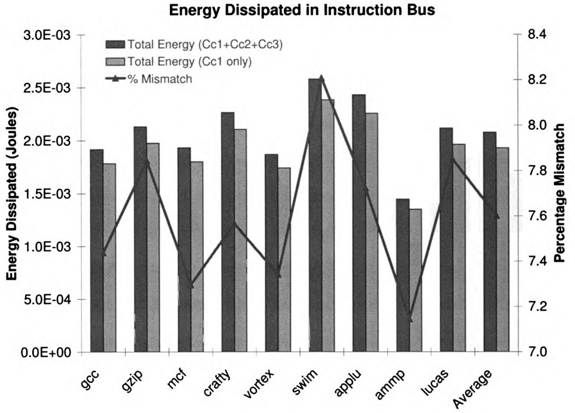


Figure 3.6. Total energy dissipated in a 128-bit instruction bus for various benchmarks. The % energy mismatch shown by the line is plotted with respect to the right-hand side Y-axis.

Figure 3.7. The energy values reported in this plot have been averaged across the nine benchmarks with each benchmark being simulated for 500 million committed instructions. From the results shown for existing bus models (Cc1 only), we find that all four encoding schemes reduce self energy, with segmented BI being the best. Coupling charge/discharge energy dissipation increases marginally, with BI and CBI encoding but reduce somewhat when OEBI encoding is used. Here too, segmented BI shows the best reductions. The amount of energy dissipated due to toggle transitions decreases when any of the four encoding schemes are used, with segmented BI again giving the best results followed by OEBI, BI, and CBI in that order. A significant observation from these results is that existing coupling-aware encoding schemes (like

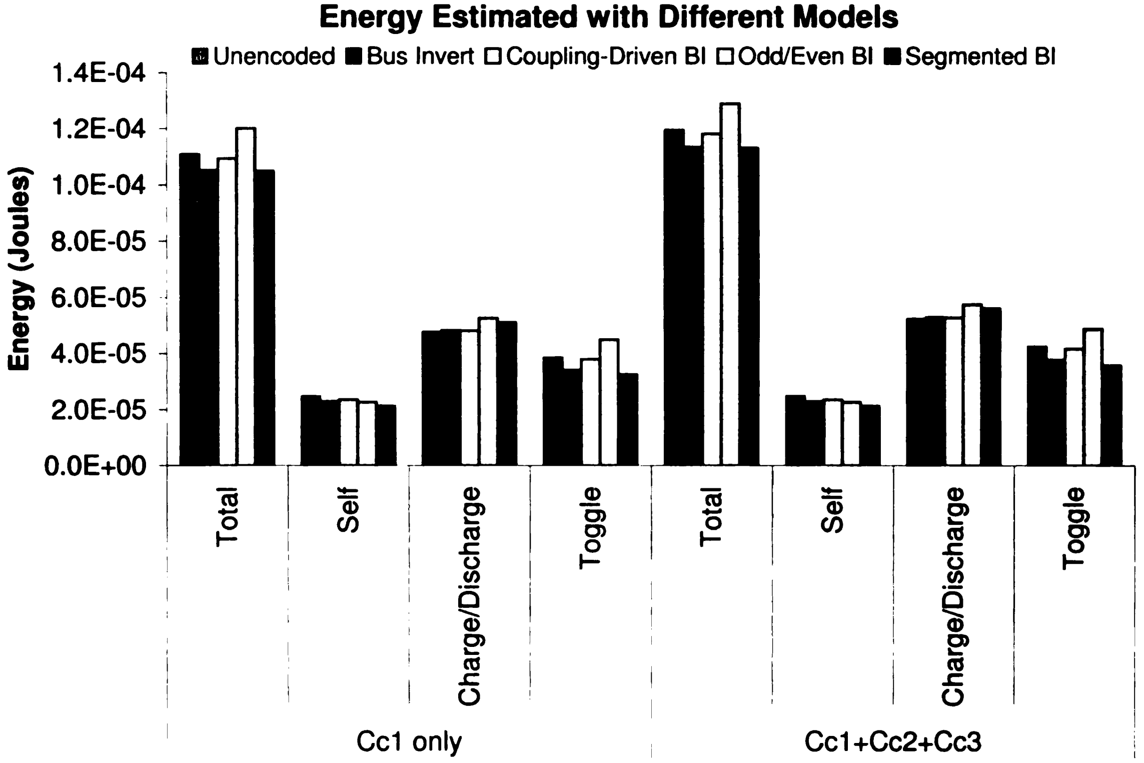


Figure 3.7. Total energy dissipated in a 64-bit data bus with various encoding schemes. 'Self' denotes self energy, 'C/D' denotes the coupling charge/discharge energy and 'Toggle' denotes the coupling toggle energy dissipation. 'Cc1 only' refers to existing energy models that consider self and adjacent coupling capacitance only and 'Cc1+Cc2+Cc3' refers to our energy model that considers self, adjacent coupling, and two non-adjacent coupling capacitances.

CBI and OEBI) have limited impact for wide data buses. Furthermore, we observed that the average number of bit transitions between consecutive cycles was very low (much less than half the bus width) for the SPEC benchmarks we analyzed. This is most likely the result of the higher order 32 bits of data not being utilized. Hence the number of inversions was small, even for CBI and OEBI, and hence most of the time, data was being transmitted in original (unencoded) form. Segmented BI performed the best in these situations because, as the effective bus width for each segment was smaller, the number of cycles during which data-inversions took place was greater. Thus, overall, while segmented BI encoding resulted in lowest energy dissipation, OEBI and BI were almost similar in impact, while CBI was significantly worse. Note that none of the coupling-aware schemes we examined yielded improvements on the order of what had been reported earlier—36% for OEBI and 30% for CBI with respect to unencoded random data—for these schemes [53, 54].

When our energy model (considering $Cc1$, $Cc2$, and $Cc3$) was used, all coupling (charge, discharge, and toggle) energies increased and the trend in charge/discharge energies remained unaffected. For toggle energies, however, we observed that OEBI performed significantly worse than others. This is clearly the effect of toggles on coupling capacitances between non-adjacent wire pairs. The net effect of this is that, with our new bus energy model, OEBI and CBI both perform significantly worse than BI and segmented BI. Based on our results, we can conclude that bus-inversion based encoding schemes do not work well for wide buses and for realistic data streams (from SPEC benchmark programs) where the number of bits that transition between consecutive cycles is low.

Impact on Wire Temperature Distribution

The influence of energy dissipation due to non-adjacent coupling capacitances on wire temperature can be illustrated with a simple example of a 5-wire bus like the one shown in Figure 2.3. Consider transitions on the five bus lines, from the most significant bit (MSB) line to the least significant bit (LSB) line as follows: $\uparrow\uparrow\downarrow\uparrow\uparrow$. The notation $\uparrow$ indicates that, in the current cycle, the line charges to V_{DD} from its previous ground state and $\downarrow$ indicates that the line discharges in the current cycle from V_{DD} held in the previous cycle. This set of transitions represents the relative thermal worst-case since most of the energy dissipation is concentrated in the center line. Numbering the bus lines from 0 (MSB) to 4 (LSB) and noting that all inter-wire transitions, if any, are toggles, the coupling energy dissipated in each line estimated using our energy model, described earlier in Section 3.3, can be written as follows:

$$\begin{aligned}
E_0^c = c_{0,2}V_{DD}^2 &= Cc2 \cdot V_{DD}^2 \\
E_1^c = c_{1,2}V_{DD}^2 &= Cc1 \cdot V_{DD}^2 \\
E_2^c = (c_{0,2} + c_{1,2} + c_{2,3} + c_{2,4})V_{DD}^2 &= 2(Cc1 + Cc2)V_{DD}^2 \\
E_3^c = c_{2,3}V_{DD}^2 &= Cc1 \cdot V_{DD}^2 \\
E_4^c = c_{2,4}V_{DD}^2 &= Cc2 \cdot V_{DD}^2
\end{aligned}$$

where $c_{i,j}$ represents the coupling capacitance between wire i and j. Note that the self energy dissipated in all five wires is the same $(\frac{1}{2}(C_w + C_{rep})V_{DD}^2)$ and hence its contributes equally to temperature rise in all five wires. The energy dissipated in the middle wire E_2^c is the highest even if Cc2 is neglected and hence, this wire is likely to have the maximum temperature. Furthermore, if non-adjacent coupling capacitances are non-negligible, the middle wire dissipates much higher energy and its temperature is likely to be even higher.

3.6.2 Correlation between Energy and Temperature

In this subsection, we examine the correlation between energy and temperature characteristics obtained using our model. We report and analyze time-varying energy and temperature profiles for only one benchmark—*gcc*, for a simulation interval of 10 billion cycles in the 130 nm technology node. We found that other benchmarks exhibited similar behavior; hence these are not reported. The energy and temperature profiles are shown in Figure 3.8. In this figure, energy and temperature, plotted on the y-axes, have been averaged across the number of bus lines. The temperature plot clearly shows that the average wire temperature continues to rise with time although the rate of change is not linear; the trend line shown on the plot is only a very coarse approximation. But, the results are significant because they show that the average wire temperature increases by about 10 degrees over six seconds of execution of a typical program like *gcc* on a 130 nm microprocessor. We also observe that short, intermittent cycles of high switching activity can trigger changes in temperature, evidenced by the regions marked 1 and 2 on the plot. Also, we notice that such bursts of energy dissipation—likely caused by increased bus utilization—cause the temperature rise to 'linger' for a short period of time as shown by the step-like changes at the beginning of regions 1 and 2.

3.6.3 Final and Peak Wire Temperatures

In this subsection, we present results obtained from simulations using our thermal model. During our simulations, we recorded type types of temperature information: (1) the temperature change in each wire between the start and end of simulation, (2) the highest temperature reached by each wire during the simulation, and (3) the temperature gradient of each wire between its sending and receiving ends. These results are presented next.

We observed that wire temperatures increased significantly over the time interval of simulation for most wires. Figures 3.9 through 3.11 show the wire temperature rise that we recorded for three integer and three floating-point programs respectively, each for one billion committed instructions of execution for all bits of the 64-bit data bus. The corresponding results for the 128-bit instruction bus are in Figures 3.11

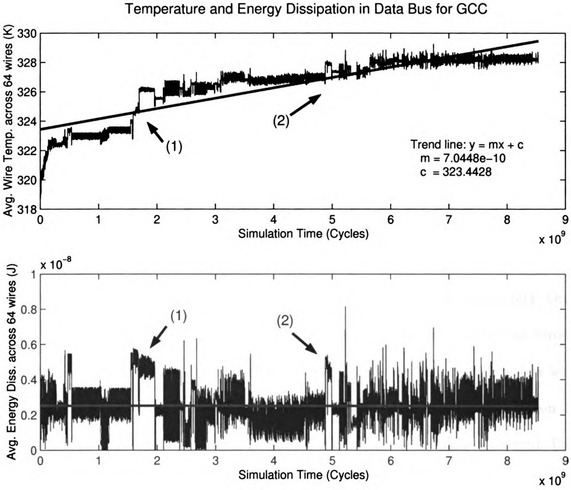
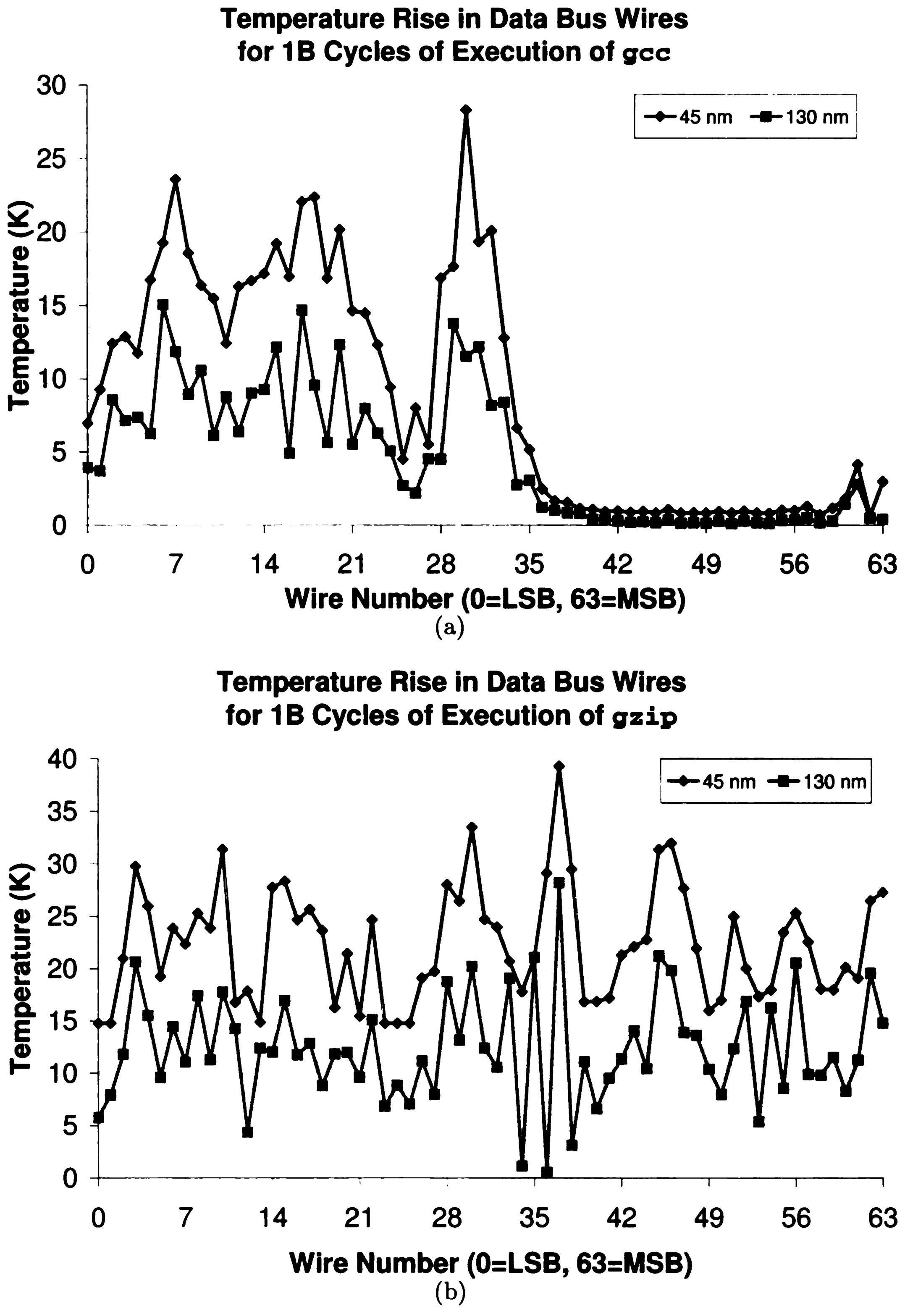


Figure 3.8. This plot shows average energy dissipation and wire temperature of the bus for a simulation interval of 10 billion cycles. The continuing temperature rise can be clearly observed.

through 3.14. We show detailed results for only these six benchmarks since they exhibit interesting behavior. The highest temperature rise recorded for any wire during our simulation, for the 12 benchmarks we analyzed is given in Table 3.2. We show results for both 130 nm and 45 nm technologies in the figures and in the table. The time taken to commit a billion instructions in the pipeline which is typically on the order of a few seconds is much longer than the thermal RC time constant of the wire, which is only a few microseconds. Thus our simulation interval is large enough to allow temperatures to settle to their characteristic values. Furthermore, we initialized wire and heat sink temperatures to their steady state values as described earlier in Section 3.5.2, to prevent cold-start effects.

From the knowledge of characteristics of instruction and data traffic, all lines in an instruction bus, which is 128 bits wide (fetch-width= 4 instructions), can be considered equally active, while in a load/store data bus, which is 64 bits wide, the lower order 32-bits are expected to be most active due to data value locality. The results shown in Figures 3.9–3.14 reflect these observations to some extent. For integer data, we observe that the hottest wires are the ones that carry lower-order bits. One notable exception is *gzip* in which all wires show significant temperature rise across the simulation. This is expected because, when executing *gzip*, the data bus will carry primarily 8-bit characters packed in the 64-bit bus. Another observation is that, for *mcf*, the middle wire is the hottest at the end of the simulation interval. For floating-point benchmarks, temperature rise is somewhat evenly distributed across the 64 bits because the higher-order wires, which carry the exponent bits, are also quite active. Also, *lucas* shows higher temperatures in some lower order bits. Finally, we notice that the highly active wires are likely to end up at higher temperatures when executing integer workloads as against floating-point workloads.

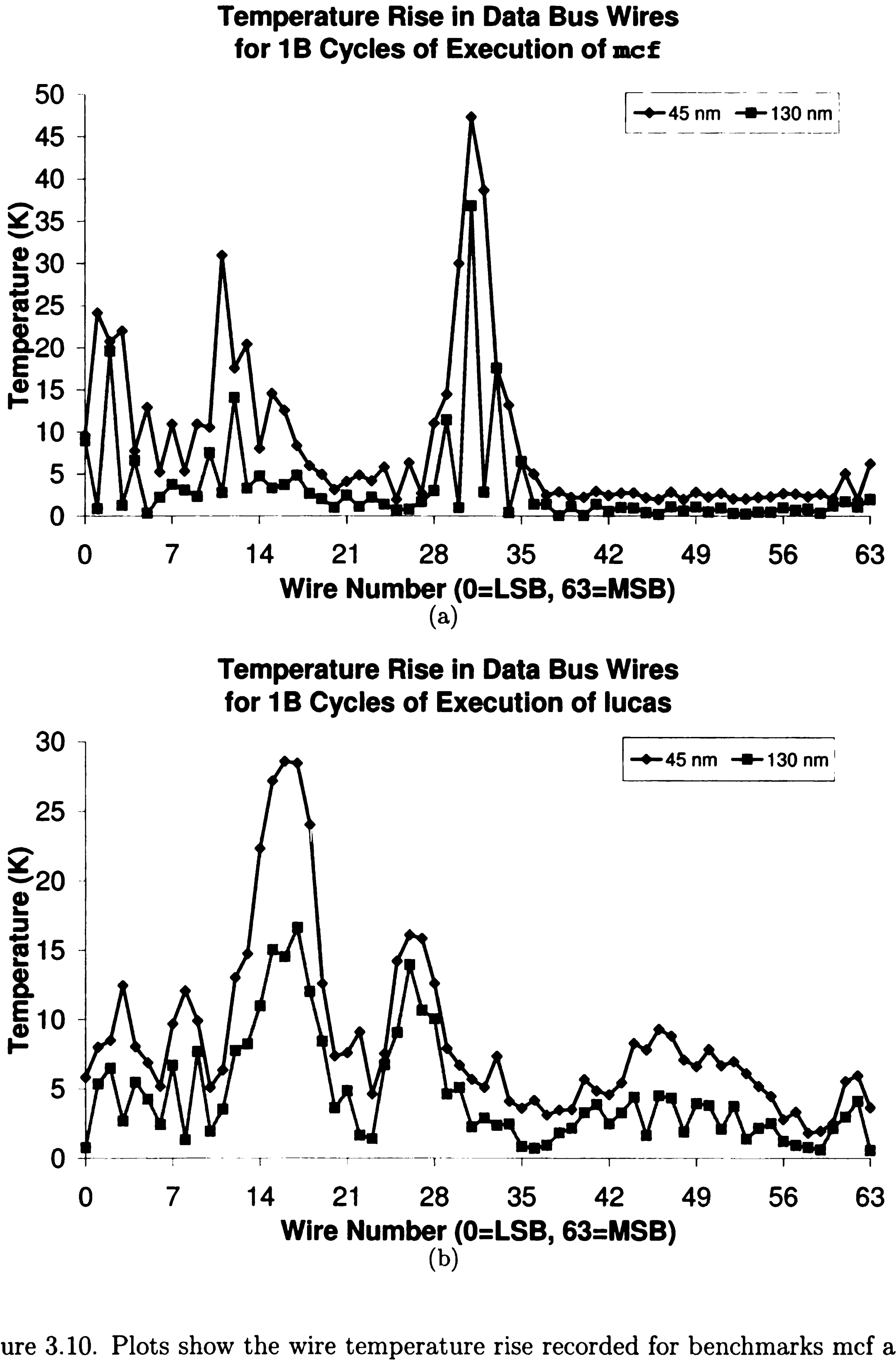
During the course of simulation, we observed that wire temperatures rose and fell as bus activity, the number of transitions, and the energy dissipation varied. A three-dimensional plot showing the variation across time and across the lower-order 32 bits of the data bus, plotted for three billion cycles of execution of the *gcc* benchmark is shown in Figure 3.15. This plot shows that there are intervals during which wire temperatures rise to higher values due to a sudden rise in energy dissipation and then



(a)

(b)

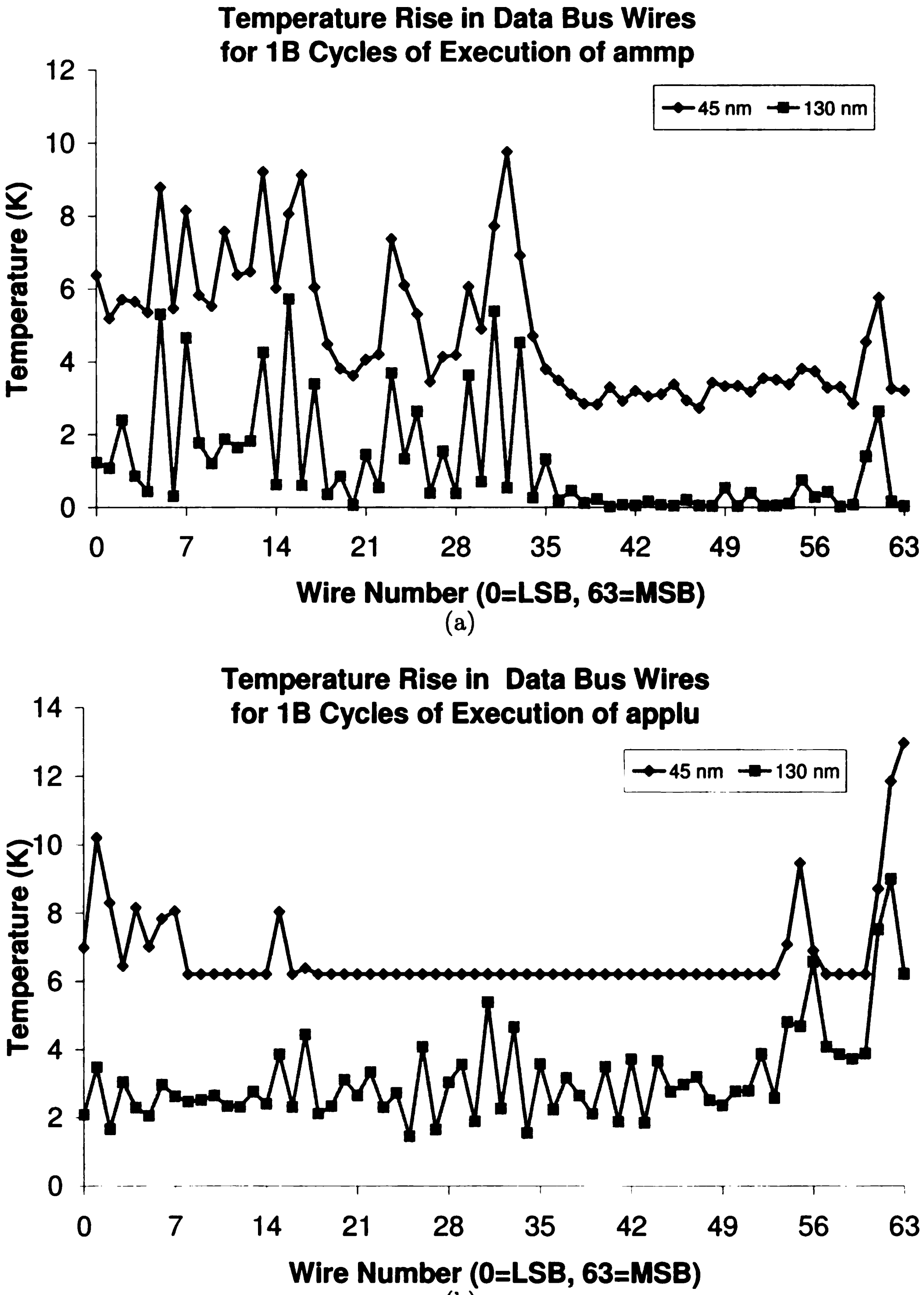
Figure 3.9. Plots show the wire temperature rise recorded for benchmarks gcc and gzip for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.



(a)

(b)

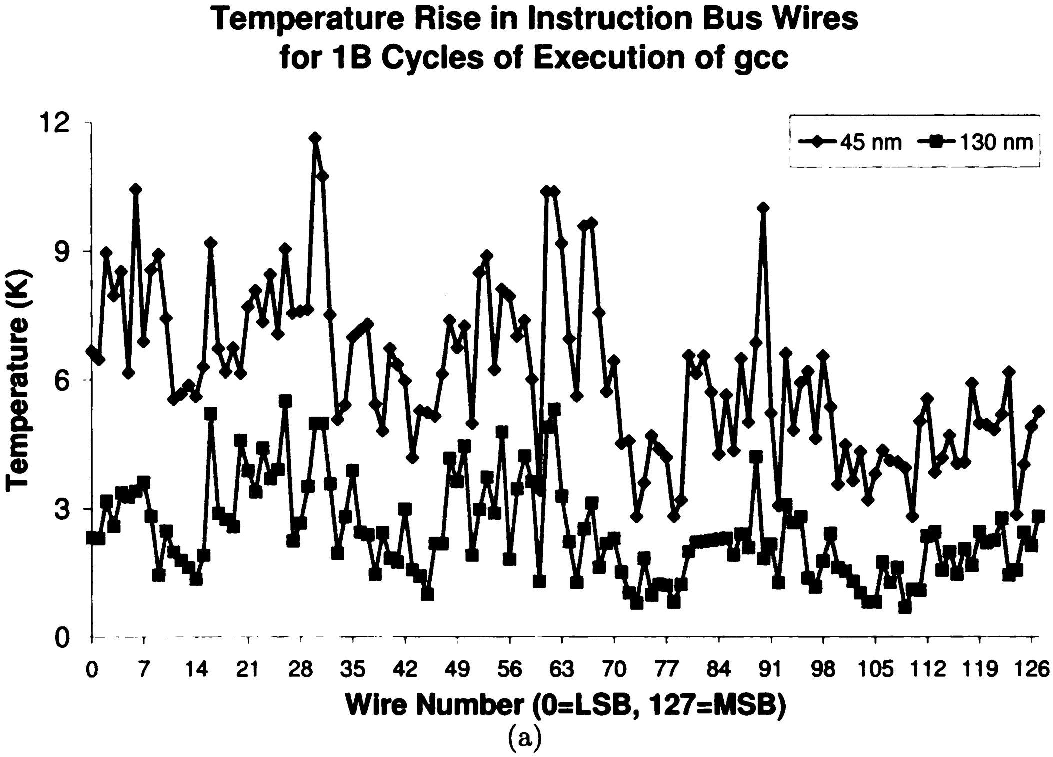
Figure 3.10. Plots show the wire temperature rise recorded for benchmarks mcf and lucas for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.



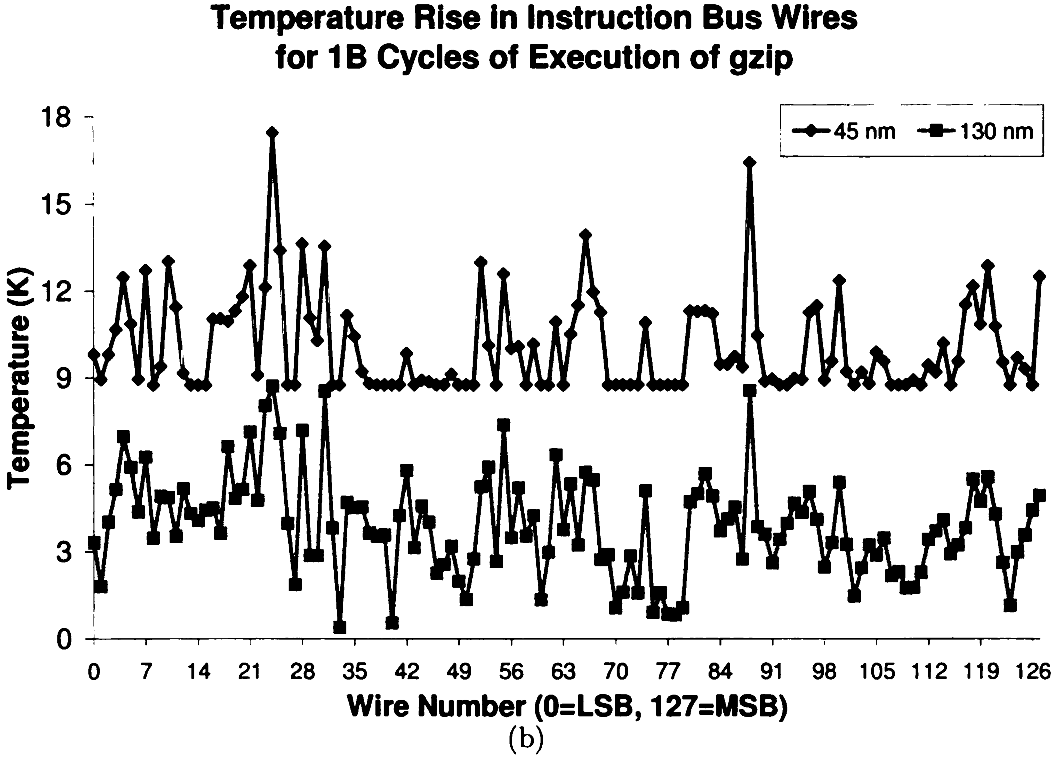
(a)

(b)

Figure 3.11. Plots show the wire temperature rise recorded for benchmarks ammp and applu for the data bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.

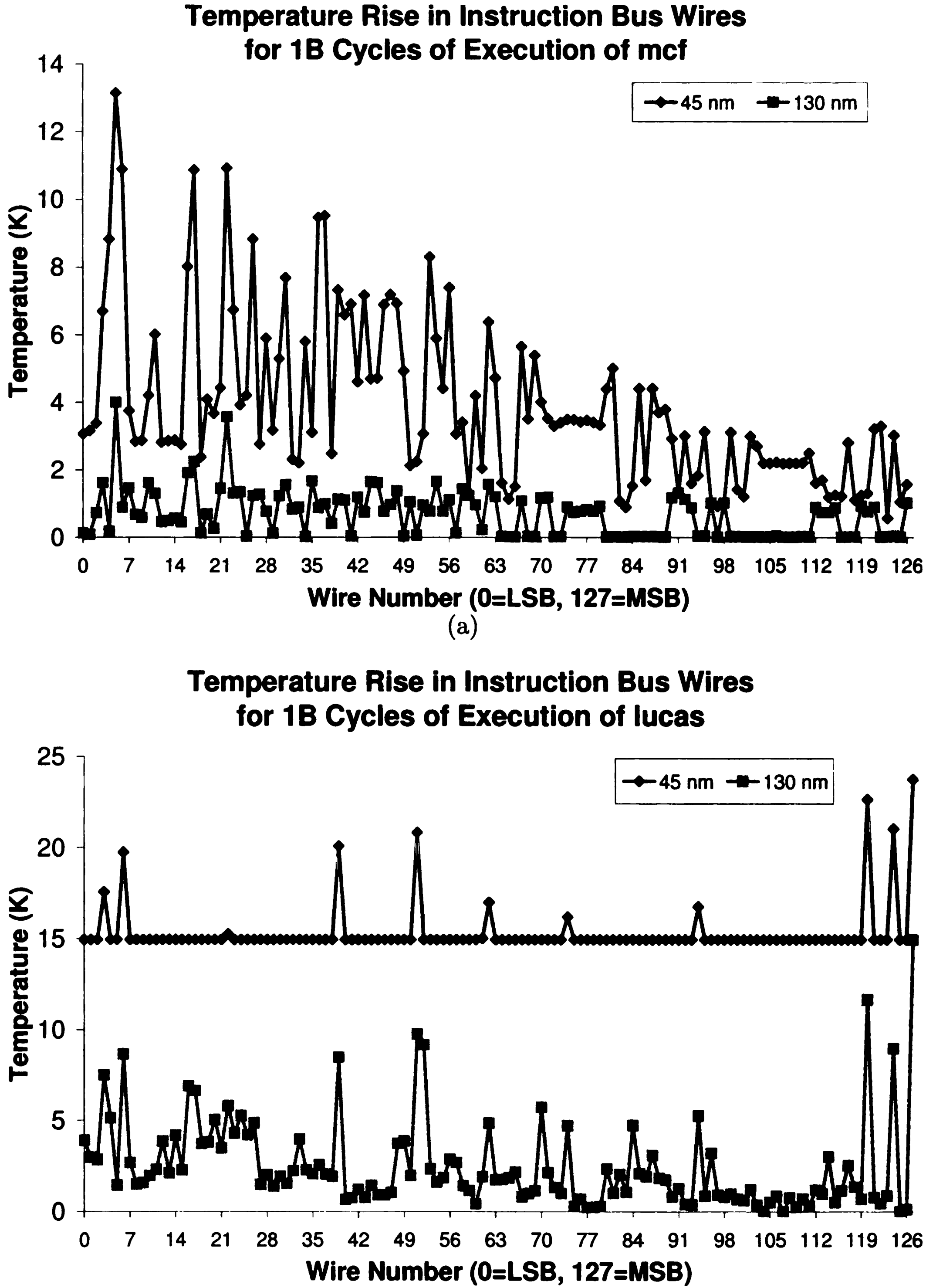


(a)



(b)

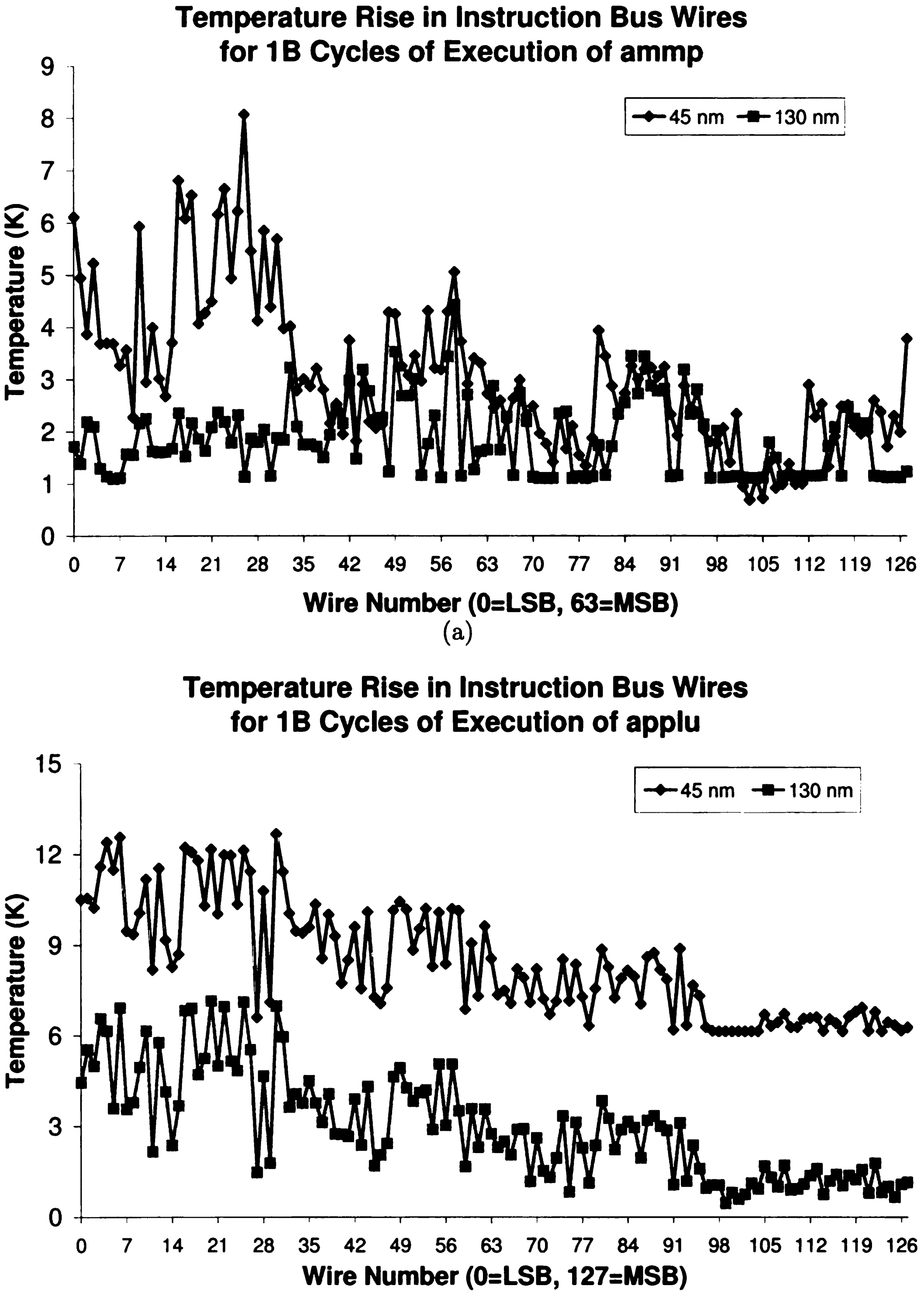
Figure 3.12. Plots show the wire temperature rise recorded for integer benchmarks gcc and gzip for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.



(a)

(b)

Figure 3.13. Plots show the wire temperature rise recorded for integer benchmarks mcf and lucas for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.



(a)

(b)

Figure 3.14. Plots show the wire temperature rise recorded for integer benchmarks ammp and applu for the instruction bus in 130 nm and 45 nm technology nodes over a simulation interval of one billion committed instructions for each benchmark.

settle at lower values. Such intervals neither occur synchronously across wires nor are uniformly distributed among them.

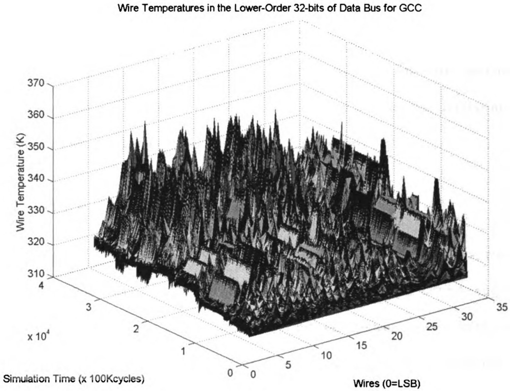


Figure 3.15. A three-dimensional plot showing spatial and temporal variations in wire temperature for the lower-order 32 bits of the load/store data bus for the *gcc* benchmark.

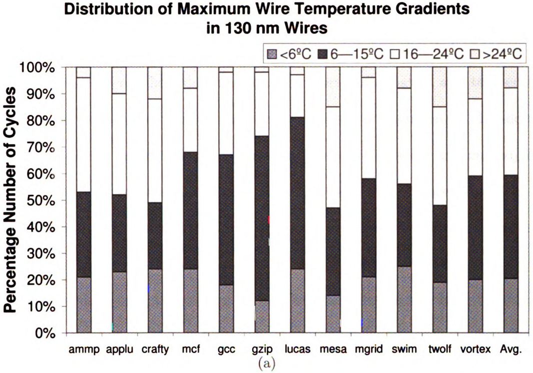
Table 3.2 lists the absolute maximum temperatures attained by any wire during the course of simulation. As can be seen, we found that wire temperatures may reach up to 104°C for data bus wires and 89.6°C for the instruction bus in the 130 nm technology node. For the 45 nm node, data bus wire temperature was found to go as high as 128.7°C and instruction bus wire temperature as high as 104.9°C. Note that these values are higher than 100° which is the maximum temperature assumed during interconnect design. We also observed that maximum temperature trends for data buses are very similar to those observed earlier for temperature rise. That is, the largest temperature change over the simulation interval occurs for bus wires whose transient temperature also touches maximum value, showing that different data bus wires experience varying amount of thermal stress depending on their location. For instruction buses, the maximum temperatures observed across bus wires were more or less similar. Hence, all instruction bus wires experience more or less similar amounts of transient thermal stress during the simulation interval.

3.6.4 Wire Temperature Gradients

Next, in Figures 3.16(a) and (b), we show the frequency distribution of the wire temperature gradients that we recorded during our simulations, for 130 nm and 45 nm load/store data bus wires. These plots show that, on the average across the benchmarks we analyzed, the temperature gradient in this bus can be expected to be between 6–15°C for 130 nm technology. For 45 nm technology temperature gradients between 16–34°C were most commonly observed for the same set of benchmarks and simulation sample. During our simulations, the maximum temperature gradient observed was 31°C for 130 nm and 42°C for 45 nm simulations. These wire temperature gradients are the result of two factors: (1) the non-uniform dissipation of Joule heat along the wire length which is modeled using Equation 3.8, and (2) due to the difference in temperature of the underlying substrate blocks which was obtained using HotSpot and applied during our thermal simulation. Temperature gradients across the length of the wire also affect delay. It has been reported that for a 1 mm long wire with the driver in the hot region and receivers in a cooler region, a temperature difference of 10°C results in a 5 ps ($\approx 8\%$) additional delay at the receiver [93].

		ammp	applu	crafty	mcf	gcc	gzip	lucas	mesa	mgrid	swim	twolf	vortex
130 nm	D-Bus	96.43	102.74	109.09	104.01	94.58	99.65	94.53	102.12	99.20	93.56	94.37	95.38
	I-Bus	83.50	85.81	89.59	85.70	83.00	83.89	83.59	86.76	86.14	81.36	82.39	80.34
45 nm	D-Bus	118.58	120.12	118.62	123.76	111.62	109.81	106.72	128.70	113.76	105.59	110.03	116.59
	I-bus	98.73	100.67	104.92	102.37	98.51	101.20	103.57	103.39	104.23	99.06	101.92	97.57

Table 3.2. Maximum wire temperatures in °C recorded during a simulation of one billion committed instructions for data and instruction buses using 130 nm and 45 nm parameters.



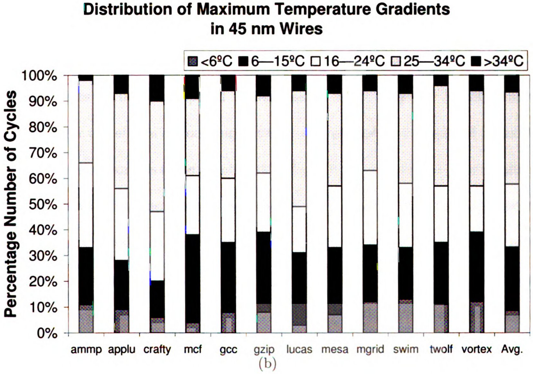


Figure 3.16. Frequency distribution of maximum wire temperature gradients for 130 nm and 45 nm processor wires.

3.7 Summary

In this chapter, we presented a unified nanometer-scale bus energy dissipation and thermal model that can help designers monitor energy dissipation and temperature change in individual wires during trace- or execution-driven simulation. In addition to self capacitance, our model incorporates the effects of adjacent and non-adjacent capacitive coupling on bus energy dissipation, the effect of repeater insertion, the effect of lateral heat transfer between adjacent wires, and the effect of inter-layer heat transfer. Unlike existing models which provide estimates for total bus energy, our model can estimate energy dissipated in each bus line; this feature helps to estimate wire temperatures also. Using this integrated model in a first-of-its-kind study, we studied energy and thermal characteristics of instruction and data buses using an execution-driven simulation of a billion or more instructions of nine SPEC CPU2000 benchmarks. We found that existing bus energy models provide estimates that are about 7-8% less accurate compared to our energy model. This is because they do not account for the effects of coupling between non-adjacent wire pairs of a bus. Our model, which incorporates these effects, is the first of its kind to do so. Our results also show that, in wide instruction and data buses used in modern processors executing SPEC CPU2000 workloads, existing bus encoding schemes show no significant energy benefit due to the nature of data traffic. When non-adjacent coupling effects between wire pairs are considered, energy dissipation savings reduce considerably. Based on simulations using our thermal model, we found that average wire temperatures in data and instruction buses may rise 10-37 °C during a simulation run of only a billion cycles in a 130 nm spuerscalar processor executing SPEC CPU 2000 benchmark programs.

This temperature rise is primarily due to heat generation as a result of currents flowing in the wire during bit switching. Changes in substrate temperature may cause other effects in the temperature profile which we did not explore in this work.

In a future 45 nm technology node, wire temperature rise for the same set of benchmarks and simulation sample was found to be between 20-58°C. We observed that instruction and data bus wires attained absolute temperature in the range 80.3-104°C and 97.6-123.7°C, in 130 nm and 45 nm processors, respectively, during the course of our simulation, showing that signal lines attain significant temperatures too. Significant wire temperature gradients of magnitude between 16-25°C were found to be most common between the sending and receiving ends of the wires during the course of simulation. Notable correlation was found to exist between energy dissipation behavior and wire temperature rise in buses across time; short, intermittent cycles of high energy-dissipating switching activity trigger step changes in temperature.

The impact of these results, especially, the highly fluctuating—both in time and space—energy and temperature profiles of instruction and data buses that we observed, is the following. Since the energy dissipation of the wire roughly represents the square of the time-varying current, fluctuations in the energy mean that a highly varying load is being placed on the power supply network by the driving circuits through which the currents flowing in the wires are drawn. This varying load can cause inductive voltage drops or $L\frac{di}{dt}$ noise. This motivates the need to smoothen temporal variations in energy dissipation of wires with appropriate techniques. Also, the substantial disparity in wire temperatures across the bus motivates schemes that, based on information from interconnect thermal sensors, can migrate bus transmissions dynamically to cooler wires.

CHAPTER 4

DATA- AND TEMPERATURE-DEPENDENT DELAY VARIABILITY MODEL

4.1 Introduction

Rising wire temperatures are becoming an important issue in high-performance bus design, especially in current and future nanometer technology nodes, as the previous chapter showed. Higher temperatures adversely impact wire delays—due to the temperature dependence of metal resistivity—causing timing violations when the end-to-end propagation delay exceeds the designed value. The factors that influence the dynamic propagation delay of a signal transmitted on the wire can be classified into two types, *intrinsic* factors that are related to the switching activity of the wire and/or its neighbors and *extrinsic* factors like process and voltage variations. As shown in the earlier chapters, the temperature distribution along the wire is a function of the switching activity in the wire and hence it is also an intrinsic factor.

In the context of global interconnect lines, temperature variations occur due to two reasons that are both important to study. First, energy is dissipated in a non-uniform manner across the length of the wire. In Chapter 3.4, we showed that the temperature at the sending end of a wire will be higher than that of the receiving end of the wire. In this chapter, we develop a model to estimate the impact of this temperature gradient on the propagation delay of the signal. Substrate temperature gradients, when present, will exacerbate thermal gradient-dependent delay. Second, temperature variations are also non-uniform across time since the characteristics of programs dictate the amount of switching activity in signal wires and consequently, the energy dissipated in them and their temperature. When switching activities rise, it also causes the wire temperature gradient to increase.

Due to lack of detailed models, existing early-stage design exploration methods lump the effects of process, voltage, and temperature (PVT) variations. This results in overly pessimistic and/or incorrect delay estimates. Even in later stages of the design process, a constant temperature value across the chip is assumed to analyze of the electrical characteristics of devices and interconnects. In reality, given that on-chip power dissipation in devices as well as interconnect is workload-dependent, the temperature distribution within the chip is far from uniform, and thus the constant-temperature assumption will result in a design which will result in problems during validation and necessitate costly re-spins. Using detailed temperature models developed previously, this chapter examines the impact of data- and temperature-dependent delay variations for various on-chip high performance processor buses.

The organization of the rest of the chapter is as follows. In Section 4.2, we discuss related work. Following that, in Section 4.3, we describe our models. Then, we present results and discuss them in Section 4.4. Finally, we summarize in Section 4.5.

4.2 Related Work and Our Contributions

This section reviews related work. The impact of increasing interconnect temperatures has been well studied in [21, 38, 69]. However, they do not use real data from benchmark programs and hence their estimates are somewhat pessimistic. Also, these models are not amenable to use in microarchitecture-level exploration tools. Recent interest in temperature- and reliability-aware microarchitectures has led to the development of tools [64,66,94] and techniques [58,71] for processor thermal and reliability management. However, these tools do not address an important temperature-related reliability issue in on-chip interconnects: transient faults or timing violations due to temperature-dependent resistivity changes. In contrast to these, we seek to develop activity-dependent models that estimate the distribution of Joule heat across the length of a wire, the wire temperature gradient across it, and finally, the actual delay due to crosstalk and temperature-induced resistivity changes. Using these models, we analyze the number of delay violations occurring for different benchmark programs from the SPEC CPU2000 suite in 130 nm and 45 nm processor designs.

In current design methodologies, temperature-related wire reliability problems are identified late in the design cycle and hence their rectification involves substantial cost and effort. But this overhead can be avoided by properly accounting for temperature-related effects in early stage design. To our knowledge, no early stage microarchitecture exploration tool currently offers the capability of estimating temperature-induced timing violations in high-performance buses; our work is likely the first of its kind to develop such a model. Our model can also be used in temperature-aware delay and skew analysis in clock trees, although we do not examine this aspect in this paper. Specific contributions and key results from this paper are outlined next.

- Using a cycle-accurate microarchitectural simulator, we show that timing violations due to temperature gradients are somewhat likely in 130 nm designs—average of 2.27 per hundred bus references for an ALU result bus across ten

SPEC CPU2K programs—and increases in the future 45 nm technology node to 6.20 per hundred for the same processor design.

- We found that, by an optimistic analysis, the performance impact of overcoming temperature induced timing violations by re-transmitting data will be about 4% in a superscalar design at 130 nm and 11.92% at 45 nm.

- We also found that conventional techniques like bus encoding that seek to reduce energy dissipation and potentially wire temperatures have limited impact on alleviating temperature-induced timing violations. Reducing the bus clock frequency yielded a better impact, reducing average error rate to 1.07 in the 130 nm processor compared to encoding which reduced it to only 1.93 per hundred references.

4.3 Temperature Dependent Delay Variability Model

In this section, we present analytical models for estimating the spatial distribution of Joule heat, temperature, and temperature-dependent delay in RC interconnects. Versions of the well-known energy model for a lumped-RC wire, discussed in Chapter 2.1.2, has been traditionally used in interconnect analysis to estimate energy dissipated due to self and coupling transitions. But this model assumes that Joule heat is dissipated uniformly across the length of a wire and hence leads to conservative temperature estimate for the wire. Furthermore, it does not capture the spatial distribution of Joule heat, without which the impact of temperature on delay cannot be estimated accurately. In Chapter 3.3.3, we derived a new expression for energy distributed along the length of a wire and validated it using circuit simulation. We also constructed a thermal model and found wire temperature gradients using this model. The effect of this temperature gradient on wire delay is found as discussed next.

4.3.1 Wire Delay Considering Temperature Impact

The propagation delay of a lumped-RC wire considering only data dependent crosstalk was presented in Chapter 2.1.5. For a distributed RC line partitioned into n subsegments each of length l, the Elmore delay D of a signal passing through the line is the following:

$$D = R_d \cdot (C_r + \int_0^L c_0(x)dx) + \int_0^L r_0(x) \cdot (\int_x^L c_0(\tau)d\tau + C_r)dx, \qquad (4.1)$$

where $c_0(x)$ and $r_0(x)$ are the per-unit length wire capacitance and resistance, respectively, R_d is the driver resistance, and C_r is the receiver capacitance. Since the resistance of a wire segment changes with temperature, we can write: $r_0(x) = \rho_0(1+\beta \cdot T(x))$, where $T(x)$ represents the temperature profile along the length of the wire, ρ_0 is the unit length resistance at a reference temperature (273 K), and β is the temperature co-efficient of resistance for Copper ($\beta = 3.9e{-3}$ per°C). Substituting in Equation 4.1, we get:

$$D = D_0 + (c_0 L + C_r)\rho_0\beta \int_0^L T(x)dx - c_0\rho_0\beta \int_0^L xT(x)dx, \qquad (4.2)$$

$$\text{where } D_0 = R_d(C_r + c_0 L) + (c_0\rho_0\frac{L^2}{2} + \rho_0 L C_r), \qquad (4.3)$$

D_0 is the Elmore delay corresponding to a unit length resistance at reference temperature. In Equation 4.3, $\int_0^L T(x)dx$ represents the area under the temperature curve, denoted as A in a plot of temperature vs. wire-length. Let $T(x)$ be a straight line with $T(x = 0) = T_A$ and $T(x = L) = T_B$, $T_A \geq T_B$. The area under $T(x)$ gives the value of A. Now the x-coordinate of the centroid of this region is given by $x_C = \frac{1}{A}\int_0^L xT(x)dx$ [95]. Thus $\int_0^L xT(x)dx = x_C \times A$. Note that both x_C and A can be found easily using geometry, if $T(x)$ is assumed linear.

Thus, by estimating $T_A = \theta_{i,0}$ and $T_B = \theta_{i,n}$ using the model in Chapter 3.4, and the area under the temperature curve for any given sampling window, we can estimate the actual delay which includes the effect of temperature-dependent resistance. Using this, we can determine if a timing violation has occurred as described next.

4.3.2 Wire Delay Variability Considering Crosstalk and Temperature

During early stage design exploration, the designer's aim is to ensure that the microarchitecture meets all its performance expectations at the target clock frequency. The target frequency itself is decided based on knowledge of typical operating conditions that determine parameters like temperature, etc., and knowledge of process variations that are used to account for deviations from expected values. Based on estimates available from prior work, we assume that the delay can increase by up to 20% due to back end of line (BEOL) process variations and an additional 10% due to voltage drop and temperature variability [96,97]. Thus, we assume a guard band of 30% for the delay of a global wire due to PVT. Hence $t_{bus\_clk} = 1.3 \times D$.

We described earlier in Chapter 2.1.5 the procedure to estimate the worst-case data dependent delay (Equation 2.4) and estimate the safe clock frequency at which the bus can be operated. From that discussion, we note that not all bus references will trigger the worst case for delay in a bus line, resulting in varying amounts of delay slack across lines and also across time. As such, the actual delay for a line estimated using Equation 4.3 depends on the wire temperature gradient and the nature of its crosstalk with its neighboring wires. If the neighbors both switch oppositely with respect to the line, the delay will be t_{WC} and, if the temperature gradient is sufficiently high, the actual delay may exceed t_{bus_clk}. This is a timing violation. Note that, when this occurs, the temperature impact on delay overwhelms the 30% guard band that we have allocated to account for worst case PVT variations.

Given the current and previous data to be transmitted on the bus, we do the following to determine if a temperature-induced timing violation has occurred for the bus as a whole, in our cycle-accurate simulator. First, for each wire in the bus that changed state from the previous cycle, we compute the delay slack by examining coupling transitions with respect to its neighbors and determining its nominal delay $t_{p,\,k}$. Then, depending on the Joule heat dissipated and the thermal gradient across its length, we determine its actual delay using Equation 4.3. Finally, we consider a temperature-induced timing violation to have occurred for the bus as a whole if the actual delay in any of the lines, exceeded t_{bus_clk}. We report the number of such violations per hundred bus references in our results.

4.4 Results and Discussion

We study the delay variability of the 64-bit result bus that runs over the integer and floating-point execution units of the processor. This bus was chosen since it is highly capacitive and dissipates a substantial amount of energy in the processor core [45,58]. Also, it is routed over the execution unit consisting of ALUs and register files that are highly active; hence, the substrate temperature under the result bus will also be significantly higher than in other units. The result bus is also on the critical path and will be impacted most by any temperature-dependent timing violations, which may require retransmission of the data to maintain correct program execution.

4.4.1 Maximum Wire Temperatures and Gradients

The maximum wire temperatures that we recorded during the simulation of the result bus is shown in Table 4.1 for 130 nm and 45 nm technology nodes. It can be seen that the wire temperature can be as high as 103°C in a 130 nm processor and about 117°C in a 45 nm processor. Note that the design temperature for global wires was assumed to be 100°C but significantly higher temperatures were observed during our simulation. As mentioned earlier, higher wire temperatures increase wire delays by about 5% for every 20°C rise in temperature [38].

Next, in Figure 4.1, we show the distribution of the maximum wire temperature gradient that we determined using our model in Chapter 3.4. This plot shows that, on the average, the maximum temperature gradient in a wire can be expected to be between 16 and 24 degrees. These temperature gradients across the length of the wire also affect delay. It has been reported that for a 1 mm long wire with the driver in the hot region and receivers in a cooler region, a temperature difference of 10°C results in a 5 ps ($\approx 8\%$) additional delay at the receiver [93].

Having shown that significant wire temperatures and gradients occur when executing the benchmark programs in our workload, we examine next whether these

Maximum Wire Temperatures										
	gcc	gzip	bzip2	crafty	eon	twolf	art	mesa	mgrid	swim
130 nm	376.11	369.62	371.45	369.26	371.42	369.34	375.45	374.86	368.88	364.78
45 nm	382.08	386.78	375.12	382.38	389.87	375.70	382.71	378.87	376.27	379.06

Table 4.1. Maximum wire temperatures recorded for the ALU result bus. Ambient temperature is 318.15 K.

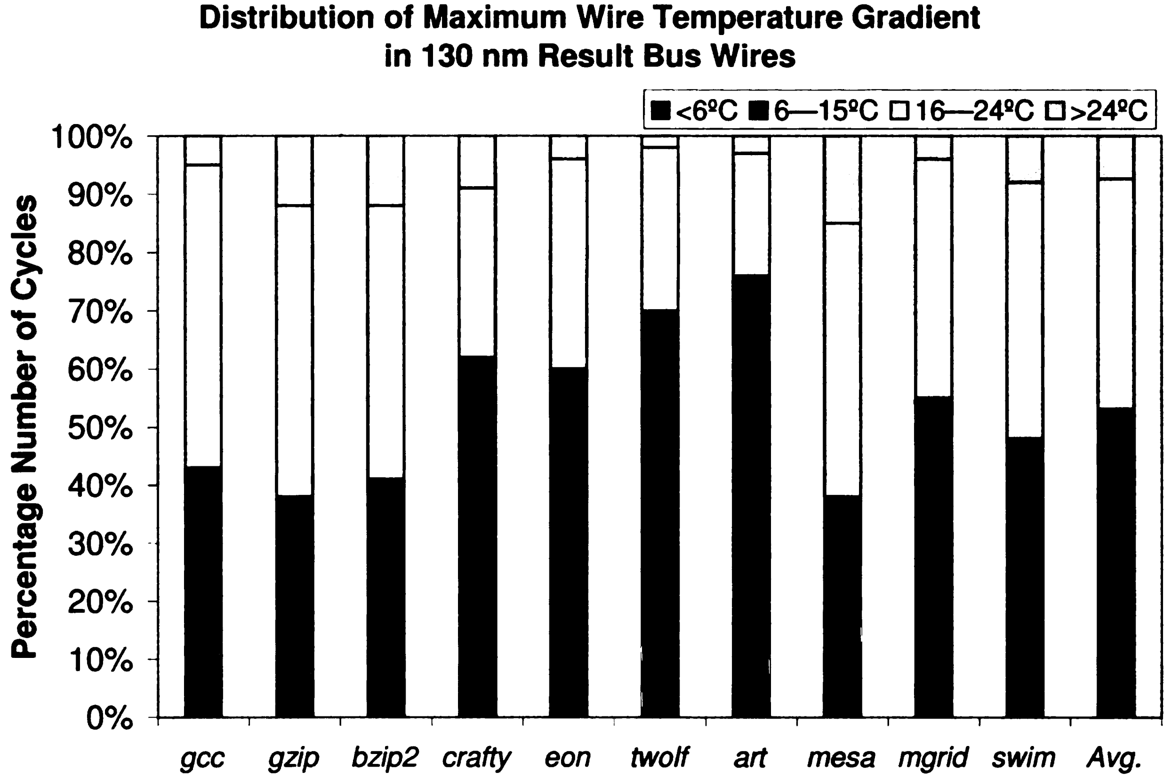


Figure 4.1. Distribution of maximum wire temperature gradients in result bus wires for the 130 nm processor.

result in timing violations in the ALU result bus.

4.4.2 Frequency of Timing Violations

Figures 4.2 and 4.3 show the temperature-induced delay violations per hundred bus references for a 130 nm and a 45 nm processor, respectively, in the ALU result bus using our temperature-dependent delay model when running different benchmarks. The base case—processor operating at nominal clock frequency, 1.68 GHz for 130 nm and 11.51 GHz for 45 nm—is represented by the data series labeled "@ Nominal Freq." in the two plots. For this case, we observe that the average error rate across our benchmark set was 2.27 per hundred bus references for the 130 nm design. For the same processor in 45 nm technology node, the error rate increased to 6.2 per hundred references on the average. Some benchmarks like gcc, gzip, bzip2, and art show higher than average error rates due to the fact that they had higher values of wire temperatures and/or gradients than other benchmarks as observed by results shown in the previous subsection. It should be noted that the timing violation error rates reported here represent temperature-induced violations only; other factors like process variations and voltage drops are not included, as mentioned earlier in Section 4.3.2. In fact, our results show that, in many cases, the extra temperature induced delay can easily overwhelm voltage drop and process variation safety margins allocated by a designer.

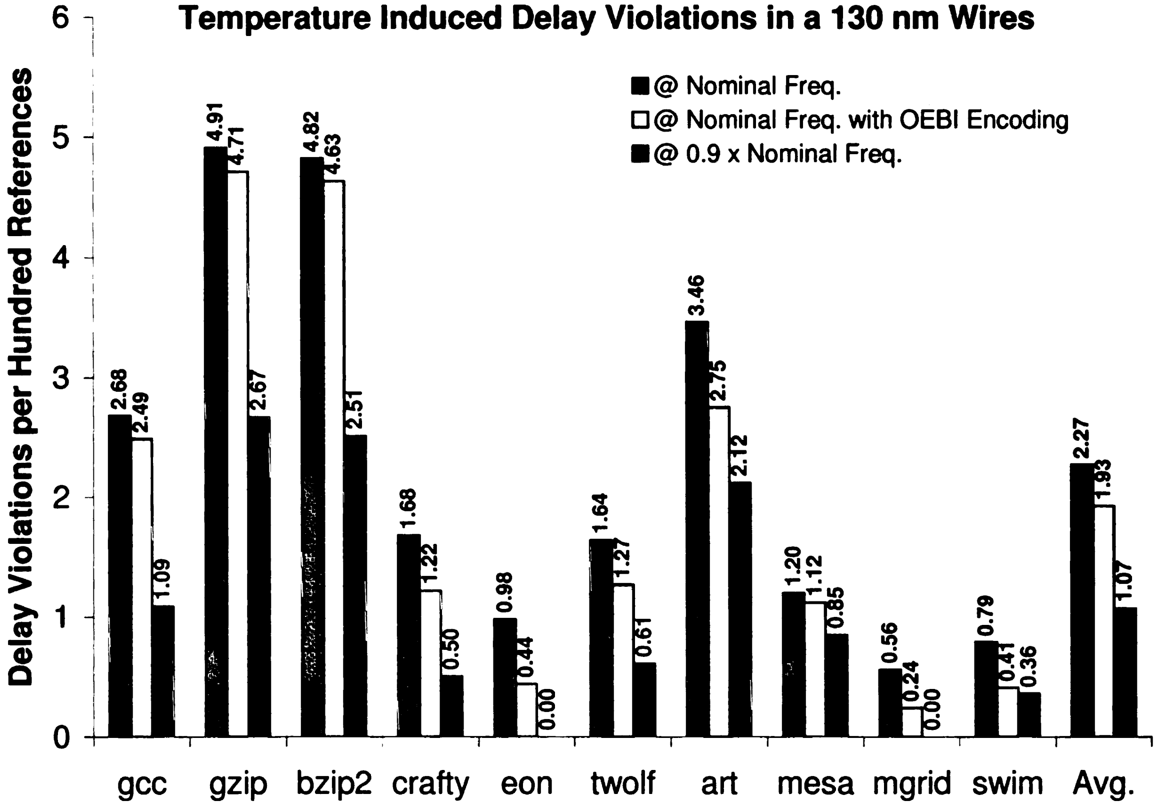


Figure 4.2. The number of temperature-induced violations per hundred bus references occurring across ten benchmark programs in a 130 nm processor.

Most superscalar processor designs adopt such overly conservative methods to

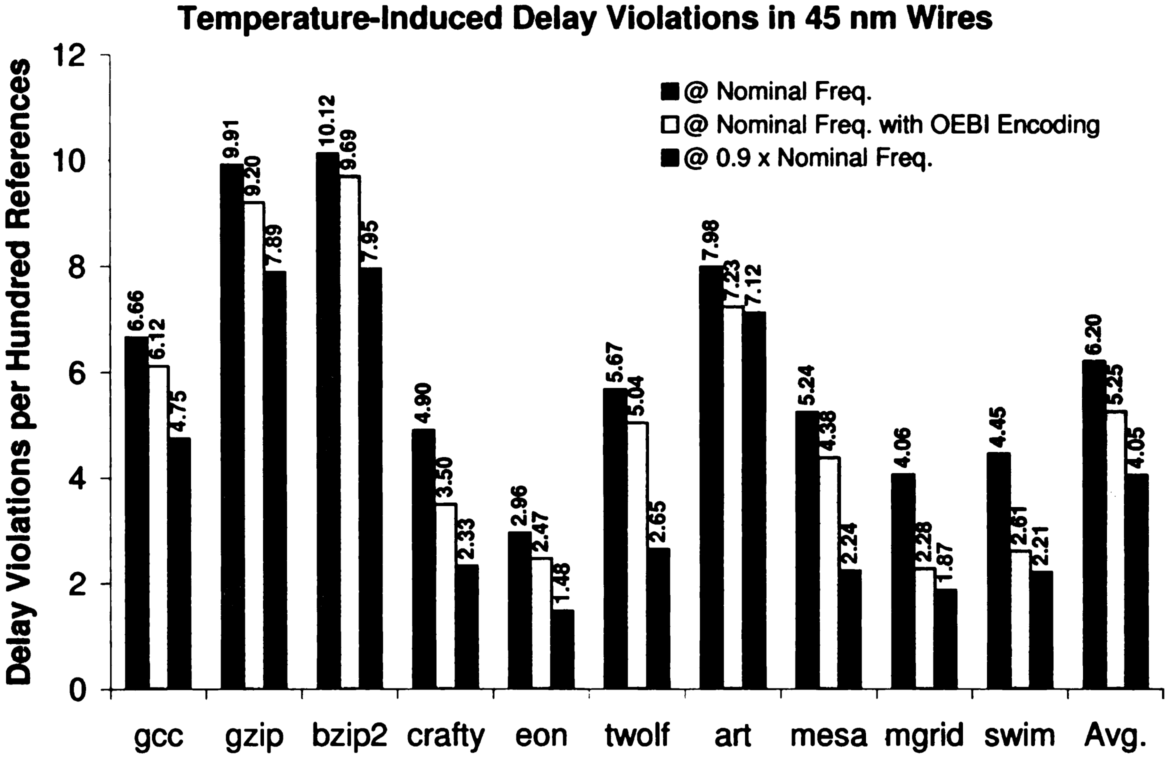


Figure 4.3. The number of temperature-induced violations per hundred bus references occurring across ten benchmark programs in a 45 nm processor.

work around dynamic delay variability-related problems—like using an extra pipeline stage is allocated to account for wire propagation delays [44]. Temperature-distribution aware delay models, such as the one we have developed, can help explore the extent of the timing violation problem during early stage design. Using this knowledge, a designer can implement schemes that address delay variability issues and avoid over-design. For example, results presented in the next subsection show that, by increasing the overall bus clock cycle time by only 10%, the error rates can be halved for a 130 nm design.

As mentioned earlier in Section 4.3.2, not all bus references—even those incurring worst case delays due to peak crosstalk—are likely to trigger timing violations. Cycles in which peak crosstalk conditions occur in a wire, coupled with high Joule heat dissipation and large temperature gradients, have high probability of causing a violation. Violations can occur during non-peak crosstalk conditions too, if wire temperature and/or gradients are large enough. The following results attempt to characterize how temperature-induced delay variations are distributed across various crosstalk conditions.

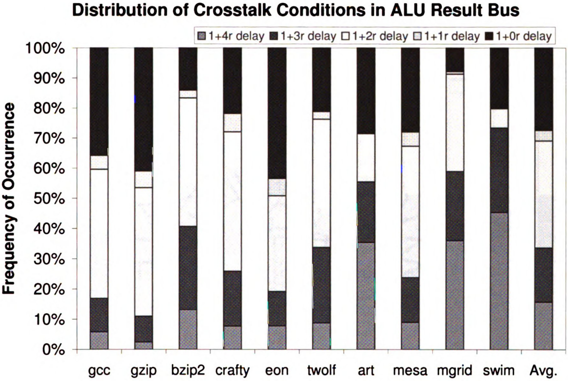


Figure 4.4. This plot shows the frequency of occurrence of five different crosstalk conditions on the bus. See Section 4.3.2 and Table 2.1 for an explanation of these crosstalk conditions. The crosstalk condition determines the actual propagation delay without considering thermal effects.

Figure 4.4 shows the frequency with which different crosstalk conditions occur on the ALU result bus for the programs we analyzed. This is at nominal temperature. It can be seen that the peak crosstalk condition labeled "1+4r delay" occurs only about 10% of the time on average across the benchmark set. The dominant condition is "1+2r delay" which occurs about 40% of the time. Next, in Figure 4.5, we show the distribution of temperature-induced delay violations across the different crosstalk conditions. As the figure shows, crosstalk conditions "1+r delay" and "1+0r delay" contribute a very low percentage (<1%) to number of total delay violations. Other cases have more significant contributions, suggesting that eliminating or reducing these crosstalk conditions can potentially reduce delay variabilities.

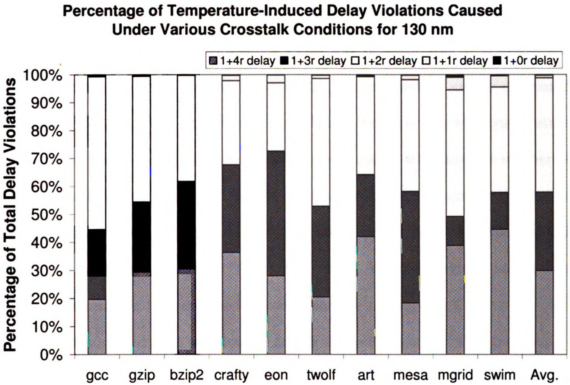


Figure 4.5. Figure shows the percentage of temperature-induced delay violations that correspond to a given crosstalk condition.

From the above discussion, it can be argued that the impact of temperature-dependent delay can be reduced by reducing energy dissipation and hence temperature. We examined two methods of reducing power: (1) a static design-time technique that uses a lower bus clock frequency and (2) a dynamic low power bus encoding scheme called odd/even bus-invert (OEBI) that reduces toggle transitions [53]. The former is represented by the data series labeled "@ 0.9×Nominal Freq." and the latter by "@ Nominal Freq. with OEBI" in Figures 4.2 and 4.3. We observe that slowing the bus down reduces delay violation rates better than applying the encoding scheme. This is because reducing the bus clock frequency results in two outcomes both of which contribute to reducing wire temperature: (1) it slows down the processor resulting in a lower number of bus references per unit time and (2) it increases the clock cycle time over which bus switching energy is dissipated. This combined effect reduces wire power dissipation and hence lowers wire temperatures. In contrast, the encoding scheme only reduces the total amount of bus switching energy dissipated in a cycle and does not affect the cycle time. Hence its impact on wire temperature is lesser. Although, the OEBI encoding scheme is designed to reduce the number of toggle transitions in wires, it has the side-effect of increasing the number of coupling charge/discharge transitions. Thus, in the context of crosstalk, an OEBI-encoded stream will have more number of "1+2r delay" cases. We have observed earlier that somewhat significant temperature-induced violations are possible for this case and this may have contributed additionally to the ineffectiveness of OEBI in reducing error rates. We also observe that frequency reduction is less effective at 45 nm node than at 130 nm node.

4.4.3 Performance Impact

Delay violations, if unchecked, will the impact performance of the processor, requiring an extra cycle to retransmit the data on the result bus. Also, dependent instructions may need to wait longer for dependencies to be resolved and this may cause pipeline stalls. Table. 4.2 shows the instructions-per-cycle (IPC) degradation observed across our benchmark set; the average performance degradation was 4.08%. Note that this is an optimistic estimate since we have assumed that the re-transmission is not affected by delay violations, which is strictly valid only if the bus has cooled down compared to its state during the previous transmission. Our focus, in this work, is not to implement a dynamic scheme that inserts appropriate number of wait cycles to cool the bus after a delay violation is detected. However, such a scheme will only cause the data re-transmission to wait longer than what we have assumed here. Hence, our IPC estimates are lower-bound values. In reality, since the operating clock frequency at 45 nm is much higher than at 130 nm, the performance impact will be much higher at the smaller technology node. Our simulations with 45 nm technology parameters found that the average performance degradation across the ten benchmarks was 11.92% (Table 4.2).

4.5 Summary

This chapter presented models for estimating the Joule heat and wire temperature across the length of a global wire, and to determine its temperature-dependent delay impact. We showed that temperature gradients exist between the sending and receiving ends of a wire and this may lead to dynamic delay variations that can exceed design margins. We used our models to explore the extent of temperature-induced delay violations that may occur in the ALU result bus of a processor in the 130 nm and 45 nm technology nodes using real data from ten SPEC CPU2000 programs.

Microarchitectural simulation results show that delay violations due to temperature gradients are somewhat likely in 130 nm designs—average of 2.27 per hundred bus references for the ALU result bus. In the future 45 nm technology node, the

Tech.	Percentage IPC degradation										
Node	gcc	gzip	bzip2	crafty	eon	twolf	art	mesa	mgrid	swim	Avg.
130 nm	5.20	12.10	8.84	0.58	2.12	3.77	4.84	2.11	0.58	0.66	4.08
45 nm	13.42	23.98	15.66	11.22	12.34	9.87	10.48	9.29	6.09	6.83	11.92

Table 4.2. Performance impact expressed as percentage IPC degradation.

error rate was found to increase to 6.20 per hundred for the same processor design. Commercial 130 nm processor adopt techniques like an extra pipeline stage to combat the influence of dynamic delay variations in wires. However, this leads to over design. Temperature-aware delay models like the one we have developed can be used to explore the design space efficiently and avoid over design. We also found that, by an optimistic analysis, the performance impact of overcoming temperature induced delay violations by re-transmitting data will be about 4.1% in a superscalar design at 130 nm and about 11.9% at 45 nm technology node. We also found that conventional techniques like bus encoding that seek to reduce energy dissipation and potentially wire temperatures have limited impact on alleviating temperature-induced delay violations. Reducing the bus clock frequency had a better impact, reducing average error rate to 1.07 per hundred references, compared to encoding which reduced error rates to only 1.93 per hundred references.

CHAPTER 5

ACTIVITY-AWARE ENERGY AND TEMPERATURE OPTIMIZATION

With increasing energy dissipation and wire temperature in processor bus wires and the inability of existing low-power encoding schemes to address these problems adequately, novel approaches need to be examined. This chapter examines a family of such energy-efficient techniques that rely on data statistics and a first-of-its-kind optimization methodology to reduce bus wire temperatures [98].

5.1 Introduction

On-chip wires are a major impediment to realizing the performance gains that motivate CMOS technology scaling in integrated circuits. At smaller technology nodes, transistors become faster and somewhat energy-efficient but wires become slower because smaller cross-sectional area increases their resistance. To counter this, wire are scaled less-aggressively than transistors. However, this scenario leads to taller and thinner wires that exacerbates parasitic effects like inter-wire coupling capacitance, thus leading to relatively more energy dissipation when wire switching charges/discharges these capacitances. Global signal-carrying wires/lines already contribute a major portion to total chip power dissipation—about 34% in an Intel 130 nm microprocessor [4]. As a result, rising wire temperatures are becoming an important issue in high-performance processor design, especially in current and future nanometer technology nodes since higher temperatures can impact wire delays and electromigration reliability [21, 66, 99].

Wires—like those that constitute address, instruction, data, and ALU result buses—routed in global metal layers are much more susceptible to higher temperatures due to the following reasons: (1) with higher clock frequencies, the amount of energy dissipated in the wire as Joule heat increases compared to the energy dissipated in the repeaters [100], (2) they are furthest away from the substrate which is attached to the heat sink and they are surrounded by low-K dielectrics that have poor thermal conductivity, resulting in inefficient heat removal, and (3) their relatively large geometries result in higher thermal capacitance, i.e., the ability to retain heat. Rising wire temperatures increase wire delays by about 5% for every 20°C rise in temperature [38]. Wire temperature gradients across the length of the wire also affect delay. It has been reported that for a 1 mm long wire with the driver in a hot region and receivers in a cooler region, a temperature difference of 10°C results in a 5 ps ($\approx$ 8%) additional delay at the receiver [93].

5.1.1 Need for Energy and Temperature Aware Bus Design

Real workloads cause bus traffic (in instruction, data, address buses) that exhibit substantial spatial and temporal locality and value redundancy. Switching activities are therefore not random. Further, there is a high degree of correlation between switching (self and coupling) activities of traffic in different execution regions of the same benchmark and across different benchmarks. These characteristics can be exploited using value-aware design of encoding schemes. Previous techniques are typically (inversion-based) dynamic encoding schemes which support a set of encoding modes, one of which is dynamically chosen at run-time in a given cycle in an attempt to reduce bus energy. These suffer from several drawbacks. First, encoding modes supported are those that are effective only for random or worse-case (highly-changing) traffic, which is not the case in realistic workloads. Such value obliviousness limits their effectiveness. We present results showing average energy reductions for dynamic encoding schemes to be only 4.19% (5.32%) at best for data (instruction) traffic across SPEC CPU2000 benchmarks. Second, being dynamic, there is a latency overhead in encoding and decoding and extra area for hardware and control lines. Also, as several earlier works have demonstrated, the efficacy of inversion-based encoders falls rapidly as bus width increases [101,102] and bus partitioning schemes have been proposed to address this issue [103]. However, with partitioned buses, the number of extra lines required for control signals increases and this restricts its attractiveness.

Third, previous schemes attempt to reduce either self or coupling energy, not total bus dynamic energy. Hence their effectiveness will change as the ratio of self to coupling energy changes with technology scaling. Finally, energy and temperature-aware design of high-performance buses are only loosely related. Reducing energy (switching activity) through encoding reduces only the average temperature of a wire (t_{avg}) since it is dependent on total energy dissipated over time which reduces due to encoding. However, existing encoding techniques do not explicitly reduce maximum temperature of wires (t_{max}) since these depend not only on the amount of energy dissipated in the wire itself but also in its neighbors. For example, a low-activity wire (victim) with highly-active neighbors (aggressors) leads to rise in the temperature of the victim wire due to thermal coupling [73]. The effects of thermal coupling can exacerbate electromigration and other related reliability problems in high performance bus wires. Further, due to data locality, a few bus lines are highly-active most of the time and this makes them more susceptible to temperature-induced failures. Such problems can be remedied by combining encoding that reduces t_{avg} with static bit reordering or permutation that seeks to reduce t_{max} by minimizing thermal coupling.

5.1.2 Key Contributions and Results

We evaluate several possible ways of signaling a bit value at design time, and then choose, based on traffic value characteristics, exactly one signaling mode for each bit statically to support in hardware to minimize total bus dynamic energy. We also consider all possible ways of mapping bits to bus lines (*bit ordering* or *permutation*), and then choose, again depending upon traffic value characteristics, exactly one bit ordering statically at design time to support in hardware to minimize total bus dynamic energy. The combination of a particular way of signaling different bits and ordering them on the bus constitutes a *static encoding scheme.* We present an integer linear program (ILP) methodology that evaluates q possible bit signaling modes and all possible bit orderings for an n-bit bus (i.e., it evaluates a total solution space of $q^n \times n!$ encoding modes) based on traffic value characteristics and then chooses an optimal encoding mode that minimizes total bus (self + coupling) dynamic energy. This selection is done at design time using data from microarchitectural simulations and the ILP problems are solved optimally in a matter of minutes. Since only one encoding mode is statically supported in hardware, encoding/decoding (latency, area, and energy) overhead is virtually non-existent and there are no control lines needed.

Since there is substantial correlation between switching characteristics across benchmarks, our static encoding scheme optimized for one set of training benchmarks works very well for a different set of test benchmarks—we refer to this as general-purpose optimization; in this case, we obtain 20.04% (38.78%) average total bus energy reductions with our best scheme for data (instruction) buses. With increasing degrees of customization (suitable for particular application domains or embedded systems), effectiveness improves: we obtain average bus energy reductions of 22.79% (40.77%) for workload-specific and 30.2% (52.1%) for program-specific optimization scenarios for data (instruction) buses. These average percentage bus energy reductions for our static encoding schemes are 5 to 10 times better compared to existing, more complex dynamic encoding schemes.

We present a new way of bit signaling based on Markov models. Markov models have been used in a variety of situations (e.g., branch prediction, instruction compression, etc.), but never in the context of bus encoding or low-power bus design.

We show that lowering bus energy (e.g., even significantly, as with our static encoding schemes), does not necessarily lower *peak wire temperatures* (the highest temperature attained by a bus wire during program run)—in fact, it often may increase it slightly. To address this, we present a novel method of efficiently exploring the peak-wire-temperature and total-bus-dynamic-energy trade-off space using a steady-state wire temperature model. Based on this, we present a new method of introducing thermal constraints into our energy optimization methodology that allows a designer to trade-off peak wire temperature with total bus dynamic energy as desired. For this thermally-constrained, energy-optimal static encoding scheme, we then perform simulations using a detailed per-wire bus thermal model to determine the actual reductions in peak temperature, which we find to be significant. For example, by sacrificing approximately 50% of the energy savings provided by the thermally-unconstrained, energy-optimal version of our scheme, we obtain up to 12.26°C (12.96°C) and on the average 8.03°C (9.24°C) peak wire temperature reductions for data (instruction) buses, while at the same time providing significant average energy savings: 14.24% (16.17%) for data (instruction) buses (still much better than previous work). No previous work has attempted thermally-constrained energy optimization of buses. A recently proposed spreading encoding technique, which targets only peak wire temperature reduction and does not perform any energy optimization, has a number of drawbacks: latency, hardware, and energy overhead of a cross-bar switch network, use of a counter, and we also find that, for the same benchmarks, it does not provide as much temperature reduction.

Finally, if needed, appropriate dynamic bus encoding schemes and the spreading technique for temperature reduction can be applied after our static encoding schemes to further optimize bus energy and temperature. Therefore, in this sense, our work is orthogonal to, although much more effective than these previous works.

The organization of the rest of this chapter is as follows. In Section 5.2, we discuss related work. Next, in Section 5.3, we discuss our methodology. Following that, in Section 5.4, we present our techniques. Then, we present results in Section 5.5. Finally, we summarize in Section 5.6.

5.2 Related Work

Prior work on low-power bus design can be classified into three categories: (1) memory bus encoding schemes that reduce only self transitions, many of which are surveyed in [86], (2) on-chip bus encoding schemes that target both self and inter-wire coupling energy reduction [53,54,104], and (3) wire permutation techniques like those proposed in [105–109] that seek to minimize coupling energy. Memory bus and on-chip bus encoding schemes are dynamic in nature and wire permutation techniques are static.

Our proposed optimization approach differs from prior related work discussed above in many ways. First, wire permutation schemes discussed in [105–107] optimize only inter-wire coupling energy, whereas our scheme combines the benefits of signaling that reduces self transitions, with permutation that seeks to minimize coupling energy. In contrast to the optimization technique suggested in [108], our work considers a wider array of signaling schemes and solves the combined signaling and permutation problem optimally, while they use a greedy algorithm. This contributes to better results using our optimization technique. Compared to the address bus ordering scheme proposed in [109] which can be applied to 8-bit buses only, our scheme can be applied to any bus regardless of bus width or transmitted data. Furthermore, their optimization uses simulated annealing technique, whereas we solve the problem optimally using integer linear programming, for much larger bus sizes and with comparable time complexity. Our optimal static encoding scheme also results in much better energy reductions compared to well-known dynamic low-power bus encoding schemes.

Most importantly, our technique incorporates a thermal optimization methodology for buses which has not been addressed by any previous work. Rising wire temperatures are becoming an important issue in high-performance processor design, especially in current and future nanometer technology nodes [21,66]. To analyze temperature-related issues, microarchitecture-level thermal models like HotSpot [64,66] have been proposed to estimate substrate (active-layer) temperatures. Interconnect thermal models have also been proposed recently [71]. It has been shown that, in global layer interconnects, activity-dependent Joule heat dissipation in the metal leads to thermal coupling between adjacent wires causing maximum wire temperature to shoot up beyond safe design limits [73].

A recent work proposed a methodology called thermal spreading encoding to reduce wire temperatures [110]. In that work, data is bit-shifted periodically before being transmitted on the bus, in an attempt to equalize wire temperatures across the bus by averaging out the Joule heat dissipated across all lines. This technique does not reduce energy dissipation since the coupling energies dissipated in the bus lines remain more or less the same after each shift. Furthermore, it does not alleviate the problem of temperature rise due to thermal coupling between wires. In contrast, our work addresses both these issues through the use of bit re-ordering instead of circular shifting. Spreading encoding, as discussed in [110], is a dynamic technique and uses a $n \times n$-crossbar for an n-bit bus and control logic (counters, etc.) to generate periodic shift signals. Our technique is completely static, incurs negligible overhead, and achieves much better temperature reductions.

5.3 Methodology

We used the SimpleScalar/Alpha microarchitecture-level simulator to design and evaluate our techniques [67]. The Alpha 21264 architecture modeled by this simulator uses a 64-bit (load/store) data bus between the processor and L1 data cache and a 128-bit instruction bus (fetch width = 4) between the processor and L1 instruction cache. Since we have assumed our processor implementation technology to be 130 nm, the clock rate was taken to be 1.68 GHz. We used little-endian Alpha executables of all 26 benchmarks from the SPEC CPU2000 suite with the `ref` input set and ran our simulations on a shared Linux cluster. We selected the SPEC suite as our target workload since pre-compiled little-endian executables for our target platform (Alpha 21264) were readily available for this suite from the SimpleScalar Website [76]. However, our optimization methodology is equally applicable to other application and benchmark suites.

We divided the 26 SPEC benchmarks into a *training* and *test* set with 13 programs in each set chosen arbitrarily. The training set comprised of *gzip, vpr(route), gcc, crafty, gap, vortex, wupwise, mgrid, mesa, art, facerec, lucas,* and *sixtrack*, and the test set had *mcf, parser, eon, perlbmk, bzip2, twolf, swim, applu, galgel, equake, ammp, fma3d,* and *apsi.* For these benchmarks, we used the 100 million single simulation point (SimPoint) sample to collect data for our analysis [77, 78].

5.3.1 Target Scenarios

The three scenarios that we consider are, in the order of increasing degrees of customization, *general-purpose*, *workload-specific*, and *program-specific.* We consider these scenarios to show that our value-aware optimization techniques work well across all scenarios. Specific details of the analysis, design, and test steps for these scenarios are shown in Table 5.1 and are elaborated next.

Analysis Step – Data Collection and Aggregation

We consider several possible ways of signaling a bit value, with exactly one signaling mode for each bit chosen statically at design time depending on traffic value

Step	Target Scenarios		
	General-Purpose	Workload-Specific	Program-Specific
Analysis	Collect energy/cost matrices from SimPoint samples of the 13 *training set* programs and aggregate them.		Collect energy/cost matrices from SimPoint samples of each program individually.
Design	Obtain the static encoding scheme using the *CPLEX* ILP optimizer.		
Test	Apply the static encoding scheme on SimPoint samples of the 13 *test set* programs	Apply the static encoding scheme on a sample of 100M committed instructions that does not overlap with the SimPoint sample for the 13 *training set* programs.	Apply the static encoding scheme on the *same* SimPoint sample used in the analysis step.

Table 5.1. Optimization scenarios considered in this work.

characteristics to minimize total bus dynamic energy. We also consider all possible ways of mapping bits to bus lines (bit ordering or permutation) and then choose exactly one bit ordering statically at design time, again depending on traffic value characteristics. Hence, in the analysis step, we collect energy information for all possible bit signalings and reordering for all pairs of wires; these are represented in the form of *energy cost matrices* whose elements are represented as $e_{l,m}[i][j]$, $\{l, m\} \in \{0, \ldots, q-1\}$, $\{i, j\} \in \{0, \ldots, n\}$, where q is the number of signaling mode choices that we consider. These signaling modes are discussed in detail in the next section.

Each element $e_{l,m}[i][j]$ is obtained by adding two components, both of which are collected using the bus line energy dissipation model [81] in the cycle-accurate simulator for our target buses: the coupling energy $c_{l,m}[i][j]$ dissipated when bits i and j, signaled using modes l and m, respectively, are placed next to each other on the bus, with j being the right-adjacent neighbor of i, and the one-half the self energy $s_{l,m}[i]$ and $s_{l,m}[j]$ of the bits, when they are signaled using the signaling modes l and m, respectively.

When individual energy/cost matrices need to be aggregated across benchmarks $\langle B_0, B_1, \ldots, B_{13}\rangle$, as required in the general-purpose and workload-specific optimization scenarios—See Table 5.1—we add the corresponding elements of the matrices across all benchmarks:

$$e^{total}_{l,m}[i][j] = e^{B_0}_{l,m}[i][j] + e^{B_1}_{l,m}[i][j] + \ldots + e^{B_{13}}_{l,m}[i][j], \forall\ (i,j), \forall\ (l,m). \qquad (5.1)$$

Design Step – Integer Linear Programming

We use *ILOG CPLEX 9.0*, a commercial mathematical programming optimizer, to solve the ILP problems [111]. *CPLEX* provides a C++ interface and a callable library that facilitates reading of input files (containing our energy/cost matrices), examining candidate solutions, and re-solving the problem after adding appropriate constraints. To improve solution times, we also added a greedy approach to find subtours at each node and included elimination constraints for such subtours in our ILP.

Test Step – Getting Results

After the static encoding techniques are designed, results are collected for the benchmarks/samples mentioned in Table 5.1, depending on scenario being considered. The effectiveness of our optimization methodology depends on the degree of similarity between the training and test benchmarks/samples. To probe the extent of similarity, we calculated the values of correlation coefficient r_{xy}, with x representing the test set energy matrix linearized into a vector and y representing the training set energy matrix also linearized into a vector, using *MATLAB* for various signaling schemes listed in Section 5.4.1. These are shown in Table 5.2. The correlation of two variables reflects the linear dependence between them, i.e., it provides an estimate of how well the value of one variable can be predicted from the value of the other. If r_{xy} is closer to unity then they are strongly correlated, which we find is the case with our training and test set coupling energy values, for both general-purpose and workload-specific optimization scenarios.

	r_{xy} for Signaling Mode				
Optimization Type	`org`	`inv`	`trs`	`itr`	`mm`
General-purpose	0.9602	0.9602	0.9609	0.9609	0.9451
Workload-specific	0.9644	0.9644	0.9687	0.9687	0.9610

Table 5.2. Correlation coefficients r_{xy} between test and training set data for various signaling schemes discussed in Section 5.4.1. Since r_{xy} values are close to 1, our training and test sets are well correlated.

5.3.2 Bus Layout and Wire Geometry

We assume a standard model of a bus consisting of a sequence of $n + 2$ parallel, minimum-width, minimum, spaced, identically-dimensioned, co-planar wires $\langle W_{n+1}, W_n, \ldots, W_1, W_0\rangle$ from left to right where $W_1, W_2, \ldots, W_n$ are signal lines and W_0 and W_{n+1} are power/ground lines that act as shields. The bus is assumed to use static logical therefore, it retains a previously-transmitted value until a different one is transmitted. We assume the bus length to be 6-mm, routed in the topmost metal layer, and buffered by identical repeaters spaced equally apart in a microprocessor fabricated in the 130 nm technology node. This global interconnect length is typical in many modern microprocessor floor plans [112]. Uniform repeater insertion methodology was used in this bus to ensure that the propagation delay did not exceed one clock cycle [46]. Several earlier works have also used this repeater model to evaluate buses. Wire geometry parameters were obtained from ITRS [1] and we used FastCap, a three-dimensional capacitance extraction program, to estimate parasitic wire capacitances of each wire [7].

5.4 Static Techniques for Bus Energy and Temperature Optimization

In this section, we present three optimization techniques for designing static encoding schemes for on-chip signal buses and minimizing energy dissipation and wire temperature based on their value characteristics.

5.4.1 Choice of Signaling Modes

We use five candidate signaling modes in our optimization technique, one of which is selected for each bit: original (`org`), inverted (`inv`), transition-signaling (`trs`), inverted transition signaling (`itr`), and Markov model signaling (`mm`). In `inv`, the data on the bit line is always transmitted in inverted form, in `trs`, the XOR of the previous and current original value of the bit is transmitted, and in `itr` the XNOR of the previous and current original value of the bit is transmitted. We chose candidate signaling modes based on three characteristics: (1) potential to reduce self switching energy, (2) potential to reduce coupling energy with neighboring bits, and (3) potential to reduce the temporal distribution of energy-causing transitions. We evaluate our candidate schemes according to these characteristics next.

Inverted signaling

Our optimization uses static inverted signaling (`inv`) as a candidate mode, i.e., the ILP is used to decide if data on a bit line is to be sent in inverted form always, depending on the value characteristics that we obtain for that bit from our training set. For any bit, this mode will be chosen if the amount of self and inter-wire coupling activities it causes with its neighboring wires is less than that for the original mode of transmission. Signaling a bit line with `inv` does not reduce the self switching activity and alters the temporal distribution of energy-dissipating transitions only slightly, but it can potentially reduce the coupling transitions in a significant manner. For example, a two-bit stream can be made completely toggle-free by inverting one of the bits and keeping the other in original mode; a significant amount of energy can be reduced since toggles dissipate most energy compared to charge/discharge and self transitions.

Transition signaling

This signaling mode (`trs`) and its dual (`itr`) affect all three characteristics listed earlier. For bit-streams that are highly-changing, this mode can reduce self switching activity significantly and also reduce coupling transitions with a neighboring `org`- or `inv`-signaled line since every toggle transition is converted to a lower-energy-dissipating charge/discharge transition. It also reduces the temporal distribution of energy-dissipating transitions by converting a highly-changing pattern into a run of ones/zeros.

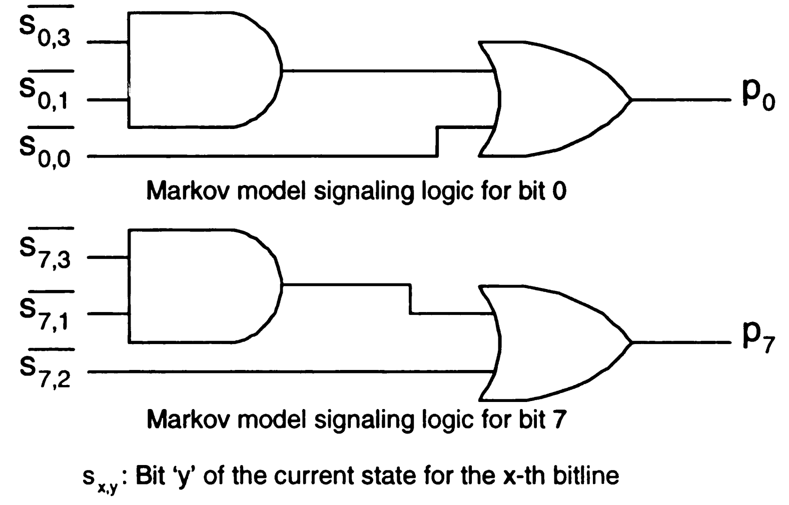
Markov model signaling

In this candidate signaling technique, we use a small amount of hardware at the sending and receiving ends for only the bits that are selected to be signaled using this scheme. To our knowledge, this work is perhaps the first to use Markov model signaling (`mm`) to reduce bus energy dissipation in a value-aware framework. For bits chosen to be signaled using `mm`, we maintain a history of k previous bits from the original data stream that was to be transmitted. These k-bits define the *current state* of the Markov model and it is maintained at both sending and receiving ends. At both ends, the encoding/decoding logic uses this current state to predict the next bit to be sent on the bus. At the sending end, if this prediction matches the actual bit value to be sent, the bus line is held at its current value. Else, we signal a transition on the bus line which indicates a mis-prediction to the receiver. The receiver can retrieve the actual data by sampling the state of the bus lines (transition or no-transition) at the end of the clock cycle since it also has information on the current state of each bus line. The key to an efficient implementation of this signaling scheme is the design of the encoding logic. We analyze SimPoint samples of our 13 training benchmarks to build a prediction table. A portion of the 4-bit/16-state prediction table—for bus lines 0 to 7 of the data bus—is shown in Figure 5.1(a). This can be translated into hardware using standard logic synthesis tools. As an example, we show in Figure 5.1(b), the logic circuits required for implementing the prediction table for bits 0 through 7, obtained by logic minimization using the Espresso tool [113]. These circuits have at most two levels of logic and hence the hardware overheads they impose will be negligible.

We tested Markov model based prediction schemes of varying depth, from 1-bit

Bit	Current State ($s_3s_2s_1s_0$)															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Next Bit Prediction															
0	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0
1	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0
2	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0
3	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0
4	1	0	1	0	1	1	1	0	1	0	1	0	1	0	1	0
5	1	0	1	0	1	1	1	0	1	0	1	0	1	0	1	0
6	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
7	1	0	1	0	1	1	1	0	1	0	1	0	1	0	1	0

(a)



Markov model signaling logic for bit 0

Markov model signaling logic for bit 7

$s_{x,y}$: Bit 'y' of the current state for the x-th bitline

(b)

Figure 5.1. Markov model-based signaling technique. (a) A 4-bit prediction table for the Markov model for bits 0–7 of the data bus obtained by analyzing training set benchmarks. Depending on which bits are selected for Markov model signaling, the corresponding row of the table can be translated to hardware using logic minimization tools. (b) Examples of sending end hardware that would be required for 2 bits (0 and 7) assuming these are chosen to be signaled using the mm scheme. As can be seen, the logic overhead required for mm signaling is very minimal.

(2 states) to 10-bit (1024 states), for their prediction accuracy. As expected, the prediction accuracy improved as the depth of the model increased. However, we found that beyond a depth of 4 (2) for data (instruction) buses, the rate of improvement in prediction accuracy dropped significantly. Hence, we chose the 4-bit Markov model for the data bus and the 2-bit Markov model for the instruction bus.

Henceforth, in this paper, we shall denote the candidate signaling schemes using subscript numbers 0 through 4 instead of `org`, `inv`, `trs`, `itr`, and `mm`, respectively. Let q represent the number of candidate signaling schemes; $q = 5$ in this work. Our ILP formulations use energy/cost matrices or vectors whose individual elements we represent as $e_{l,m}[i][j]$, $\{l, m\} \in \{0, \ldots, 4\}$, $\{i, j\} \in \{0, \ldots, n\}$. For example, $e_{0,1}[i][j]$ represents the energy dissipated between bits i and j when they are placed next to each other on the bus and wire i is signaled using the `org` scheme and wire j using the `inv` scheme. Since there are five signaling schemes, we have a total of 25 energy/cost matrices collected for the training set benchmarks and/or simulation sample, depending on the scenario that we consider.

Note that all energy/cost matrices are $(n + 1) \times (n + 1)$-matrices because we consider the two shield wires as one node, called it a *dummy node*. The solution to our ILPs—MEBO and SBOS—are obtained as Hamiltonian cycles and we use the location of the dummy node to break the cycle into a linear bit order. However, the dummy node is not used in the ILP formulation for MES. The ILP formulations using these notations are discussed next.

5.4.2 Minimum Energy Signaling (MES)

In minimum energy signaling (MES) optimization, we seek to find a static signaling scheme for each bit line of the bus, from among the five possible schemes discussed in Section 5.4.1, with the goal of minimizing total self and coupling energy dissipated. In the ILP formulation, for each adjacent bit pair $(i, i+1)$, we associate 25 binary variables $y_{l,m}[i]$, $\{l, m\} \in \{0, \dots, q-1\}$ representing all combinations of signaling two bits using five schemes. Thus, the binary variable $y_{0,0}[i] = 1$ if both the i-th and $(i+1)$-th bits are to be signaled using the original mode (i.e., the bits are transmitted as in the original traffic). Else, $y_{0,0}[i] = 0$. The formulation of MES in terms of the y variables is given next:

$$\text{Minimize} \sum_{i=0}^{n} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} (e_{l,m}[i] \cdot y_{l,m}[i]) \right\}$$

subject to :

$$y_{l,m}[i] \in \{0,1\}, \forall \{l,m\} \in \{0, \dots, q-1\}, \forall\, i \tag{5.2}$$

$$\sum_{l=0}^{q-1} \sum_{m=0}^{q-1} y_{l,m}[i] = 1, \forall\, i, \tag{5.3}$$

$$\sum_{l=0}^{q-1} y_{l,m}[i] = \sum_{l=0}^{q-1} y_{m,l}[i+1], \forall\, m, \forall\, i \tag{5.4}$$

Constraint 5.2 ensures that the variables take only binary values, Constraint 5.3 ensures that there is only one unique signaling scheme associated with each wire pair, and Constraint 5.4 ensure that the signaling schemes chosen for adjacent wire pairs are consistent. Solving this ILP yields an optimal (minimum energy) signaling scheme for the bus.

5.4.3 Minimum Energy Bit Ordering (MEBO)

In contrast to MES, the next technique, minimum energy bit ordering (MEBO), seeks to minimize inter-wire coupling energy by reordering the bits. Thus, in MEBO, all bits are signaled using the original mode. It is formulated as an instance of the traveling salesman problem (TSP), which is one of the most widely studied combinatorial optimization problems. Simply stated, in the TSP, a salesman needs to visit n cities, visiting each exactly once, and return to the starting city with the minimum total trip cost. In graph theory terminology MEBO is expressed as follows: consider a complete digraph $G = (V, A)$, where $V = \{1, \ldots, n+1\}$ is the vertex set that represents the $n + 1$ bits including the dummy node, $A = \{(i, j) : i, j \in V\}$ is the arc set, and $e_{0,0}[i][j]$ is the energy/cost associated with arc (i, j), i.e., the total energy dissipated if bit j is placed as the right-adjacent neighbor of bit i on the bus, $e_{0,0}[i][i] = \infty, \forall\, i$. Note that we use only $e_{0,0}[i][j]$ in MEBO since all bits are signaled using the original mode only. The problem is to find a minimum energy cycle that includes every node in the graph exactly once, i.e., to find the minimum weight Hamiltonian cycle in G.

The MEBO formulation has one binary variable associated with each arc of G that is represented by $x[i][j]$. In the solution, $x[i][j] = 1$ if bits i and j are to be placed next to each other on the bus, with bit j as the right-adjacent neighbor or i and it is $= 0$ if i and j are not to be placed next to each other. The ILP formulation in terms of the variables $x[i][j]$ is given next:

Step 1 : Minimize $\sum_{\forall (i,j) \in A} e_{0,0}[i][j] \cdot x[i][j]$

subject to :

$$x[i][j] \in \{0,1\}, \forall\, i,j \in V, \tag{5.5}$$

$$\sum_{\forall\, j \in V} x[i][j] = 1, \forall\, i \in V \text{ and } \sum_{\forall\, i \in V} x[i][j] = 1, \forall\, j \in V, \tag{5.6}$$

Step 2 : Solve ILP to get the solution.

Step 3 : Check if the solution has subtours. If none, go to Step 6.

Else, let there be t subtours:

$S = \{S_0(n_0), S_1(n_1), \ldots, S_t(n_t)\}$,

where $S_k(n_k)$ means that subtour S_k has length n_k.

Step 4 : Add subtour elimination constraint:

$$\sum (x[i][j] : \ (i,j) \text{ are in } S_k(n_k)) < n_k, \forall\, S. \tag{5.7}$$

Step 5 : Go to Step 2.

Step 6 : The desired solution (Hamiltonian cycle) has been obtained. Stop.

In the procedure descibed above, Constraint 5.5 ensures that the variables take only binary values. Constraint 5.6 ensures that the in- and out-degrees of every vertex are one, i.e., every bit occurs exactly once in the ordering. Eliminating all possible subtours in the beginning will increase the number of constraints substantially and may lead to a huge time overhead when solving the problem. Hence, we adopt an iterative approach to solve the problem in shorter time. First, we solve the problem with constraints eliminating all possible subtours of two nodes only. Then, we search the solution for the presence of subtours, and if any are found, we add constraints to eliminate those subtours, and then re-solve. We found that almost all problems converge to a feasible solution (i.e., a Hamiltonian cycle) within a few hundred iterations using this iterative method and in a matter of minutes (see Table 5.3).

5.4.4 Simultaneous Bit Ordering and Signaling (SBOS)

In simultaneous bit ordering and signaling (SBOS), we seek to combine the MES and MEBO and optimizations described above. Thus, for each bit, the best signaling scheme—one of the five schemes listed in Section 5.4.2—and the appropriate position of the bits on the bus lines is to be determined simultaneously. Note that combining MES and MEBO does not mean that the energy reductions with SBOS (the combined technique) will be exactly equal to the sum of savings obtained separately with MES and MEBO. In fact, the motivation for combining these problems is to enable the optimizer to select the optimal solution from a richer set of possibilities. Thus, we can view the problem as similar to MEBO but consisting of $n+1$ *supernodes* corresponding to the n bits of the bus and the dummy wire. A supernode contains five nodes, each representing a signaling scheme choice for a bit. By adding constraints that ensure that only one of these nodes is selected for each supernode and that the incoming and outgoing nodes for each supernode are the same, the ILP for SBOS is formulated as described next:

$$\text{Minimize} \sum_{\forall (i,j) \in A} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} (e_{l,m}[i] \cdot x_{l,m}[i][j]) \right\}$$

subject to :

$$x_{l,m}[i][j] \in \{0,1\}, \forall \{i,j\} \in V, \tag{5.8}$$

$$\sum_{\forall j \in V} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} x_{l,m}[i][j] \right\} = 1, \ \forall i \in V, \tag{5.9}$$

$$\sum_{\forall k \in V} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} x_{l,m}[k][i] \right\} = 1, \ \forall i \in V, \tag{5.10}$$

$$\sum_{l=0}^{q-1} x_{l,m}[i][j] = \sum_{l=0}^{q-1} x_{m,l}[j][k], \forall \{i,j,k\} \in V, \forall m. \tag{5.11}$$

Constraint 5.8 ensures that all $x_{l,m}[i][j]$s take only binary values. Constraints 5.9 and 5.10 ensure that there is exactly one outgoing and one incoming node selected, respectively, for each of the $n + 1$ supernodes. Constraint 5.11 ensures that the optimal tour enters and exits through the same node in a supernode (i.e., the signaling schemes chosen for adjacent pairs of bits in the final ordering are consistent). Costs $e_{l,m}[i][i]$, $\forall\ i \in V$ are set to ∞ (a very large integer value). Finally, in SBOS too, constraints for eliminating all subtours with two nodes are added initially, and the problem is iteratively solved as described earlier in Section 5.4.3 until a Hamiltonian cycle that visits all supernodes exactly once is found.

5.4.5 Thermal Optimization Methodology

As described earlier, two adjacent high-activity wires are likely to cause a hot-spot on the bus due to intra-layer heat transfer or thermal coupling between the wires. The peak temperature on the bus occurs at such hot-spots. In the energy optimal bit orderings obtained using MEBO or SBOS, a special class of constraints called *thermal constraints* can be added to prevent high-activity wires from being placed next to each other. Similarly, in MES signaling schemes can be chosen to prevent hotspots in a cluster of wires by adding such constraints. It is to be noted that although adding thermal constraints may decrease the energy saving potential of the energy-optimal bit ordering to some extent, it provides a designer the flexibility to effect a trade-off between optimizing energy and reducing peak wire temperatures. We use the steady state model, described earlier in Section 3.4.3 to determine, approximately, the thermal impact of various orderings and prune thermally-inefficient orderings by adding these constraints. We do this since it is virtually impossible to perform detailed thermal simulations using the model and methodology described in Section 3.4.2, for every candidate solution that we encounter during MEBO/SBOS optimization, and then select the thermally-superior solution. Using the steady state model, the procedure to effect a trade-off between energy and temperature reductions is discussed next.

Steps for thermal optimization

The switching activities of buses vary widely across bits due to the characteristics of data carried on them and hence, the solution space of energy-efficient bit orderings—that are found using MEBO and SBOS—also contains bit orderings in which the wire temperatures are reduced. The steps listed next enable us to find these thermally-superior orderings without affecting the energy optimality by much. *Note that all temperature estimates used in the steps listed below are from the steady-state model.*

1. Find the energy dissipated E_{orig} and peak wire temperature $T_{peak\ -\ orig}$ of

the unmodified bus.

2. Find energy-optimal bit ordering and/or signaling without any temperature constraints using MEBO/SBOS. Let the total energy dissipated in the bus with this (energy-optimal) ordering/signaling be E_{opt} and the ordering/signaling be represented by $\mathcal{B}_0$. Let $T_{peak-opt}$ represent the peak wire temperature corresponding to the permutation $\mathcal{B}$ obtained using the steady state model.

3. Next, we target to reduce the peak wire temperature by a fixed fraction (say η) from its original value in a step-by-step manner. Our target peak wire temperature in the pth step is $T' = (1 - p \cdot \eta) \cdot T_{peak-opt}$, where $p = 1, 2, \ldots,$ etc.. To find a permutation that achieves this peak temperature, we eliminate arcs to/from bit pairs (i, k) for any wire j that has $T(j) \geq (1 - p \cdot \eta) \cdot T_{peak-opt}$. Such a constraint will take the following form in the ILP:

$$x[i][j] + x[j][k] \leq 1 \text{ and } x[j][i] + x[k][j] \leq 1,$$
$$\forall\, i \ \exists\ T(j) \geq (1 - p \cdot \eta) \cdot T_{peak-opt} \qquad (5.12)$$

Adding this set of constraints and solving the ILP, we obtain a wire permutation $\mathcal{B}_p$ that has peak temperature of $T_p \leq (1 - \eta) \times T_{peak-opt}$. Note that since T_p is estimated using the steady state model after obtaining the wire permutation, it can be less than the target temperature. Further, the energy dissipated by this permuted bus E_p will be somewhat worse than E_{opt} The iterative process of adding the thermal constraints and re-solving continues until one of two conditions occur: (1) the ILP becomes infeasible to solve, or the energy of the bit-ordering/permutation E_p becomes worse than that of the original bus ($E_p > E_{orig}$). Figure 5.2 shows a sample temperature vs. energy trade-off curve that will be obtained by following the steps listed above. The curve shows points (T_1, E_1), (T_2, E_2), ..., (T_{10}, E_{10}), corresponding to target temperatures $0.95 \times T_{peak-opt}$, $0.90 \times T_{peak-opt}$, ..., $0.50 \times T_{peak-opt}$.

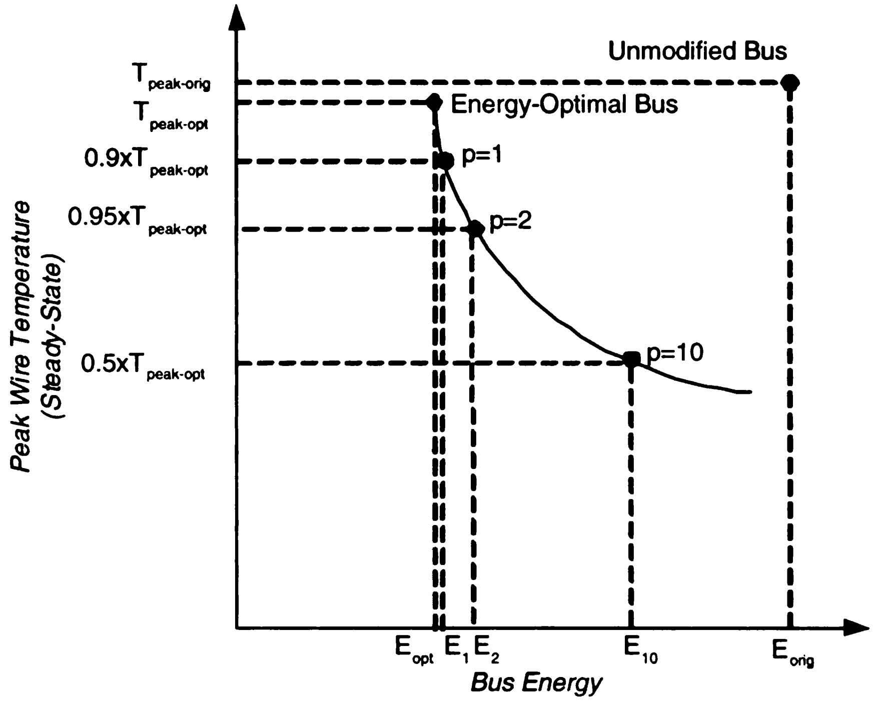


Figure 5.2. Sample peak wire temperature versus bus energy trade-off curve. The thermal optimization steps can be used to obtain curves similar to the one shown here.

The thermal constraint presented in Eq. 5.12 allows only one arc—among $x[i][j], x[j][k], x[j][i]$, and $x[k][j]$—to be present in the solution if the presence of both bits i and k as neighbors causes the temperature in bit j to equal or increase beyond the target temperature. In the *CPLEX* ILP optimizer, the inclusion of thermal constraints using the methodology outlined above can be fully automated. In our experiments, we used $\eta = 0.05$ and succeeded in reducing peak wire temperatures significantly across several benchmarks as shown by results in Section 5.5.5. Furthermore, the extra time taken for temperature optimization did not increase the overall solution time significantly compared to energy-only optimization. The running times are compared later in Section 5.5.

5.4.6 Routing Overheads

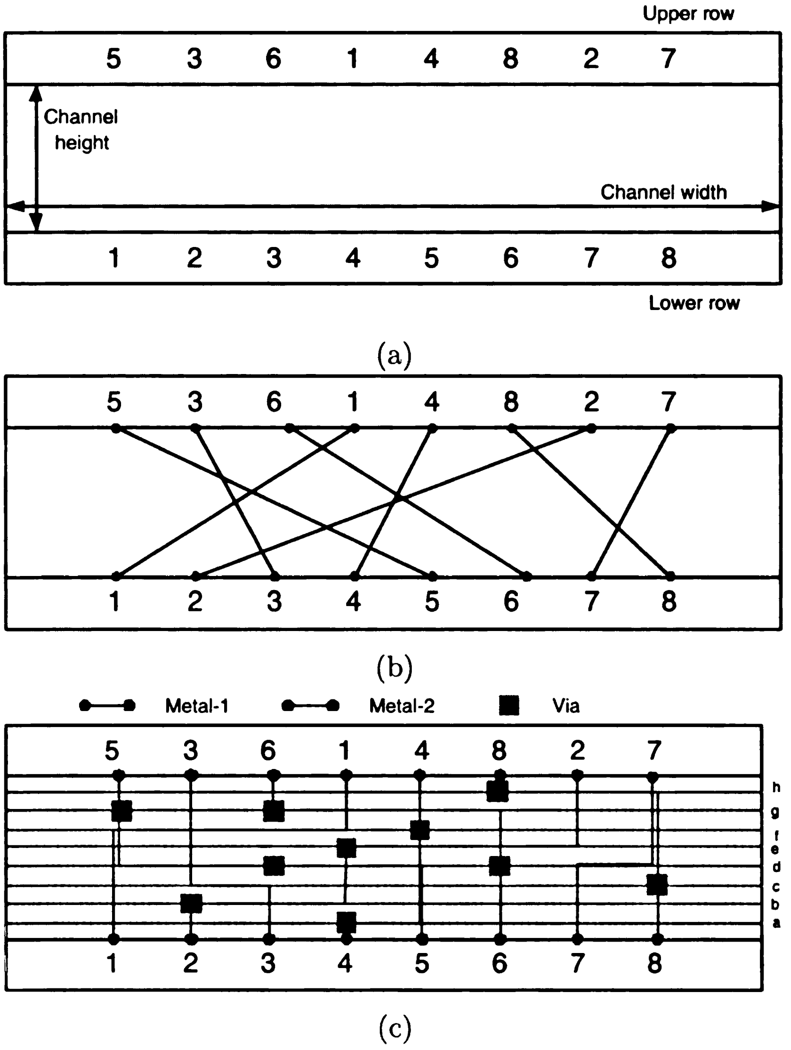
In this subsection, we analyze the overheads for the wire ordering network required to implement MEBO and SBOS. We draw from previous work on efficient techniques for solving the crossing distribution problem [114–116] and use these principles to estimate the area/cost of any ordering network.

Consider two rows, called lower and upper rows (see Figure 5.3(a)), of points called *terminals* and a collection of two-terminal nets $\mathcal{N} = \{N_1, N_2, \ldots, N_n\}$ with each net N_k connecting the terminal numbered k on the lower row to the corresponding numbered terminal on the upper row. The terminals in the lower row are numbered in-order as $1, 2, \ldots, n$ from left to right. The left-to-right ordering on the upper row defines the final re-ordered bus. Let this new ordering be represented by $\Pi = (\pi_1, \pi_2, \ldots, \pi_n), 1 \leq k \leq n$. For example, for the figure shown, $\pi_1 = 5, \pi_2 = 5, \ldots, \pi_8 = 7$.

DEFINITION: Two nets N_i and N_j are defined as *crossing* if $i > j$ and $\Pi(i) < \Pi(j)$ or vice versa. Else, they are *non-crossing*.

DEFINITION: A *matching diagram* is a straight line drawing of the nets for a given permutation Π as shown in Figure 5.3(b) and the straight line representing a net N_i is called a *chord*. The intersection of two chords N_i and N_j defines a *crossing point* C_{ij}. There are ten crossing points shown in Figure 5.3(b).

The notion of inversions can be used to calculate the minimal total number of



(a)

(b)

(c)

Figure 5.3. Routing strategy and overheads for re-ordering. (a) Definition of the routing channel. (b) Matching diagram showing ten crossing points. (c) Two-layer routing strategy using eight horizontal tracks and ten vias.

crossing points ξ for any given Π in the upper row [116]. An *inversion* is any pair (π_i, π_j) such that $i < j$ and $\pi_i > \pi_j$ [117]. Accordingly, $\xi = 10$ can be calculated for the example. The total number of crossing points ξ determines the area/cost overhead of the sorting network in two ways. Intuitively, the number of horizontal wiring tracks in the channel will not exceed ξ, since each crossing point can be taken care of by assigning it a separate track and by using a two-layer wiring strategy. Also, the total number of vias required will not exceed 2ξ, in the worst case. However, in practice, the number of horizontal track and vias required will be less than ξ and 2ξ, respectively. Figure 5.3(c) shows that the routing for this example can be achieved using two metal layers, eight horizontal tracks, and ten vias. Hence the number of crossing points ξ which is the number of inversions of the MEBO/SBOS order that we obtain can be used as a metric to select the re-ordering solution with the best energy-cost tradeoff.

5.5 Results and Discussion

In this section, we present results for energy and wire temperature reductions obtained using our optimal static encoding schemes. In all results, percentage energy reductions are reported with respect to the energy dissipated in an unmodified bus. Table 5.3 lists the running times and number of iterations for problems of different sizes that we solved using *CPLEX* on a SunFire-880 server with two 750-MHz UltraSparc-III CPUs and 8 GB of RAM. The running times for MES optimization were negligible compared to those of MEBO and SBOS and hence they are not shown. As can be seen, these problems can be solved to optimality in a reasonable amount of time.

Problem Type	**Problem Size (Bus Type)**	**# Iterations**		**Total Run Time (mins.)**	
		MEBO	**SBOS**	**MEBO**	**SBOS**
Without Thermal Opt.	64 x 64 (Data bus)	31	42	11	17
	128 x 128 (Instruction bus)	63	75	27	34
With Thermal Opt.	64 x 64 (Data bus)	50	61	24	41
	128 x 128 (Instruction. bus)	71	87	31	60

Table 5.3. Number of iterations and running times for various problem types and sizes.

5.5.1 Energy Dissipation in Processor Buses

We profiled 100M SimPoint samples of all benchmarks in the SPEC CPU2000 suite and recorded their self and coupling activity characteristics. The results of our analysis are ahown in Figures 5.5.1 and 5.5.1. For the data bus, we observed that the transition density per bit did not exceed 0.45 for any benchmark. As expected, the higher order bits (32–63) for the data bus exhibited significantly lower switching activities in integer programs compared to floating-point programs, due to small values being predominant in integer traffic. For instruction buses, switching activities were spread more or less equally in the higher and lower order portions and, here too, it did not exceed 0.5 for any benchmark, with the exception of **vpr** which caused transition densities in the range 0.5-0.8 in a few bit lines.

Next, we present results showing the ratio of self, coupling charge/discharge, and coupling toggle energy dissipated for four kinds of buses: data and instruction address, data, and instruction. To our knowledge, no previous work has profiled such an extensive set of benchmarks and reported their energy dissipation behavior. Such results help designers quantify the important contributors to bus energy dissipation, like self, charge/discharge, or toggle transitions, and explore appropriate static, dynamic, or hybrid encoding techniques to reduce energy dissipation. Figures 5.6–5.9 show the fraction of energy dissipated in self, charge/discharge, and toggle transitions for various benchmarks from the SPEC CPU2000 suites on the Alpha 21264 target systems.

As can be seen, coupling (charge/discharge+toggle) energy forms a substantial portion of the total bus energy dissipation: it contributed 70-75% in the processor buses we analyzed. Among coupling transitions, charge/discharge transitions domi-

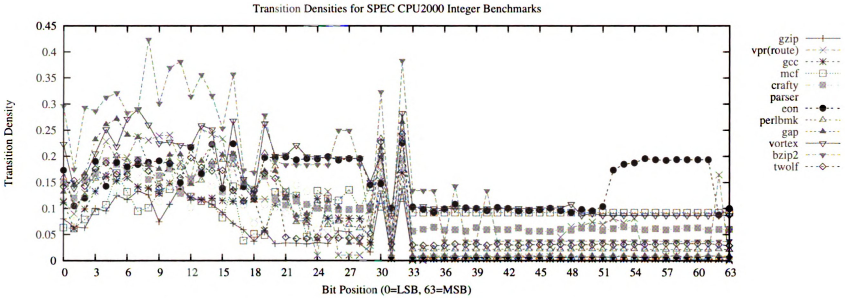


Figure 5.4. Transition Densities for the 13 integer SPEC CPU2000 Benchmarks for 64-bit Data Bus.

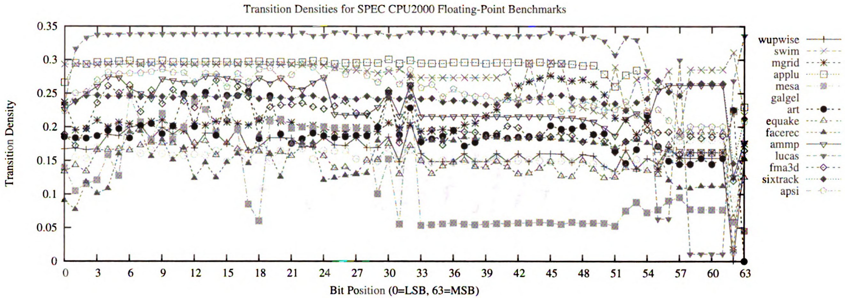


Figure 5.5. Transition Densities for the 13 floating-point SPEC CPU2000 Benchmarks for 64-bit Data Bus.

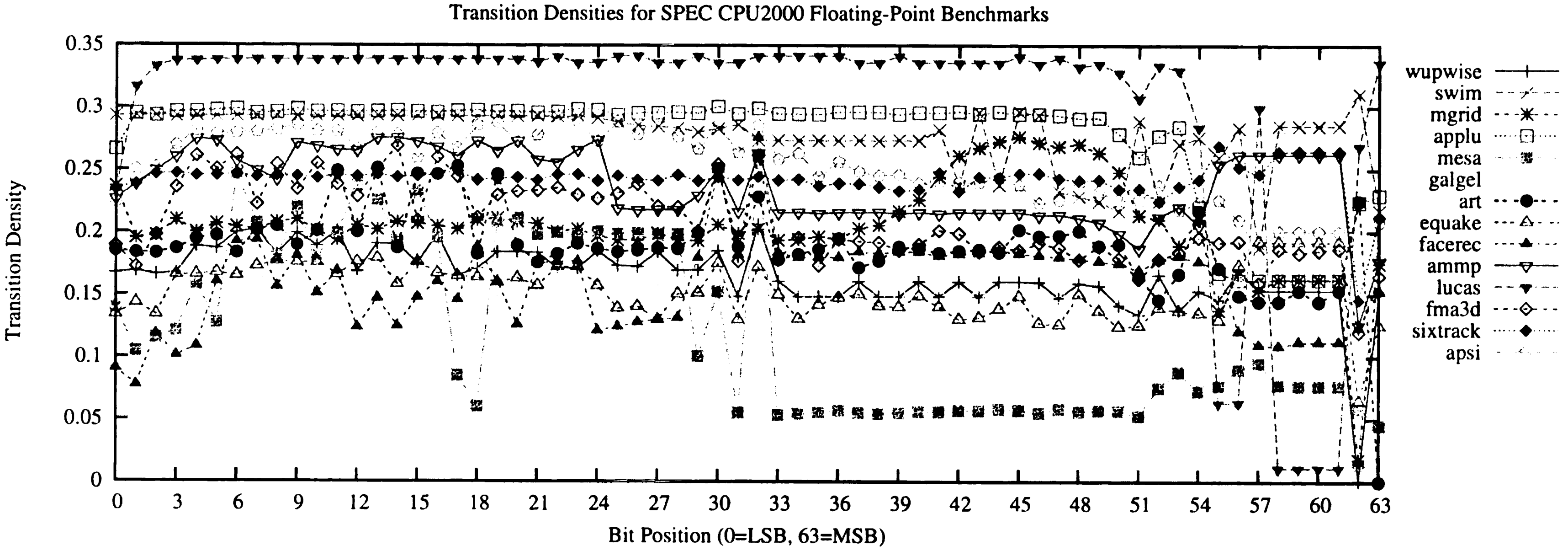


Figure 5.5. Transition Densities for the 13 floating-point SPEC CPU2000 Benchmarks for 64-bit Data Bus.

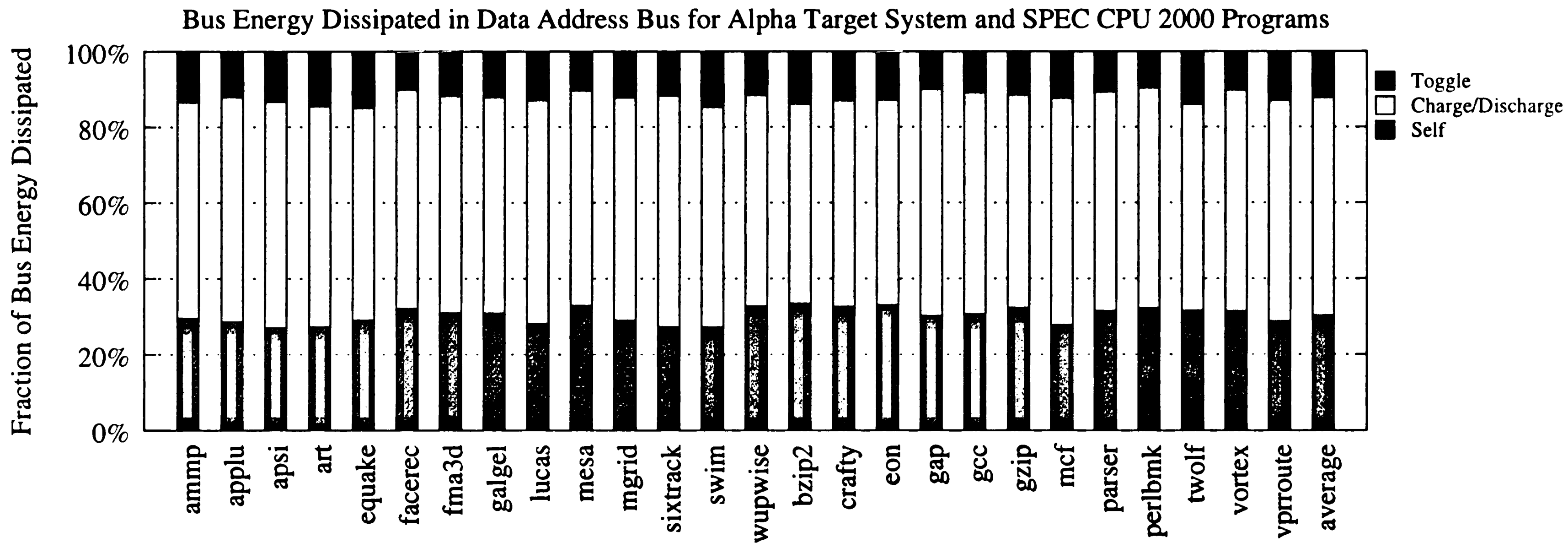


Figure 5.6. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit data address bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

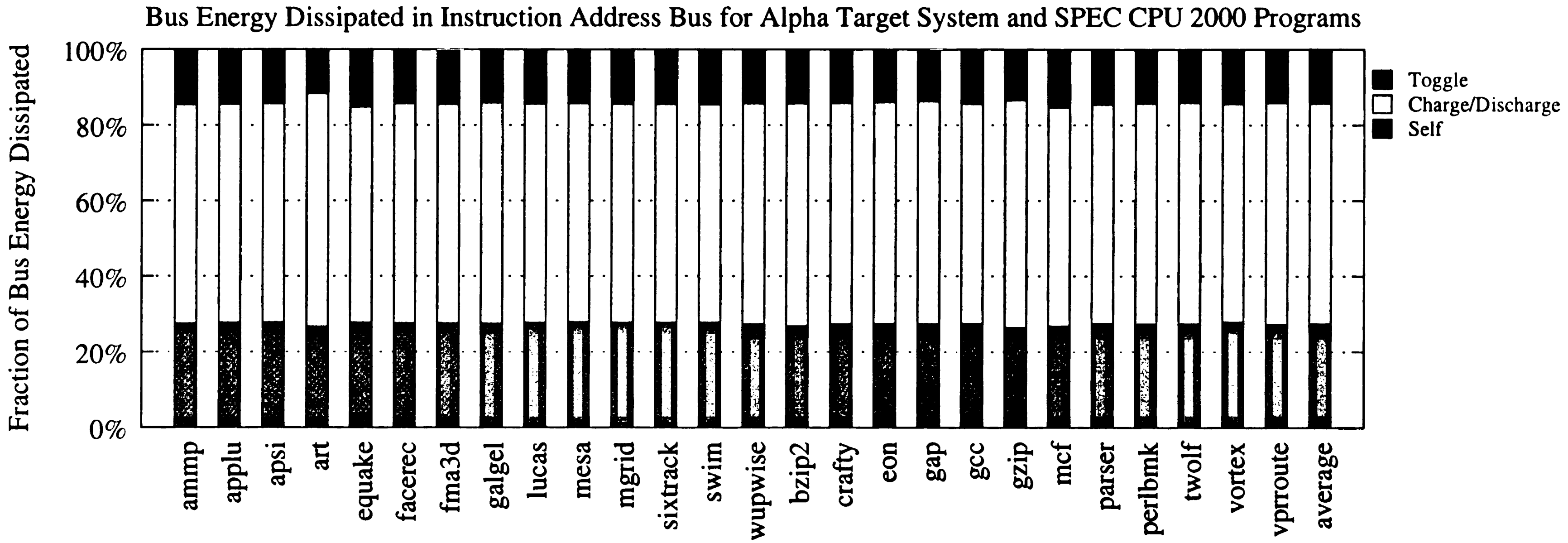


Figure 5.7. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit instruction address bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

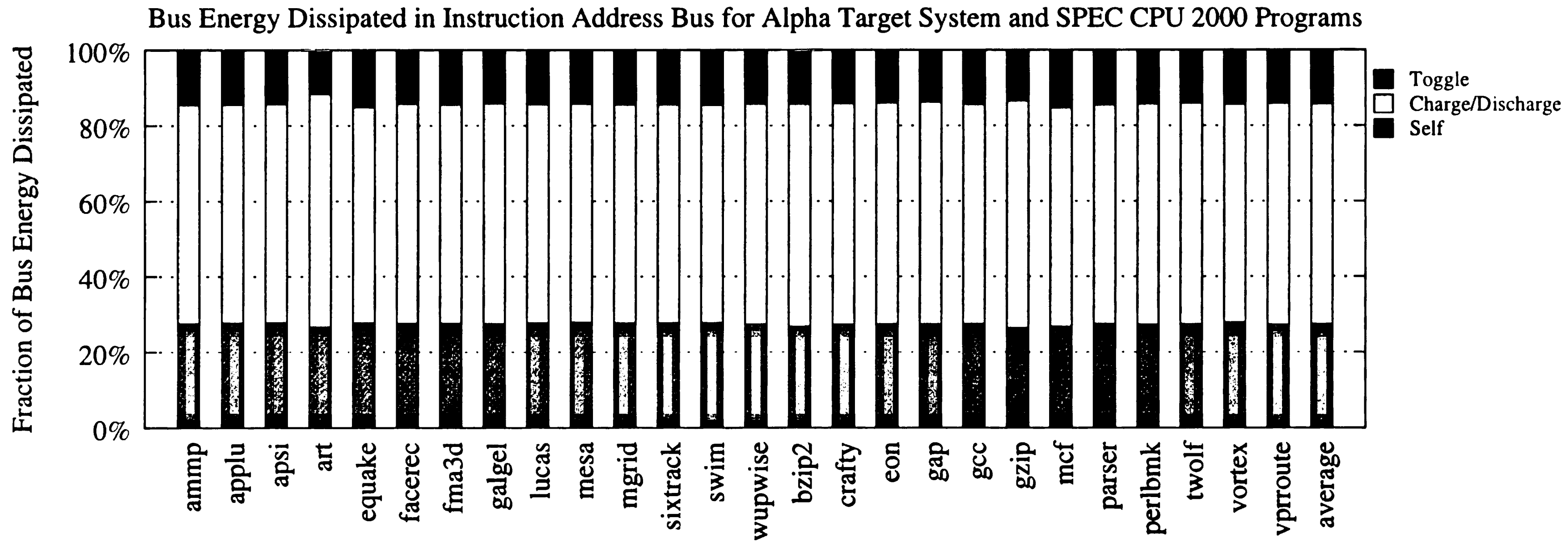


Figure 5.7. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit instruction address bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

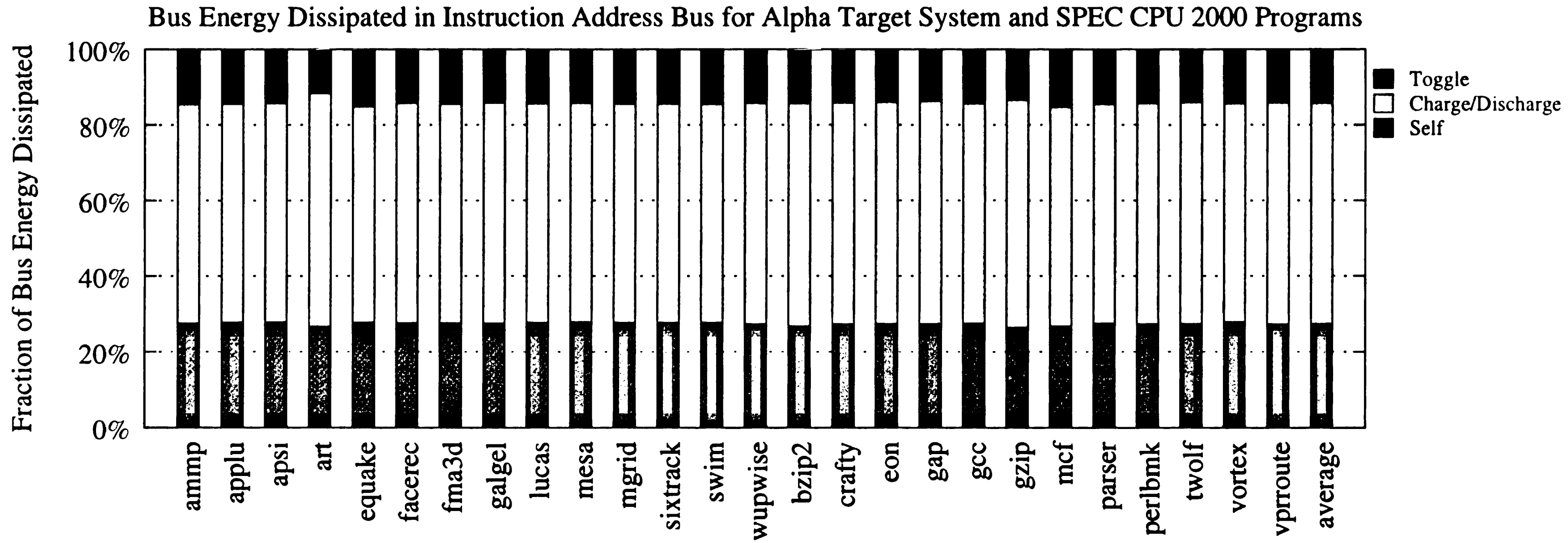


Figure 5.7. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 32-bit instruction address bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

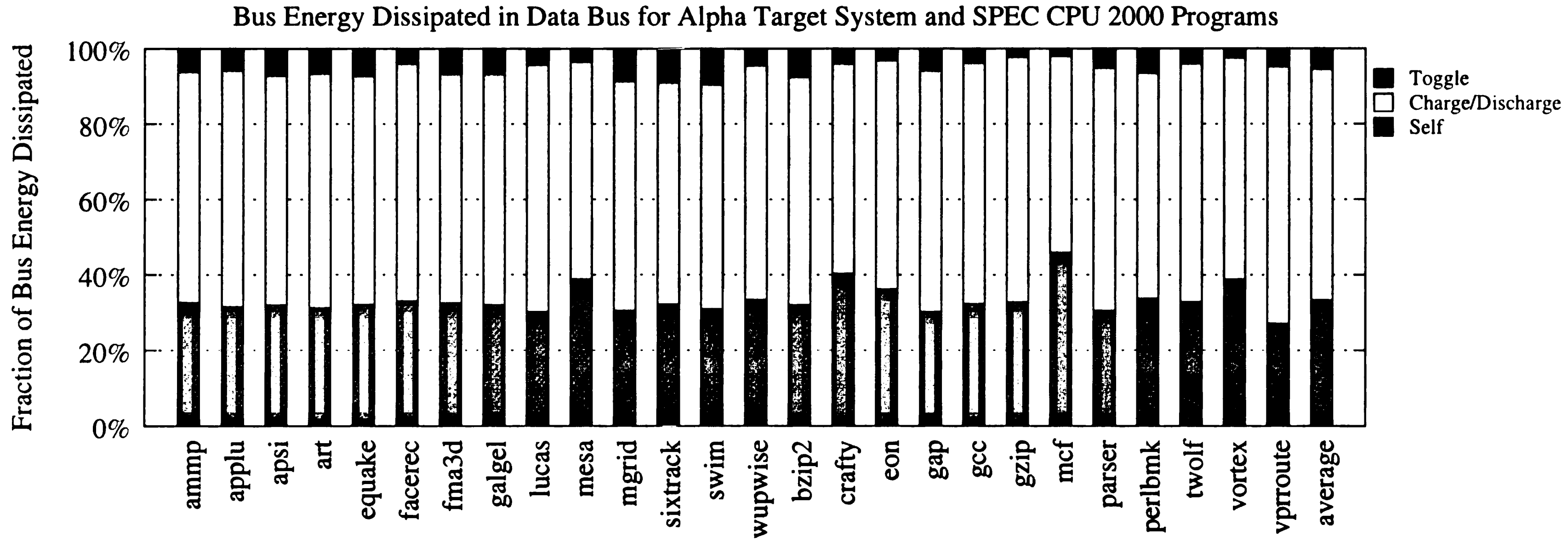


Figure 5.8. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 64-bit data bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

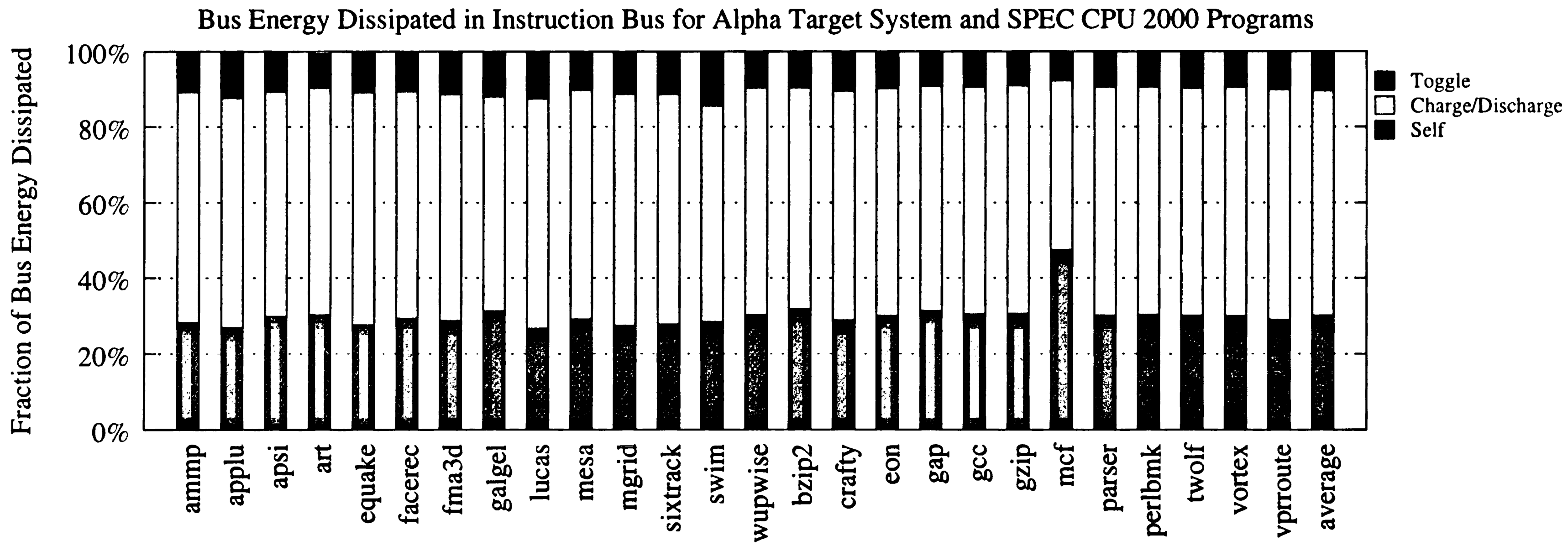


Figure 5.9. Fraction of bus energy dissipated in self and coupling (charge/discharge+toggle) transitions for 128-bit instruction bus in the Alpha 21264 target system while running SPEC CPU2000 programs.

nate. Energy dissipated in toggle transitions are responsible for only less than 20% of total energy; in data buses, they are responsible for only 10% or less. We observed no significant difference between integer programs, shown in the first 14 bars in Figure 5.6–Figure 5.9, and floating-point programs in the SPEC workload.

Next, we present results for energy reductions obtained with our static encoding schemes.

5.5.2 Energy Reduction for General-Purpose Design

For the general-purpose design scenario, our static bus encoding schemes were designed using data collected from SimPoint samples for the training benchmarks and then evaluated on test benchmarks. Results are shown in Figures 5.10 and 5.11. They show that the average bus energy reductions obtained are as follows. MES: 7.81% and 10.96%, MEBO: 11.91% and 19.85%, and SBOS: 20.04% and 38.78% for data and instruction buses, respectively. On the average, we find that optimizations on the instruction bus yield better results than on the data bus. We also observe that SBOS is easily the best scheme for both data and instruction buses.

5.5.3 Energy Reduction for Workload-Specific Design

To evaluate the effectiveness of our techniques in the workload-specific design scenario, statistics collected for SimPoint samples from 13 training set benchmarks were aggregated and used to obtain the optimal static encoding schemes. The scheme was then tested on non-overlapping samples from the same set of benchmarks. This non-overlapping sample was arbitrarily selected as a block of 100M committed instructions after the first 10 billion instructions of program execution. From the results shown

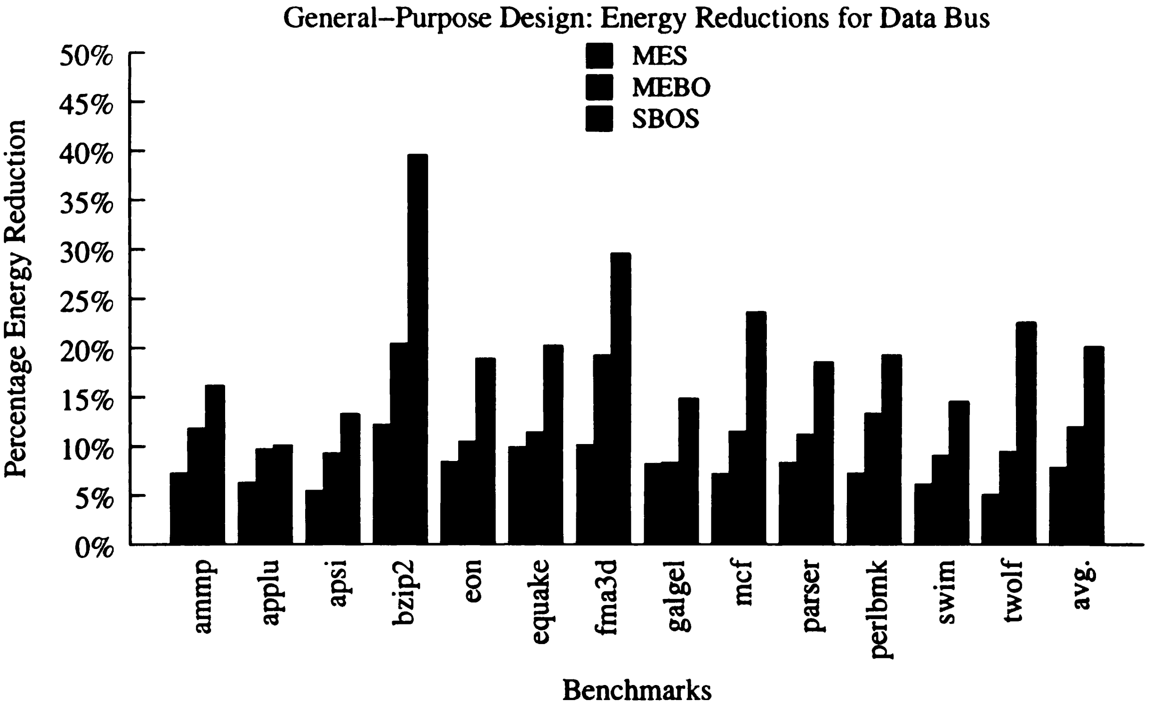


Figure 5.10. Energy dissipation results for general-purpose design for the 64-bit data bus. Statistics collected on 13 training set benchmarks were used to obtain the optimal static encoding schemes. These were tested on 13 other (test set) benchmarks. Average energy reductions are MES: 7.81%, MEBO: 11.91%, and SBOS: 20.04%.

in Figures 5.12 and 5.13, we observe that the average energy reduction across the benchmarks for the three schemes are as follows. MES: 9.73% and 10.43% for instruction bus; MEBO: 15.97% and 21.25% for instruction buses; and SBOS: 22.79% and 40.77% for data and instruction instruction buses, respectively. Our results indicate that workload-specific energy optimizations on the instruction bus are likely to yield better results than on the data bus. Among the three different schemes we proposed, SBOS gives the best results. This is expected because it combines the benefits of signaling as well as bit ordering. Table. 5.4 shows the actual bit ordering and signaling for the data bus that was obtained using the training set. The corresponding table for the instruction bus is not shown due to space constraints. For

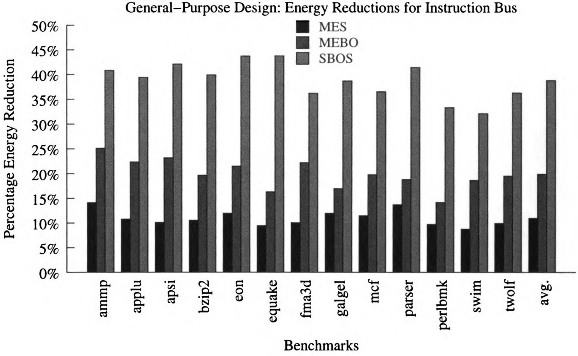


Figure 5.11. Energy dissipation results for general-purpose design for the instruction bus. Average energy reductions are MES: 10.96%, MEBO: 19.85%, and SBOS: 38.78%.

both data and instruction buses all five signaling schemes were chosen. In particular, the original mode of signaling was retained for 36 (38) lines, inversion was chosen for 12 (45) lines, and Markov model signaling for 11 (40) lines in the data (instruction) bus. Relatively, transition and inverted transition signaling were chosen for a fewer number of wires, a total of 5 (5) nodes in data (instruction) bus.

5.5.4 Energy Reduction for Program-Specific Design

In program-specific design, coupling energy/cost matrices collected for the SimPoint samples of each benchmark are used to design a signaling/encoding scheme and tested on the same benchmark and sample. This is expected to yield best results as the static encoding schemes are specific to that sample and benchmark. Results for

Wire #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit #	48	31	52	47	37	41	39	36	54	43	45	62	38	46	42	35	57	61	50	63	49	59	56	58	51	33	55	34	44	60	53	40
Sigl.	♠	♠	△	♠	♠	♣	♠	△	♣	◇	♠	◇	△	△	♠	♠	♠	△	◇	♡	△	△	♠	♡	△	♠	♠	♠	△	♠	♠	♡
Wire #	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Bit #	24	22	26	23	27	28	14	15	10	8	3	18	29	2	0	12	9	17	13	16	32	30	20	21	25	19	11	7	1	4	6	5
Sigl.	♡	♠	♠	♠	♡	♠	♡	♠	♡	♠	♠	♠	◇	♡	♡	♡	♠	♠	♠	♠	♠	♠	♠	♡	♡	♠	♠	♠	♠	♡	♠	△

Table 5.4. Optimal signaling and ordering obtained for workload-specific design of the data bus (0=LSB, 63=MSB). ♠ = `org`, ♡ = `inv`, ◇ =`trs`, ♣ =`itr`, and △ =`mm`.

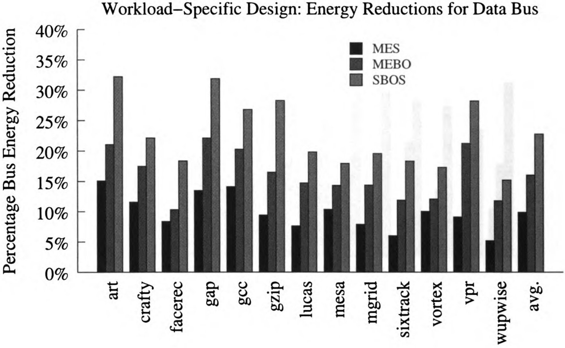


Figure 5.12. Energy dissipation results for workload-specific design of the 64-bit data bus. Statistics collected for SimPoint samples from 13 training set benchmarks were aggregated and used to obtain the optimal static encoding schemes. These were then tested on a non-overlapping sample from the same set of benchmarks. The average energy reductions are MES: 9.73%, MEBO: 15.97%, and SBOS: 22.79%.

26 benchmarks are shown in Figures 5.14 and 5.15 for data and instruction buses, respectively. For custom optimization of the data bus, energy reductions in the range of 50-60% can be obtained for some benchmarks like art, bzip2, and fma3d with SBOS. In comparison, dynamic bus encoding schemes BI and OEBI provide only up to about 10% energy reduction for a few of the benchmarks, for data and instruction buses. For a majority of the programs, reductions with BI and OEBI are less than 5% for data as well as instruction buses. The average energy reductions were BI: 4.19%, OEBI: 1.58%, for the data bus and BI: 2.63%, OEBI: 5.32%, for the instruction bus. For the data bus, where self switching activities are dominant, OEBI results in an energy increase for some benchmarks since many higher order lines remain inactive. This is

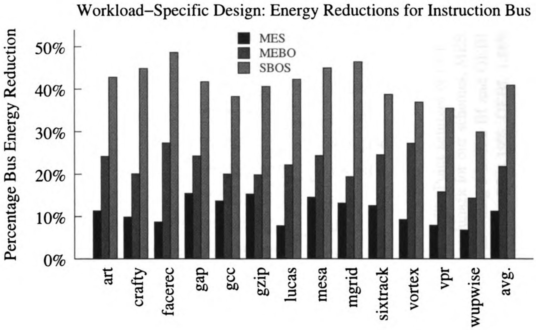


Figure 5.13. Energy dissipation results for workload-specific design for the 128-bit instruction bus. The average energy reductions are MES: 10.43%, MEBO: 21.25%, and SBOS: 40.77%.

because it does not take into account both self and coupling activities when deciding on the inversion mode. As a result, self switching activities increase significantly in the encoded data stream since the mode chosen to reduce coupling energy does not necessarily reduce total (self + coupling) energy. The switching activity in the instruction stream is coupling dominant. Hence OEBI performs better on this type of data. However, the energy reductions are only marginally better compared to BI. Our static encoding schemes, which optimize for both self and coupling energy by considering signaling and reordering, show much better energy reductions than previous dynamic encoding scheme for all benchmarks. The average energy reductions are: data bus, MES: 19.7% and 21.7%, MEBO: 23.25% and 32.1%, and SBOS: 30.2% and 52.1% for data and instruction buses, respectively.

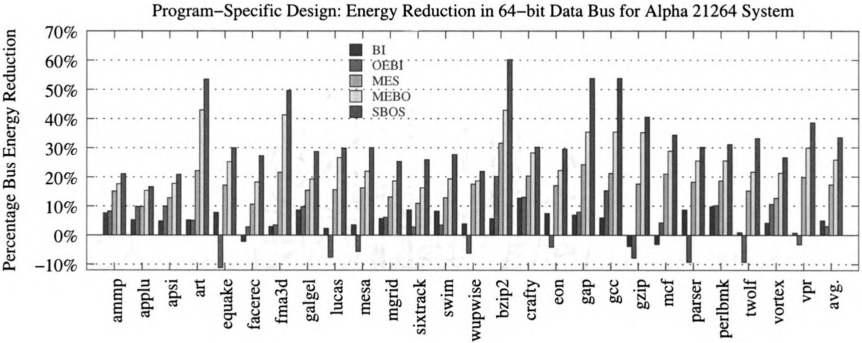


Figure 5.14. Energy reduction results for program-specific design. Statistics collected for SimPoint samples of each benchmarks was used to obtain the optimal static encoding schemes specific to that benchmark for our schemes, MES, MEBO, and SBOS. These were then tested on the same sample. Results for dynamic encoding schemes BI and OEBI proposed in previous work are also shown. The average energy reductions for the data bus are BI: 4.19%, OEBI: 1.58%, MES: 19.7%, MEBO: 23.25%, and SBOS: 30.2%.

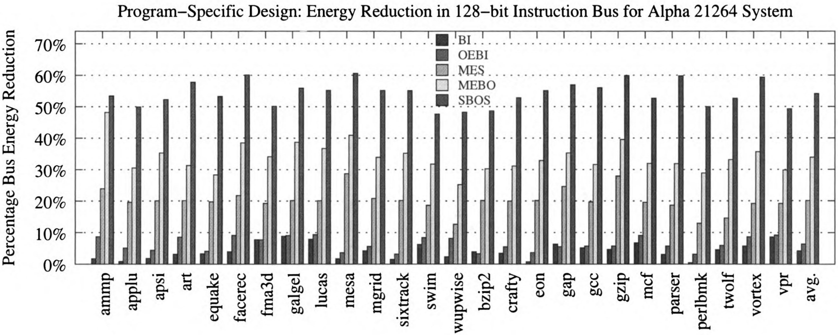


Figure 5.15. Energy reduction results for program-specific design. Statistics collected for SimPoint samples of each benchmarks was used to obtain the optimal static encoding schemes specific to that benchmark for our schemes, MES, MEBO, and SBOS. These were then tested on the same sample. Results for dynamic encoding schemes BI and OEBI proposed in previous work are also shown. The average results for the instruction bus are BI: 2.63%, OEBI: 5.32%, MES: 21.7%, MEBO: 32.1%, and SBOS: 52.1%.

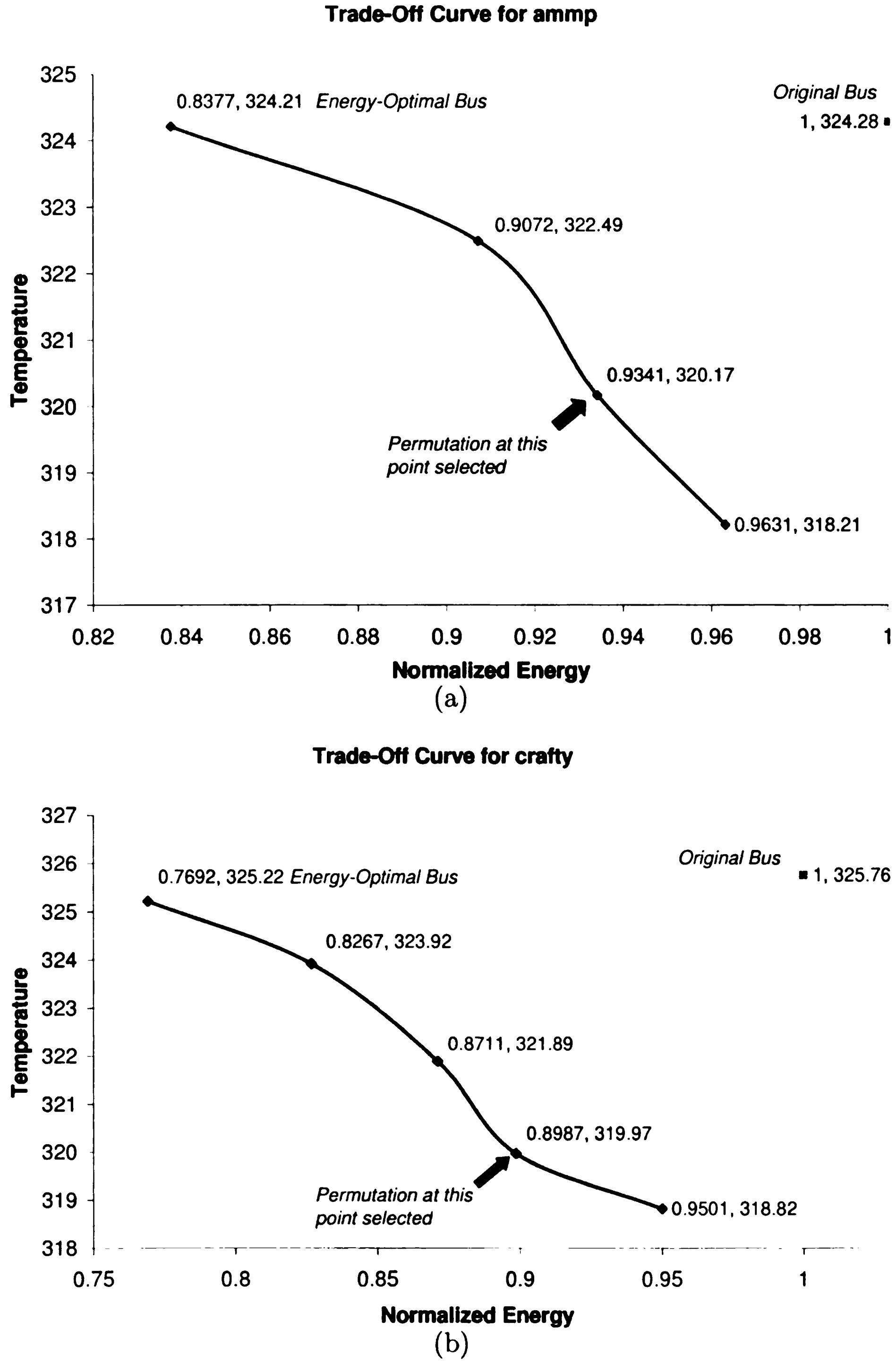
	ammp	crafty	eon	gcc	gzip	lucas	mesa	mgrid	swim	twolf	Avg.
	Data bus peak wire temperature										
Orig. temp. (K)	324.28	325.76	330.04	324.82	330.56	331.71	325.54	327.49	326.34	327.18	327.37
SBOS w/o TC (K)	324.31	325.22	329.82	325.81	331.94	331.08	326.42	329.11	325.97	327.54	327.72
SBOS with TC (K)	320.73	319.38	319.87	319.21	321.18	319.45	319.15	319.76	318.17	320.02	319.69
Redn. w.r.t. orig. (K)	3.55	6.38	10.17	5.61	9.38	12.26	6.39	7.73	8.17	7.52	8.03
Energy redn. (%)	6.59	10.13	9.87	14.24	15.23	13.51	10.09	9.57	11.79	10.20	11.12
	Instruction bus peak wire temperature										
Orig. temp. (K)	329.77	331.9	349.55	329.11	333.38	339.87	344.82	341.17	337.42	340.58	337.76
SBOS w/o TC (K)	329.75	331.92	348.57	328.71	332.96	340.56	342.87	340.79	338.24	338.34	337.27
SBOS with TC (K)	320.10	322.18	338.72	320.56	324.92	330.82	331.96	329.50	331.73	329.80	328.03
Redn. w.r.t. orig. (K)	9.67	9.72	10.83	8.55	8.46	9.05	9.91	12.96	5.69	10.78	9.24
Energy redn. (%)	12.39	14.11	13.30	15.77	15.89	14.20	11.67	15.03	16.17	15.82	14.44

Table 5.5. Thermal Optimization Results. Peak wire temperatures (K) in data and instruction buses for SBOS scheme with and without thermal constraints (TC) applied during optimization. The methodology described in Section 5.4.5 was used to obtain the trade-off curves in Figures 5.16–5.20 and the wire permutations that resulted in bus energy reduction closest to $0.5(1 - \frac{E_{opt}}{E_{orig}})$ were chosen from each benchmark's tradeoff curve. Results shown here are for detailed thermal simulations with this permutation.

5.5.5 Wire Temperature Reduction

Our work is the first of its kind to design static encoding schemes that seek to reduce peak wire temperatures in addition to reducing bus energy. The thermal optimization methodology was explained earlier in Section 5.4.5 and thermal models used to estimate activity-dependent wire temperatures in Sections 3.4.2 and 3.4.3. Table 5.5 shows the reductions in peak temperature that we obtained for different benchmarks with and without the thermal optimization methodology. In this table, we show the peak wire temperature observed for the unoptimized (original) bus and the wire temperatures after SBOS with thermal constraints was applied. We show results for temperature-optimized SBOS only since best results were obtained using this technique; temperature reductions for MEBO were consistently lower. This is expected because the SBOS optimization technique has a larger solution space from which it can choose the best solution.

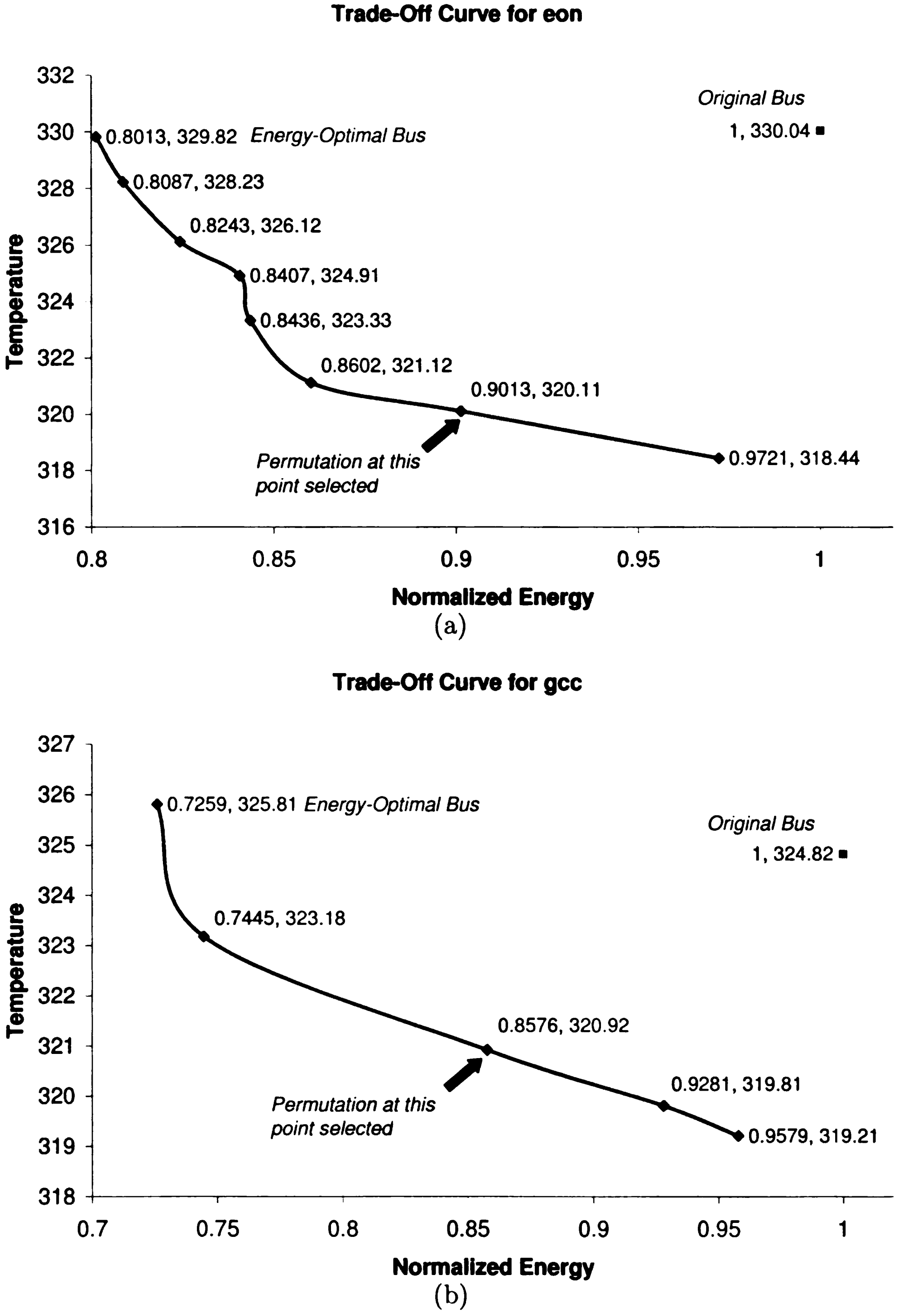
From Table 5.5, we note that applying SBOS without thermal constraints, which reduces energy of the bus by 20% or more for data buses (Figure 5.13), does not always reduce the peak wire temperature observed in the simulation window. In fact, it is seen that, for the data bus, the average peak temperature, across the ten benchmarks studied, actually rises slightly above that of the original bus by 0.35°C and it falls only slightly for the instruction bus by 0.49°C, which is not a lot considering the significant energy reductions we obtained for these buses. This can be attributed to the fact that the energy optimization does not explicitly consider thermal coupling when deciding on the bit ordering and signaling. However, by adding explicit thermal constraints using the methodology in Section 5.4.5, temperature of the hottest wire can be reduced. Recall that our thermal optimization methodology trades off some



(a)

(b)

Figure 5.16. Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for ammp and crafty. The permutation selected for each benchmark was the one that resulted in bus energy reduction closest to $0.5(1 - \frac{E_{opt}}{E_{orig}})$ compared to the original bus.



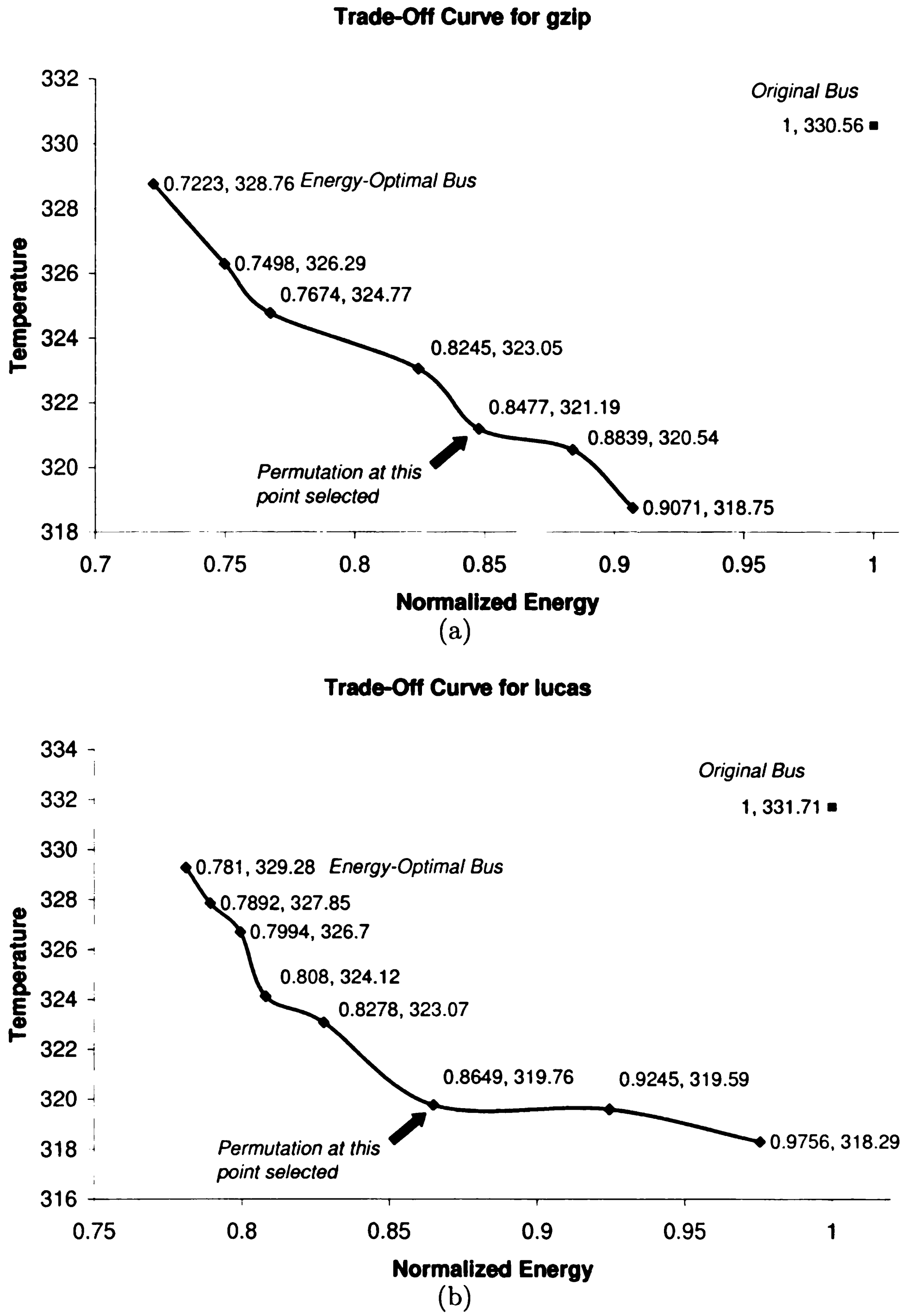
Trade-Off Curve for eon

(a)

Trade-Off Curve for gcc

(b)

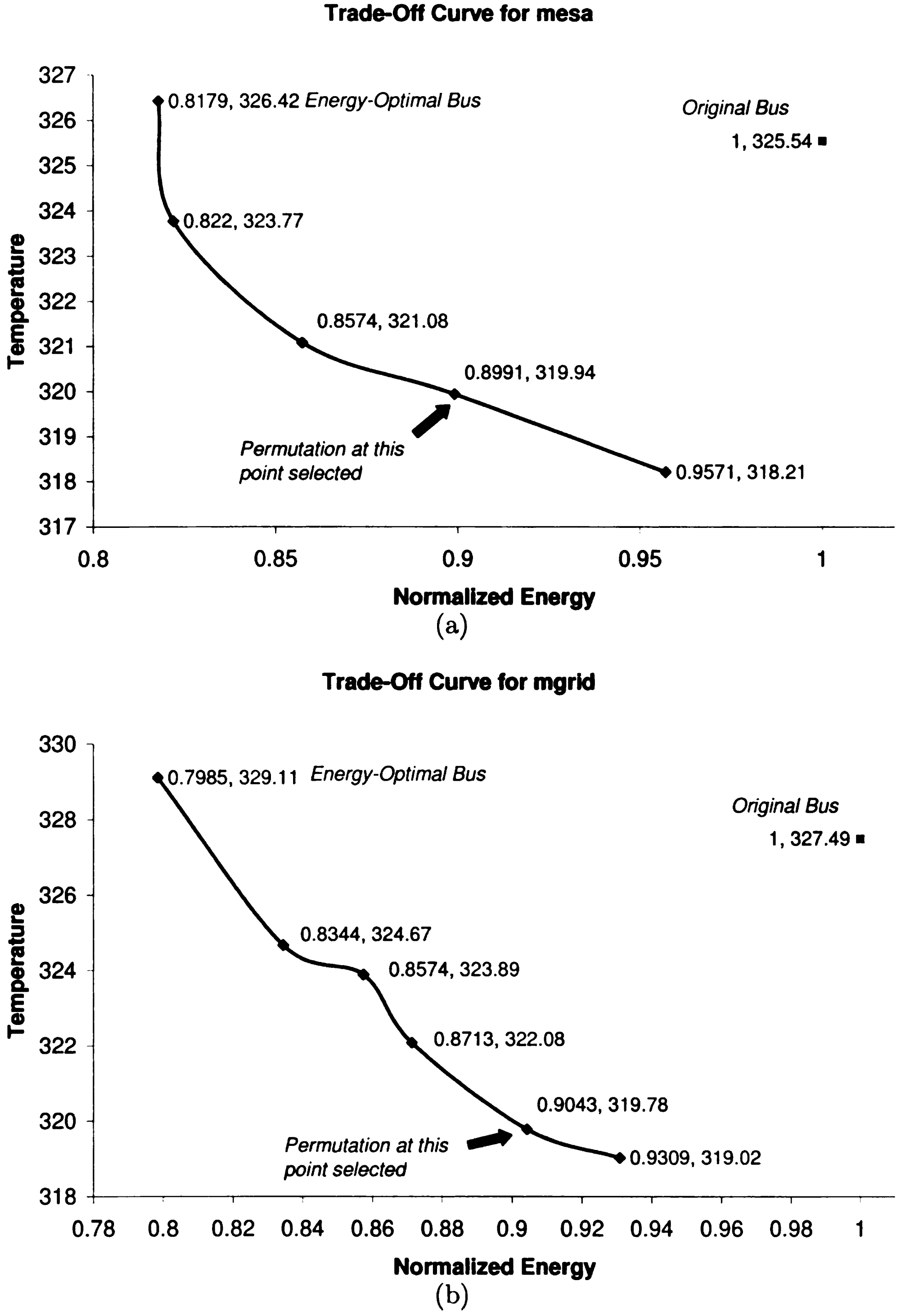
Figure 5.17. Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for eon and gcc.



(a)

(b)

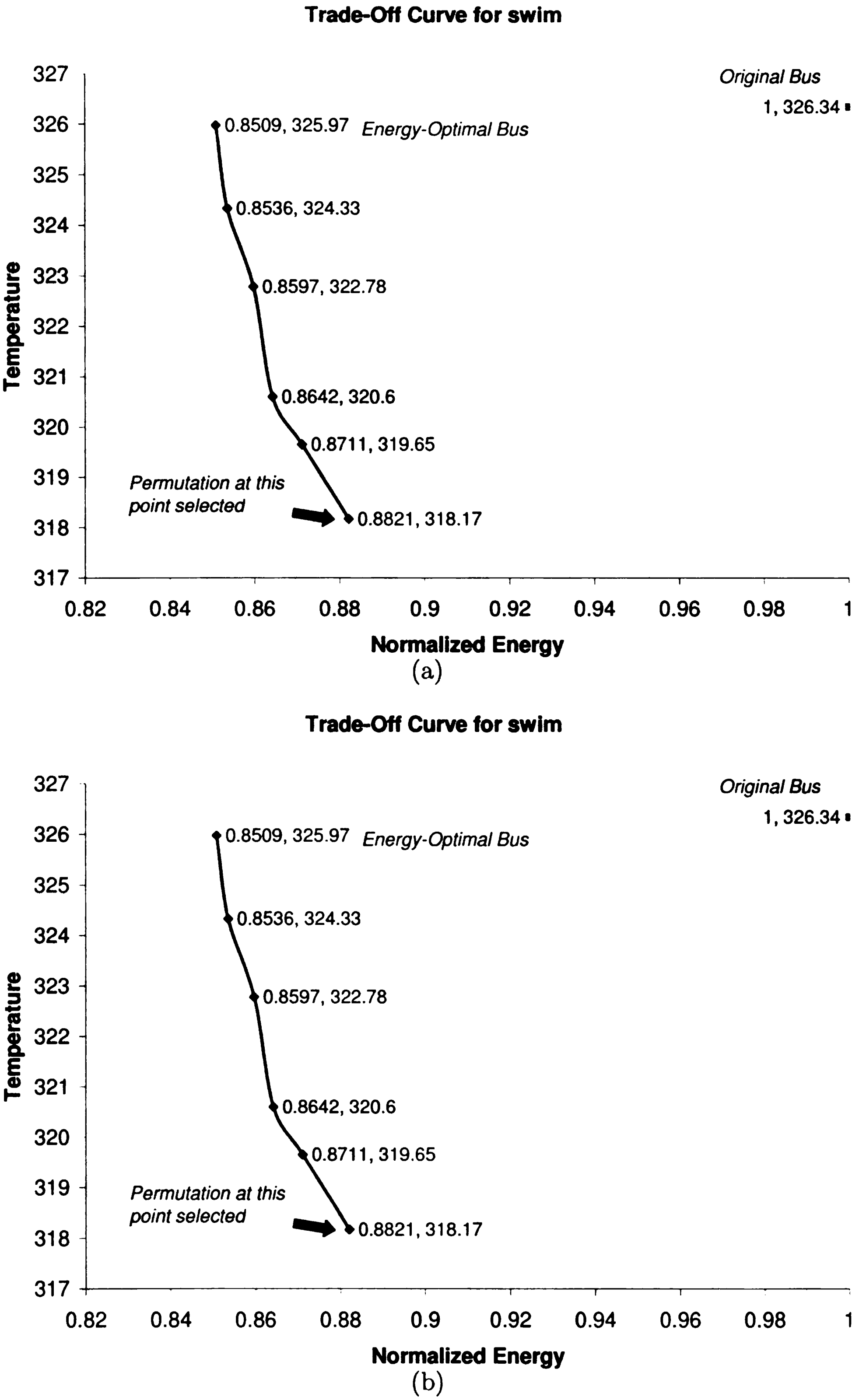
Figure 5.18. Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for gzip and lucas.



(a)

(b)

Figure 5.19. Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for `mesa` and `mgrid`.



(a)

(b)

Figure 5.20. Energy vs. temperature trade-off curves. Plots show the energy vs. temperature tradeoff curves obtained for the data bus for `swim` and `twolf`.

of the energy savings for more thermally-efficient orderings at each step. The steady-state temperature vs. energy tradeoff curves for nine benchmark programs are shown in Figures 5.16–5.20. For each point shown in these graphs, thermal constraints were added and the ILPs were re-solved to get a new wire ordering and permutation. As can be seen, in all the cases the ILP infeasibility occurred before the energy of the reordered bus approached E_{orig} and hence, the optimization terminated. Using these curves, we selected the wire permutation—marked by the arrow in the plots—that resulted in bus energy reduction closest to $0.5(1 - \frac{E_{opt}}{E_{orig}})$, since this represents the midway point for trading off temperature with energy savings. The peak wire temperature obtained for this selected thermally-efficient permutation is shown in the third row of Table 5.5. *Note that the temperatures reported in this row are derived from detailed thermal simulations using the model in Section 3.4 and the not the steady state model.*

Temperature reductions we obtained with temperature-optimized SBOS range from 3.55 to 12.26 degrees for the data bus and from 5.69 to 12.96 degrees for the instruction bus, while still resulting in total energy reductions of 6.59 to 15.23% and 11.67 to 16.17% for data and instruction buses, respectively. Compared to the dynamic spreading encoding technique proposed in [110], our temperature-optimized SBOS provides much better temperature reductions. We compare results for three benchmarks that are common in their work and ours. The temperature reductions they report for the instruction bus are, *gzip*: 6.5 K, *mesa*: 6.25 K, and *ammp*: 4.75 K. Our results shown in Table 5.5 are much better, *gzip*: 15.89 K, *mesa*: 11.67 K, and *ammp*: 12.29 K, for these benchmarks. Note that our techniques are static and incur negligible overhead compared to the overheads for the crossbar switch and control logic used in the spreading encoding technique.

5.6 Summary

In this chapter, we presented a value aware optimization methodology to design static encoding schemes to reduce energy dissipation and temperature of global signal buses. Our methodology examines two aspects: (1) several possible ways of signaling a bit value, with exactly one signaling mode for each bit chosen, and (2) all possible ways of mapping bits to bus lines (*bit ordering* or *permutation*) and then chooses exactly one bit ordering, both statically at design time depending upon traffic value characteristics to minimize total bus dynamic energy. We present an integer linear program (ILP) methodology that evaluates several possible bit signaling modes and all possible bit orderings for an n-bit bus based on traffic value characteristics and then chooses an optimal encoding mode that minimizes total bus (self + coupling) dynamic energy. We use the SimpleScalar/Alpha simulator, profile SimPoint samples of SPEC CPU 2000 benchmarks to collect data, and use the *CPLEX* ILP optimizer design our encoding scheme. Results for three degrees of customization show increasingly better results for average bus energy reduction: *general-purpose optimization*: 20.04% (38.78%), *workload-specific optimization*: 22.79% (40.77%), and *program-specific optimization* 30.2% (52.1%), for 64-bit data (128-bit instruction) buses, respectively. In contrast, existing dynamic bus encoding techniques yield only 4.19% (5.32%) reductions at best for data (instruction) buses for the same set of programs.

We show that lowering bus energy—even significantly, as with our static encoding schemes—does not necessarily lower peak wire temperatures. To address this, we present a novel method of efficiently exploring the peak/hottest wire temperature and total bus dynamic energy trade-off space using a steady-state wire temperature model. Based on this, we present a new method of introducing thermal constraints into our energy optimization methodology that allows a designer to trade-off peak wire temperature with total bus dynamic energy as desired. For this thermally-constrained, energy-optimal encoding scheme, we then perform simulations using a detailed per-wire bus thermal model to determine the actual reductions in peak temperature, which we find to be significant—up to 12.26°C (12.96°C) for data (instruction) buses—while at the same time providing significant average energy savings: 14.24% (16.17%) for data (instruction) buses that are still much better than previous work.

CHAPTER 6

ACTIVITY-AWARE PERFORMANCE OPTIMIZATION

The data-dependent nature of inter-wire crosstalk necessitates bus cycle time to be designed for the worst-case. This pessimistic approach incurs significant performance penalty since the worst case arises least frequently in actual applications. In this chapter, we examine an activity-aware technique that substantially reduces the frequency of worst case crosstalk and improve the bus performance by using a variable cycle bus architecture.

6.1 Introduction

Inter-wire capacitive crosstalk is the primary factor that affects the propagation delay of interconnects. In high-performance processor buses, crosstalk on a victim wire depends on the nature of transitions on its two adjacent wires, known as aggressors. Designers estimate the worst case crosstalk condition for a wire and set the bus clock cycle time greater than this value, ensuring that the signal transmission occurs in the correct manner. However, this is a pessimistic approach since worst case crosstalk conditions do not occur across all wires very frequently.

An introduction to interconnect analysis and the impact of crosstalk on bus design was presented earlier in Section 2.1.5. Table 2.1 listed five different crosstalk conditions based on transitions in the victim and aggressor wires: $1 + 0r$ (*mode-0*), $1 + 1r$

(*mode-1*), $1 + 2r$ (*mode-2*), $1 + 3r$ (*mode-3*), and $1 + 4r$ (*mode-4*), where the coupling ratio r is the ratio of the adjacent coupling capacitance and the line capacitance including the contribution of repeaters. The coupling ratio is greater than unity for nanometer-scale technologies as can be seen from Table 2.2.

We address two aspects of the bus crosstalk problem to improve overall performance of global processor bus in the presence of crosstalk. First, we reduce the frequency of various crosstalk conditions by using a profile-guided wire reordering and signaling approach. Second, we propose a bus clocking approach that eliminates the need to use a pessimistic cycle time. Instead, our approach dynamically controls the number of cycles required for transmission of the data depending on its crosstalk mode. By doing so, we can use the average or most frequent crosstalk pattern to design the cycle time of the bus.

This chapter is organized as follows. Next, Section 6.2 briefly reviews related work. Then, we present our techniques in Section 6.3. Following that, in Section 6.4 we present results. Finally, we summarize in Section 6.5.

6.2 Related Work

Many crosstalk reduction techniques have been proposed in literature. These are reviewed briefly next. Several techniques such as dense wire fabrics [56] and net ordering and shield insertion techniques [118, 119] have been proposed to reduce crosstalk noise in signal interconnects. The effectiveness of shielding and spacing techniques have also been explored [57]. Many coding techniques to reduce crosstalk have also been proposed, all of which rely on using a significant number of extra wires to eliminate worst case crosstalk conditions: crosstalk protection code (CPC) [55], transition pattern code (TPC) [120], crosstalk avoidance code (CAC) [121], and the codes proposed in [122]. A technique that uses variable cycle transmission to improve the bus performance has also been suggested but it does not address crosstalk reduction [123].

6.3 Techniques for Performance Optimization

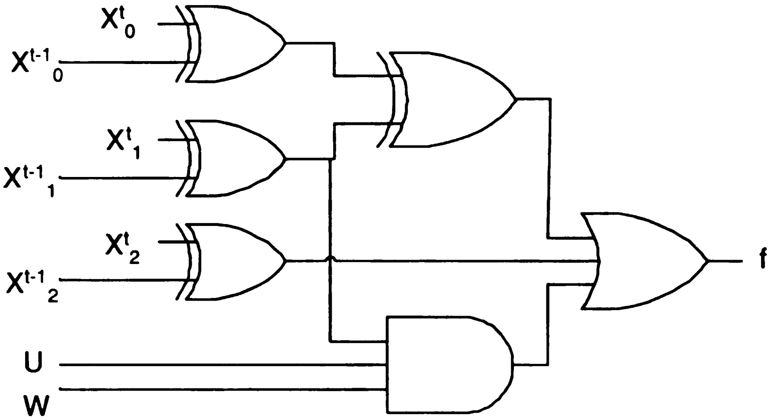
In this section, we describe techniques to optimize bus performance by reducing crosstalk and using a non-pessimistic approach to bus clocking.

6.3.1 Variable Cycle Bus (VCB) Design

We propose an adaptive bus architecture called a *variable cycle bus* (VCB) that uses a faster bus clock and dynamically controls the number of cycles required for transmission based on the estimated delay of the data pattern to be transmitted. This removes the need to design the bus clock cycle in a pessimistic manner based on the worst-case crosstalk pattern. The VCB works as follows. The data to be transmitted in the current cycle is compared to the data that was transmitted in the previous cycle and the *crosstalk group* that it belongs to is determined. There are two groups: a *Group-I* data word is one that has at the most one *mode-2*, *mode-1*, or *mode-0* crosstalk pattern and none higher and a *Group-II* data word is one that has at least one *mode-3* or *mode-4* pattern. The crosstalk group is determined using the crosstalk analyzer (CA) circuit described next. In the VCB, we transmit *Group-I* data in one clock cycle and *Group-II* data using two clock cycles. A *DATA_READY* line indicates to the receiver when to latch the current value being transmitted on the bus. The *DATA_READY* control line is completely shielded, i.e., it is routed with V_{DD}/GND lines on each side so that is completely unaffected by crosstalk.

Inputs: U	W	S_0	S_1	S_2	Output: f
0	0	1	1	1	1
-	-	1	1	0	1
-	-	0	1	1	1

(a)



(b)

Figure 6.1. Three-bit crosstalk analyzer truth table and circuit. (a) Truth table showing only the ON-set. "-" indicates a don't care input. (b) Logic circuit implementing the truth table.

Our *crosstalk analyzer* (CA) circuit identifies the crosstalk mode for each transmission in an efficient manner. It compares the current information, three bits at a time, with corresponding bits in the pattern transmitted in the previous clock cycle and determines if the current pattern falls under one of two crosstalk groups. The way to determine the crosstalk group for a three-bit case is shown next. Consider two three-bit vectors, $X^{t-1} = (X_0^{t-1}, X_1^{t-1}, X_2^{t-1})$ representing the data transmitted in the previous cycle and $X^t = (X_0^t, X_1^t, X_2^t)$ representing data to be transmitted in the current cycle. At the first level of the CA circuit, the following logic outputs are evaluated in parallel:

$$S_0 = X_0^{t-1} \oplus X_0^t, \tag{6.1}$$

$$S_1 = X_1^{t-1} \oplus X_1^t, \tag{6.2}$$

$$S_2 = X_2^{t-1} \oplus X_2^t, \tag{6.3}$$

$$U = X_0^{t-1} \cdot X_1^{t-1} + X_1^{t-1} \cdot X_2^{t-1}, \text{ and} \tag{6.4}$$

$$W = X_0^t \cdot X_1^t + X_1^t \cdot X_2^t. \tag{6.5}$$

Using these signals, the truth table and a gate-level representation of the three-bit CA circuit can be constructed as shown in Figure 6.1. The truth table in Figure 6.1(a) shows only the ON-set of the Boolean function, i.e., the inputs for which the output evaluates to logic "1". The corresponding two-level realization of this table is obtained using Espresso [113]:

$$f = S_0 \cdot \overline{S_1} \cdot S_2 + S_0 \cdot S_1 \cdot \overline{S_2} + \overline{U} \cdot \overline{W} \cdot S_0 \cdot S_2, \tag{6.6}$$

$$= S_0 \cdot (S_1 \oplus S_2 + \overline{U} \cdot \overline{W} \cdot S_2). \tag{6.7}$$

The CA circuit outputs a logic "1" if the three bits it examined result a *Group-II* pattern and logic "0" if not. Thus, for an n-bit bus there are $n-2$ three-bit CA circuts working in parallel to determine the crosstalk group. At the second level, these $n-2$ outputs can be combined using the wired-OR logic style in which outputs from the three-bit CA circuits are simply connected together, as shown in Figure 6.2(a). Thus, the final wired-OR output is high if the output of at least one of the three-bit CA circuits is high. The wired-OR connection is used to simplify the hardware required at the sending end. The signal *DATA_READY* obtained from the bus crosstalk analyzer synchronizes the sender and receiver. When $F = 0$, the data can be transmitted in one cycle and hence *DATA_READY* is taken high. Else, the data is transmitted in two cycles and, in this case, *DATA_READY* is kept low for the first cycle and taken high in the second. The receiver uses a clock signal gated by *DATA_READY* and this ensures that the data is latched and read correctly.

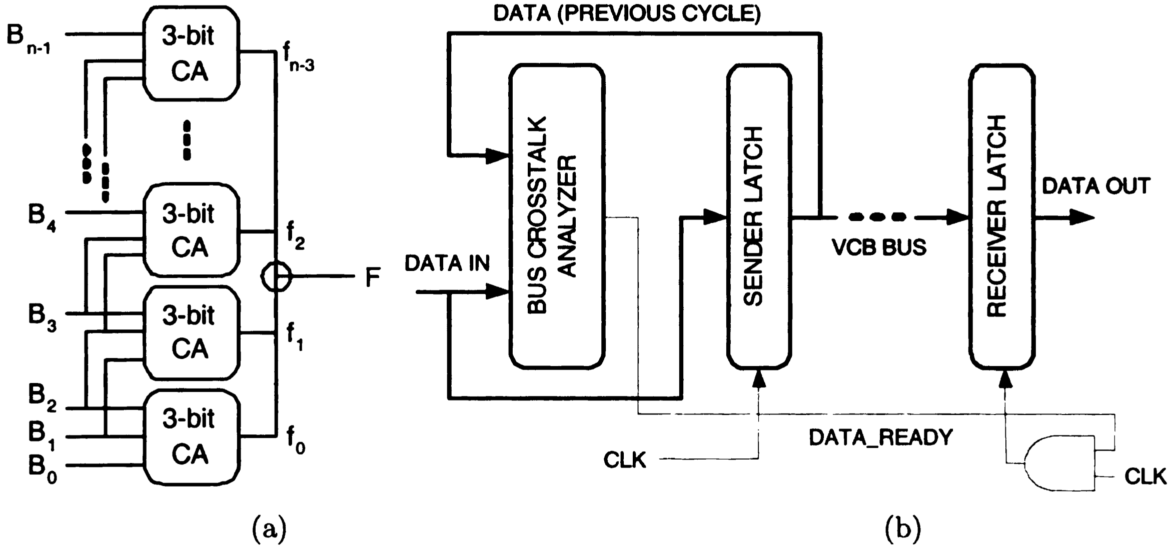


Figure 6.2. Variable cycle bus. (a) Complete bus crosstalk analyzer for an n-bit bus. (b) Sender and receiver logic for VCB.

6.3.2 Minimum Crosstalk Bit Ordering (MCBO)

Our basic technique for profile-guided optimization was discussed earlier in Section 5.4. It may be noted that the objective function that we minimized earlier was the total energy of the bus. In the current problem, we minimize the combined probability of occurrence of the worst-case crosstalk condition for the bus as a whole.

Let Ψ_{2r}, Ψ_{1r}, and Ψ_{0r} be three $n \times n$ *bit-pair crosstalk probability* matrices which record the probability of occurrence of the three crosstalk conditions possible for the bit pair $(i, j), \forall (i, j) \in \{0, n-1\}, i \neq j$: *mode-2*, *mode-1*, and *mode-0*. Note that $\Psi_{2r} + \Psi_{1r} + \Psi_{0r} = J_n$, where J_n is the $n \times n$ unity matrix, since all the probabilities sum up to unity. These matrices are collected by aggregating data obtained by analyzing information patterns transmitted on the target bus when running the training set benchmarks, similar to the procedure outlined in Section 5.3.1.

For three neighboring wires i, j, and k the worst case ($1+4r$ or *mode-4*) crosstalk on the victim wire j occurs when both bit-pairs (i,j) and (j,k) have a *mode-2* crosstalk pattern. Similarly, the next worst case ($1 + 3r$ or *mode-3*) crosstalk occurs when one bit pair has a *mode-2* and the other has a *mode-1* pattern. Both of these situations necessitate transmission in two cycles with our VCB. Let *event* "A" represent the occurrence of *mode-1* or *mode-2* pattern in the first bit-pair (i,j) and event "B" the occurrence of *mode-2* or *mode-1* pattern in the second bit-pair (j,k), i.e., $P(A) = 1-\psi_{0r}[i][j]$ and $P(B) = 1-\psi_{0r}[j][k]$. Note that we use lower-case symbols (ψ) to represent individual elements of the crosstalk matrix Ψ. Since events A and B are mutually exclusive, we have $P(A \text{ or } B) = P(A) + P(B)$. We are interested in obtaining $P(A \text{ or } B)$ since this represents the probability of a *mode-3* or a *mode-4* crosstalk on the bus. Thus, we have: $P(A \text{ or } B) = (1 - \psi_{0r}[i][j]) + (1 - \psi_{0r}[j][k])$.

Following the example above, we combine the bit-pair crosstalk matrices Ψ_{2r}, Ψ_{1r}, and Ψ_{0r}, to get one matrix $\Psi = J_n - \Psi_{0r}$. As noted earlier, our VCB design transmits *mode-4* and *mode-3* patterns in two clock cycles and *mode-2*, *mode-1*, and *mode-0* patterns in one clock cycle. Hence, we seek to minimize the total probability of occurrence of *mode-4* and *mode-3* patterns across all bit-pairs through wire reordering and signaling using integer linear programming. Thus the objective function is the sum of all these probabilities since the events are mutually exclusive and the occurrence of a *mode-4* or *mode-3* event in *any one bit-pair* means that the transmission takes two cycles instead of one. The simple wire reordering formulation, called minimum crosstalk bit ordering (MCBO) using this objective function is discussed next.

As before, the MCBO problem is formulated as an ILP by considering binary variables $x[i][j]$ associated with each bit pair (i, j). In the solution, $x[i][j] = 1$ if bits i and j are to be placed next to each other on the bus and $x[i[j] = 0$, otherwise. Let $V = \{1, \ldots, n\}$ be the vertex set that represents the bits, $A = \{(i, j) : i, j \in V\}$ represent the set of possible triplets of bits, and $\Psi[i][j]$ is the bit-pair crosstalk matrix. The ILP formulation in terms of the variables $x[i][j]$ and the iterative procedure used to solve the ILP is given next:

Step 1 : Minimize $\displaystyle\sum_{\forall (i,j) \in A} \psi[i][j] \cdot x[i][j]$

subject to :

$$x[i][j] \in \{0, 1\}, \forall\, i, j \in V, \tag{6.8}$$

$$\sum_{\forall\, j \in V} x[i][j] = 1, \forall\, i \in V \text{ and } \sum_{\forall\, i \in V} x[i][j] = 1, \forall\, j \in V, \tag{6.9}$$

Step 2 : Solve ILP to get the solution.

Step 3 : Check if the solution has subtours. If none, go to Step 6.

Else, let there be t subtours:

$S = \{S_0(n_0), S_1(n_1), \ldots, S_t(n_t)\}$,

where $S_k(n_k)$ means that subtour S_k has length n_k.

Step 4 : Add subtour elimination constraint:

$$\sum (x[i][j] : \; (i, j) \text{ are in } S_k(n_k)) < n_k, \forall\, S. \tag{6.10}$$

Step 5 : Go to Step 2.

Step 6 : The desired solution (Hamiltonian cycle) has been obtained. Stop.

In the above procedure, Constraint 6.8 ensures that the variables take only binary values and Constraint 6.9 ensures that the in- and out-degrees of every vertex are one, i.e., every bit occurs exactly once in the ordering. As explained in Section 5.4.3, we add subtour eliminations iteratively and solve the ILP efficiently with the *CPLEX* optimizer tool.

6.3.3 MCBO with Signaling (MCBOS)

In MCBO with signaling (MCBOS), the best signaling scheme—one of the five schemes listed in Section 5.4.1—and the appropriate position of the bits on the bus lines is determined simultaneously. As in the case of energy optimization, the motivation for using signaling is to enable the optimizer to select the optimal solution from a richer set of possibilities. Thus, we can view the problem as similar to MCBO but consisting of n *supernodes* corresponding to the n bits of the bus. Each supernode contains five nodes, each representing a signaling scheme choice for a bit. By adding constraints that ensure that only one of these nodes is selected for each supernode and that the incoming and outgoing nodes for each supernode are the same, the ILP for MCBOS is formulated as given next:

$$\text{Minimize} \sum_{\forall(i,j)\in A} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} (\psi_{l,m}[i] \cdot x_{l,m}[i][j]) \right\}$$

subject to :

$$x_{l,m}[i][j] \in \{0,1\}, \forall \{i,j\} \in V, \tag{6.11}$$

$$\sum_{\forall j \in V} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} x_{l,m}[i][j] \right\} = 1, \ \forall i \in V, \tag{6.12}$$

$$\sum_{\forall k \in V} \left\{ \sum_{l=0}^{q-1} \sum_{m=0}^{q-1} x_{l,m}[k][i] \right\} = 1, \ \forall i \in V, \tag{6.13}$$

$$\sum_{l=0}^{q-1} x_{l,m}[i][j] = \sum_{l=0}^{q-1} x_{m,l}[j][k], \forall \{i,j,k\} \in V, \forall m. \tag{6.14}$$

Constraint 6.11 of SBOS ensures that all variables $x_{l,m}[i][j], (l,m) \in \{0,\ldots,q-1\}$, each of which represents a choice of signaling schemes for a pair of bits, take only binary values. Constraints 6.12 and 5.10 ensure that there is only one outgoing and one incoming node, respectively, for each of the n supernodes. Constraints 6.14 ensures that the optimal tour enters and exits through the same node in a supernode (i.e., the signaling schemes chosen for adjacent pairs of bits in the final ordering are consistent). Crosstalk probabilities $\Psi_{l,\,m}[i][i]$, $\forall\, i \in V$ are set to ∞ (a very large integer value). Finally, constraints for eliminating all subtours with two nodes are added initially, and the problem is iteratively solved as described earlier in Section 5.4.3 until a Hamiltonian cycle that visits all supernodes exactly once is found.

6.4 Results and Discussion

We study the effect of MCBO and MCBOS on the 64-bit ALU result bus of our superscalar processor architecture. As explained earlier in Section 5.5, the result bus is on the critical path and is sensitive to delay variations due to crosstalk. Also, the performance of the processor can be improved if faster transmissions are enabled on this bus. We present two results for this bus next: crosstalk reduction using MCBO and MCBOS and performance improvement when VCB is used with MCBO and MCBOS.

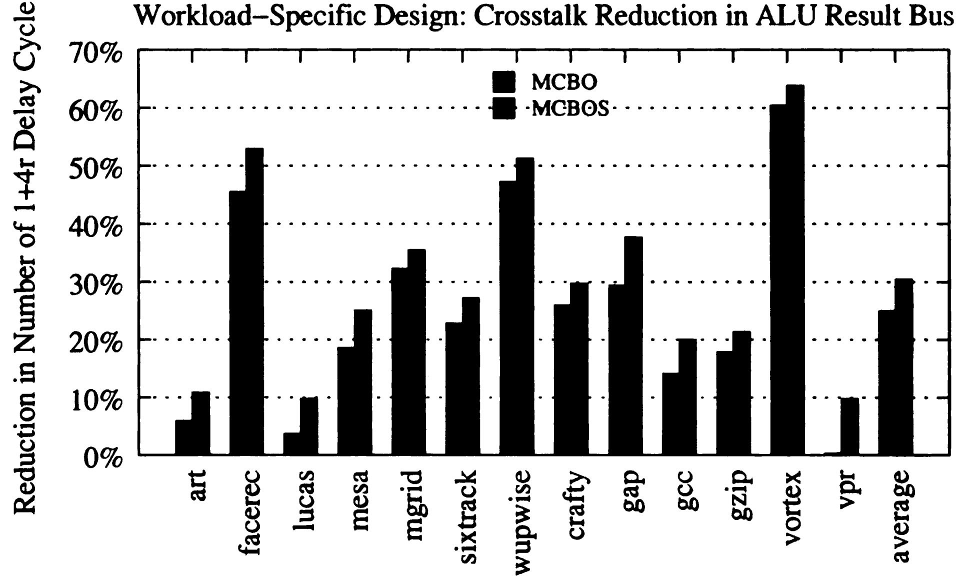
6.4.1 Peak Crosstalk Reduction

In workload-specific design, statistics collected for SimPoint samples from 13 training set benchmarks were aggregated and used to obtain the optimal static encoding schemes. The scheme was then tested on non-overlapping samples from the same set of benchmarks. The non-overlapping sample was selected as explained in Section 5.3.1. As explained earlier, our crosstalk optimization techniques MCBO and MCBOS seek to reduce the number of cycles that carry *mode-4* and *mode-3* patterns. From the results shown in Figures 6.3(a) and (b), we observe that both MCBO and MCBOS reduce *mode-4* and *mode-3* patterns significantly. The average reductions in number of 1+4r delay cycles were MCBO: 24.89% and MCBOS: 30.61% and the average reductions in number of 1+3r cycles were MCBO: 19.21% and MCBOS: 23.42%.

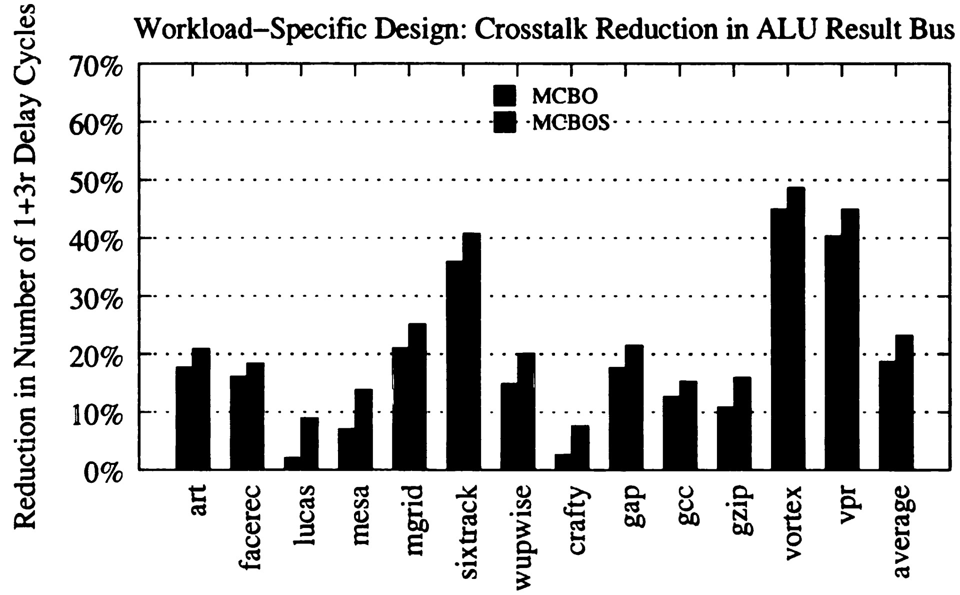
For the general-purpose design scenario, our static schemes were designed using data collected from SimPoint samples for the training benchmarks and then evaluated on test benchmarks. Results are shown in Figures 6.4(a) and (b), for reductions in the number of *mode-4* and *mode-3* cycles, respectively. We observe that the average reductions in number of 1+4r delay cycles were MCBO: 21.22% and MCBOS: 29.35% and the average reductions in number of 1+3r cycles were MCBO: 16.77% and MCBOS: 20.29%.

6.4.2 Performance Improvement with VCB

The reduction in the number of cycles required to transmit the information with our techniques applied is shown in Figure 6.5(a) and (b). On the average, MCBOS which is our best technique reduces the number of cycles by 17.68% for workload-specific optimization and by 18.30% for general purpose optimization while MCBO reduces the number of cycles by 13.89% and 14.44% for workload-specific and general-purpose optimizations, respectively.

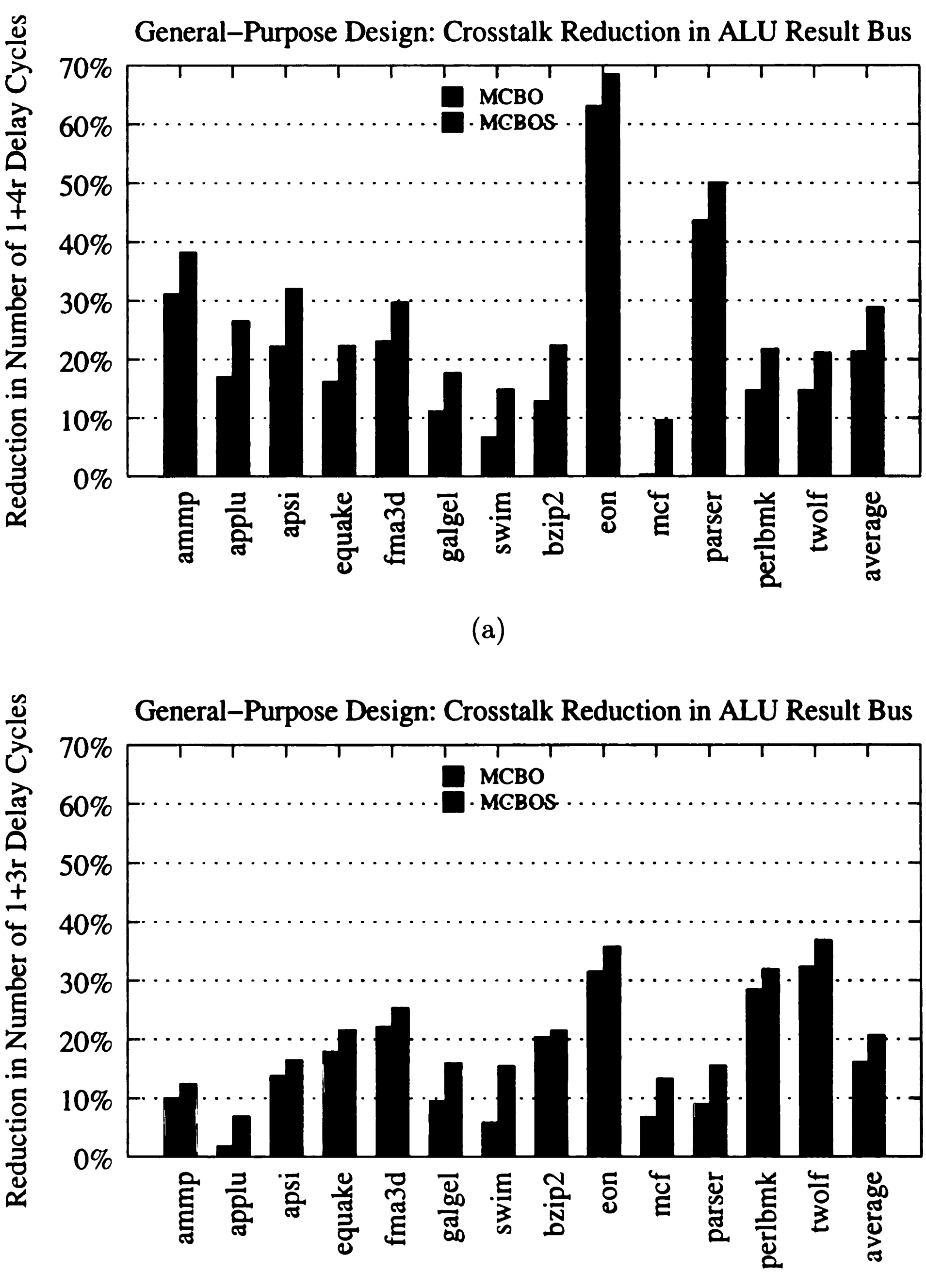


(a)



(b)

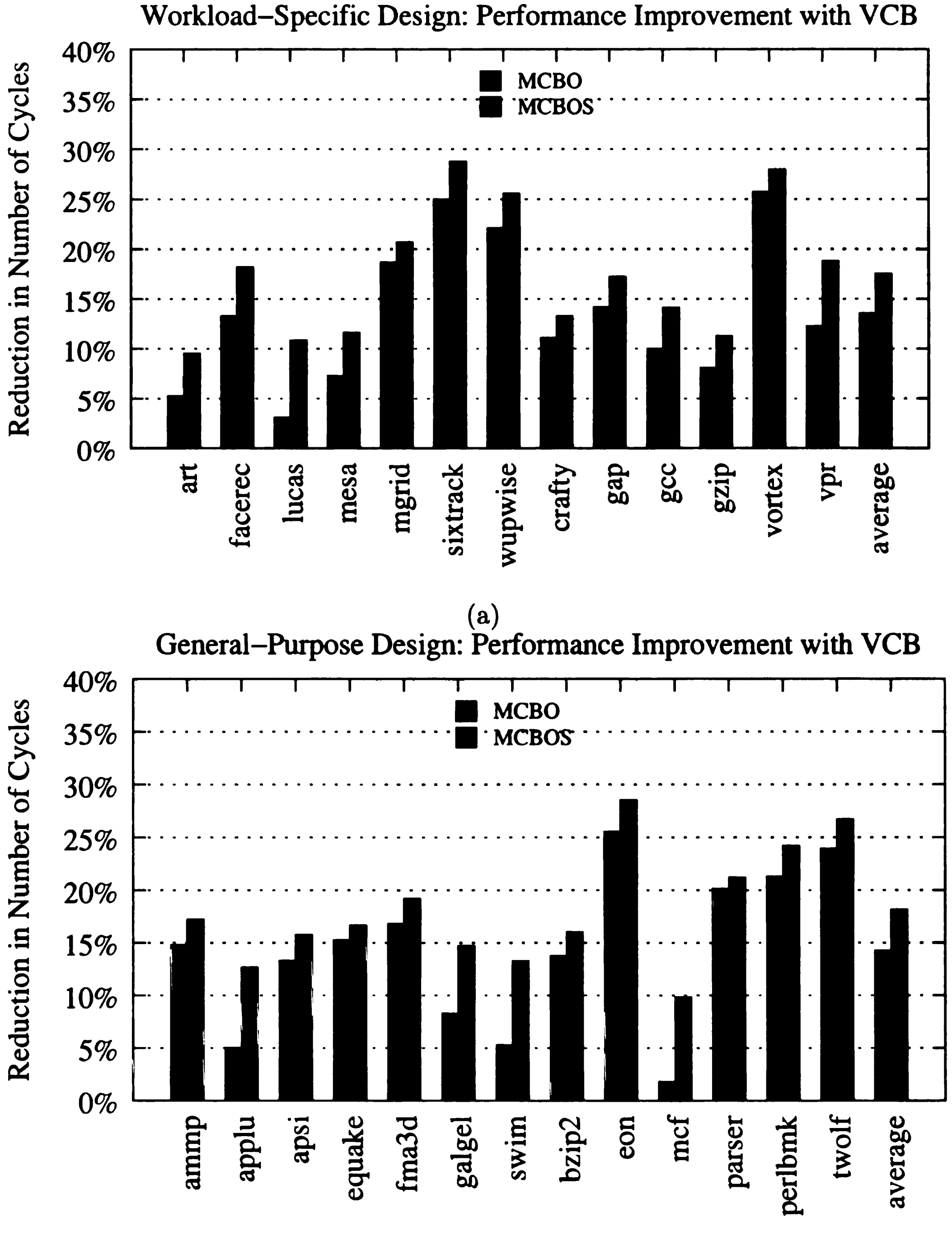
Figure 6.3. Crosstalk reduction results for workload-specific design of the 64-bit ALU result bus. (a) Average reductions in number of 1+4r delay cycles. For MCBO: 24.89% and MCBOS: 30.61%. (b) Average reductions in number of 1+3r cycles. For MCBO: 19.21% and MCBOS: 23.42%.



(a)

(b)

Figure 6.4. Crosstalk reduction results for general purpose design of the 64-bit ALU result bus. (a) Average reductions in number of 1+4r delay cycles. For MCBO: 21.22% and MCBOS: 29.35%. (b) Average reductions in number of 1+3r cycles. For MCBO: 16.77% and MCBOS: 20.29%.



(a)

(b)

Figure 6.5. Reduction in the number of cycles taken to transmit the information with MCBO and MCBOS applied to the result bus. (a) Workload-specific optimization. (b) General-purpose optimization.

6.5 Summary

This chapter presented a performance-oriented adaptive bus design technique that helps reduce the frequency of crosstalk conditions and adopts an adaptive approach to improve bus performance. We presented a variable cycle bus (VCB) architecture and a crosstalk analyzer circuit that can transmit the data using either one or two clock cycles depending on the type of crosstalk pattern. Consequently, the bus clock cycle time no longer needs to be greater than the worst-case (1+4r) crosstalk pattern but it can be designed using the average case or the most frequent (1+2r) crosstalk pattern. We also presented a profile-guided optimization that reduced the frequency of occurrence of 1+4r and 1+3r crosstalk patterns and thus helped improve the performance of the VCB bus significantly. Results on SPEC CPU 2000 benchmarks, in a general-purpose optimization scenario, show a 29.35% reduction in 1+4r cycles, a 20.29% reduction in 1+3r cycles, and a bus performance improvement of 17.42% for a static reordering and signaling technique targeting bus crosstalk minimization.

CHAPTER 7
CONCLUSION

In this dissertation, we presented our research on activity-aware modeling and design optimization for on-chip interconnects in current and future nanometer-scale technologies. We addressed three important issues in high-performance bus design for nanometer-scale microprocessors: accurate energy and thermal modeling, energy optimization techniques, and crosstalk reduction. Key contributions and results from our research are summarized next

7.1 Contributions and Key Results

In Chapter 3, we presented a unified nanometer-scale bus energy dissipation and thermal model that can help designers monitor energy dissipation and temperature change in individual wires during trace- or execution-driven simulation. In addition to self capacitance, our model incorporates the effects of capacitive coupling between adjacent as well as non-adjacent pairs of wires and repeater insertion on switching energy, the effect of lateral heat transfer between adjacent wires to estimate wire temperatures, and also estimates wire temperature gradients and its impact on wire delay, all of which were not available in earlier models.

Using this model, we studied energy and thermal characteristics of instruction and data buses using an execution-driven simulation of a billion or more instructions of nine SPEC CPU2000 benchmarks. We found that existing bus energy models provide estimates that are about 7-8% less accurate compared to our energy model. This is because they do not account for the effects of coupling between non-adjacent wire pairs of a bus. Our model, which incorporates these effects, is the first of its kind to do so. Our results also showed that, in wide instruction and data buses used in modern processors executing SPEC CPU2000 workloads, existing bus encoding schemes show no significant energy benefit due to the nature of data traffic. When non-adjacent coupling effects between wire pairs are considered, energy dissipation savings reduce considerably. Based on simulations using our thermal model, we found that average wire temperatures in data and instruction buses may rise 10-37 °C during a simulation run of only a billion cycles for a 130 nm superscalar processor running SPEC benchmarks. This temperature rise is primarily due to heat generation as a result of currents flowing in the wire during bit switching.

In a future 45 nm technology node, wire temperature rise for the same set of benchmarks and simulation sample was found to be between 20-58°C. We observed that instruction and data bus wires attained absolute temperature in the range 80.3-104°C and 97.6-123.7°C, in 130 nm and 45 nm processors, respectively, during the course of our simulation, showing that signal lines attain significant temperatures too. Significant wire temperature gradients of magnitude between 16-25°C were found to be most common between the sending and receiving ends of the wires during the course of simulation. Notable correlation was found to exist between energy dissipation behavior and wire temperature rise in buses across time; short, intermittent cycles of high energy-dissipating switching activity trigger step changes in temperature.

In Chapter 4, we developed models that track the impact of changing wire temperature on timing/delay violations occurring in global signal buses during microarchitecture-level exploration. Results show that for a 130 nm processor with no power and thermal management the temperature-induced clock cycle time violations in an ALU result bus—which is on the critical path—was 2.27 per hundred bus references, averaged over ten programs in the SPEC CPU2000 workload. It increased to an average of 6.20 per hundred bus references for the same processor at the 45 nm technology node. We found that wire delay variability led to degradation in overall performance by about 4.1% in 130 nm processors and about 11.9% in 45 nm processors. Our analysis also showed that conventional techniques like bus encoding that seek to reduce energy dissipation and potentially wire temperatures have limited impact on alleviating temperature-induced delay violations.

In Chapter 5, we formulated an optimization methodology to design energy and temperature optimized static bus encoding schemes through early stage microarchitecture-level exploration, exploiting value characteristics of a target workload. Binary integer linear programs (ILPs) were formulated and solved optimally to determine the signaling, bit ordering, or a combination of both that minimizes bus energy dissipation. For the SPEC CPU2K workload, our static bit ordering and signaling (SBOS) technique reduced total bus energy dissipation by 22.79%/40.77% for data/instruction buses in an application-specific scenario, where the technique was designed individually using statistics collected for each benchmark and tested on the same benchmark. In a much more general scenario, where the scheme was designed using statistics collected from 13 out of 26 benchmarks and tested on the remaining 13, the corresponding reductions were 20.04%/38.78%. These reductions are significantly higher compared to those obtained from dynamic encoding schemes for the same benchmarks. We also proposed a first-of-its-kind methodology to design temperature-aware encoding schemes by trading off some of the energy gains we obtain with static encoding techniques to achieve wire temperature reduction. In this methodology we add temperature constraints during energy optimization, and our ILP produces a static encoding scheme that reduces maximum/hottest wire temperatures by up to 15.23 K/16.17 K for data/instruction buses while still producing significant total bus energy reductions.

Finally, in Chapter 6, we examined techniques to reduce bus crosstalk and improve overall bus performance. We presented a variable cycle bus (VCB) architecture and a crosstalk analyzer circuit that can transmit the data using either one or two clock cycles depending on the type of crosstalk pattern. Consequently, the bus clock cycle time no longer needs to be greater than the worst-case crosstalk pattern but it can be designed using the average case or the most frequent crosstalk pattern which results in roughly doubling the bus clock frequency. We also presented a profile-guided optimization that reduced the frequency of occurrence of worst-case crosstalk patterns and thus helped improve the performance of the VCB bus significantly. Results on SPEC CPU 2000 benchmarks show at least 29.35% reduction in number of worst case crosstalk cycles and a bus performance improvement of 17.42% for a VCB with static reordering and signaling technique targeting bus crosstalk minimization.

Our work represents a significant advancement over existing approaches that are activity-oblivious and/or consider worst-case traffic conditions. The microarchitecture-level activity-driven spatiotemporal bus energy and thermal model we present is the first of its kind. Our static value-aware bit reordering and signaling techniques are also highly-novel solutions that work remarkably well in real applications.

7.2 Directions for Future Research

Some potential research directions for the future are outlined next.

- A methodology to dynamically select between different static wire orderings and signaling strategies for energy and/or thermal optimization can be investigated. In such a scheme, a controller will select a particular strategy based on input or hints from the compiler through data stored in the program's executable.

- The wire ordering and signaling strategies can be used to create configurable *interconnect intellectual property* (IIP) blocks similar to configurable IP blocks available today for logic circuits. Such an IIP block will contain routing specification for all on-chip high-performance signals between logic blocks, suitably optimized for power, temperature, crosstalk, or a combination of the tree, automatically synthesized by a CAD tool by analyzing the user-supplied workload.

- The thermal model can be enhanced to investigate thermal issues in clock trees and a temperature-aware clock-tree synthesis approach can be developed. The thermal model can also be used as a starting point for analyzing issues related to three-dimensional interconnects. In such systems, the presence of multiple vertically connected interconnect stacks emphasizes the need to investigate thermal issues, since heat dissipation paths from interconnect layers may be several times longer than conventional designs.

BIBLIOGRAPHY

[1] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors (ITRS), 2005 edition," URL: http://public.itrs.net.

[2] M. Mui, K. Banerjee, and A. Mehrotra, "A Global Interconnect Optimization Scheme for Nanometer Scale VLSI with Implications for Latency, Bandwidth, and Power Dissipation," *IEEE Transactions on Electron Devices*, vol. 51, no. 3, pp. 195–203, Feb. 2004.

[3] S. Rusu, "Circuit Technologies for Multi-Core Design," Talk at the *IEEE Santa Clara Valley Solid-State Circuits Society*, slides at: http://www.ewh.ieee.org/r6/scv/ssc/April06.pdf, Apr. 2006.

[4] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-Power Dissipation in a Microprocessor," in *Proceedings of the 2004 International Workshop on System level Interconnect Prediction (SLIP'04).* New York, NY, USA: ACM Press, 2004, pp. 7–13.

[5] S. Im and K. Banerjee, "Full Chip Thermal Analysis of Planar (2-D) and Vertically Integrated (3-D) High Performance ICs," in *Proceedings of the IEEE International Electron Devices Meeting (IEDM).* Piscataway, NJ, USA: IEEE Press, Dec. 2000, pp. 727–730.

[6] P. Gelsinger, "Microprocessors for the New Millennium: Challenges, Opportunities and New Frontiers," in *Proceedings of the IEEE Solid-State and Circuits Conference (ISSCC).* Piscataway, NJ, USA: IEEE Press, Dec. 2001, pp. 22–25.

[7] K. Nabors, S. Kim, J. White, and S. Senturia, "Fast Capacitance Extraction of General Three-Dimensional Structures," in *Proceedings of International Conference on Computer Design (ICCD).* Washington DC, USA: IEEE Computer Society, Oct. 1991, pp. 479–484.

[8] M. Bohr, "Interconnect Scaling: The Real Limiter to High Performance ULSI," in *Proceedings of the International Electron Devices Meeting (IEDM).* Piscataway, NJ, USA: IEEE Press, Dec. 1995, pp. 241–244.

[9] L. Lev and P. Chao, "Down to the Wire: Requirements for Nanometer Design Implementation," White Paper, Cadence Design Systems Inc., 2002.

[10] W. Li, B. Mbouombouo, and L. Tsai, "Needed: High-Level Interconnect Methodology for Nanometer ICs," *EE Times*, http://www.eetimes.com/story/OEG20030623S0039, June 2003.

[11] P. Green, "A GHz IA-32 Architecture Microprocessor Implemented on 0.18μm Technology with Aluminum Interconnect," in *Proceedings of the IEEE Solid-State and Circuits Conference (ISSCC).* Piscataway, NJ, USA: IEEE Press, 2000, pp. 98–99.

[12] J. Heidenreich, D. Edelstein, R. Goldblatt, W. Cote, C. Uzoh, N. Lustig, T. McDevitt, A. Stamper, A. Simon, J. Dukovic, P. Andriacacos, R. Washnik, H. Rathore, T. Katsetos, P. McLaughlin, S. Luce, and J. Slattery, "Copper Dual Damascene for sub-0.25μm CMOS," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 1998, pp. 151–153.

[13] B. Zhao, D. Feiler, V. Ramanathan, Q. Liu, M. Brongoa, J. Wu, H. Zhang, J. Kuei, and D. Young, "A Cu Low-k Dual Damascene Interconnect for High-Performance and Low Cost Integrated Circuits," in *Proceedings of the IEEE Symposium on VLSI Technology.* Piscataway, NJ, USA: IEEE Press, June 1998, pp. 28–29.

[14] P. Zarkesh-Ha, J. Davis, and J. Meindl, "The Impact of Cu/Low-k on Chip Performance," in *Proceedings of the IEEE International ASIC/SoC Conference.* Piscataway, NJ, USA: IEEE Press, 1999, pp. 257–261.

[15] H. Feng, F. Ercal, and F. Bunyak, "Systolic Algorithm for Processing RLE Images," in *IEEE Southwest Symposium on Image Analysis and Interpretation.* Piscataway, NJ, USA: IEEE Press, 1998, pp. 127–131.

[16] S. Chai, A. Gentile, W. Lugo-Beauchamp, J. Fonseca, J. Cruz-Rivera, and D. Wills, "Focal Plane Processing Architectures for Real-Time Hyperspectral Image Processing," *Applied Optics: Special Issue on Optics in Computing*, vol. 39, pp. 835–849, Feb. 2000.

[17] W. Dally, "Interconnect-limited VLSI architecture," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 1999, pp. 15–17.

[18] J. Goodman, R. Kostuk, and B. Clymer, "Optical Interconnects: An Overview," in *Proceedings of the 2^{nd} International IEEE VLSI Multilevel Interconnection Conference.* Piscataway, NJ, USA: IEEE Press, 1985, pp. 219–224.

[19] A. Rahman, A. Fan, J. Chung, and R. Reif, "Wire-Length Distribution of Three-Dimensional Integrated Circuits," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 1999, pp. 233–235.

[20] S. Souri and K. Saraswat, "Interconnect Performance Modeling for 3D Integrated Circuits with Multiple Si Layers," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 1999, pp. 24–26.

[21] K. Banerjee, "Trends for ULSI Interconnections and Their Implications for Thermal, Reliability and Performance Issues (Invited Paper)," in *Proceedings of the Seventh International Dielectrics and Conductors for ULSI Multilevel Interconnection Conference (DCMIC).* Tampa, FL, USA: IMIC, Mar. 2001, pp. 38–50.

[22] W. Dally and J. Poulton, *Digital Systems Engineering.* Cambridge University Press, 1998.

[23] A. Krishnamoorthy and D. Miller, "Scaling Optoelectronic-VLSI Circuits into the 21st century: A Technology Roadmap," *IEEE Journal on Selected Topics in Quantum Electronics*, vol. 2, no. 1, pp. 55–76, 1996.

[24] T. Mule, S. Schultz, T. Gaylord, and J. Meindl, "An Optical Clock Distribution Network for Gigascale Integration," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 2000, pp. 176–179.

[25] J. Joyner and J. Meindl, "Opportunities for Reduced Power Dissipation Using Three-Dimensional Integration," in *Proceedings of the IEEE International Interconnect Technology Conference.* Piscataway, NJ, USA: IEEE Press, June 2002, pp. 148–150.

[26] J. Joyner, P. Zarkesh-Ha, J. Davis, and J. Meindl, "Vertical Pitch Limitations on Performance Enhancement in Bonded Three-Dimensional Interconnect Architectures," in *Proceedings of the International Workshop on System-Level Interconnect Prediction.* New York, NY, USA: ACM Press, 2000, pp. 123–127.

[27] K. Saraswat, S. Souri, K. Banerjee, and P. Kapur, "Performance Analysis and Technology of 3-D ICs," in *Proceedings of the International Workshop on System-Level Interconnect Prediction.* New York, NY, USA: ACM Press, Apr. 2000, pp. 85–90.

[28] A. Shilov, "Intel to Cancel NetBurst, Pentium 4, Xeon Evolution: Tejas, Jayhawk Reportedly Shelved," *X-Bit Laboratories*, http://www.xbitlabs.com/news/cpu/display/20040507000306.html, May 2004.

[29] J. Kovar, "Sun Cancels UltraSPARC V, Gemini, But Not Future Processor Development," *CMP Media's CRN*, http://www.crn.com/sections/breakingnews/dailyarchives.jhtml?articleId=18841521, Apr. 2004.

[30] K. Krewell, "Multicore Mania is Here to Stay," *Electronic Design News (EDN)*, http://www.edn.com/article/CA6302185.html?partner=eb&pubdate=2%2F1%2F2006, Feb. 2006.

[31] D. Brooks, M. Martonosi, J. Wellman, and P. Bose, "Power-Performance Modeling and Tradeoff Analysis for a High End Microprocessor," in *Proceedings of the First International Workshop on Power-Aware Computer Systems (PACS'00) held with ASPLOS-IX*, Nov. 2000.

[32] J. Cong, "An Interconnect-Centric Design Flow for Nanometer Technologies," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 505–528, April 2001.

[33] V. Agarwal, M. Hrishikesh, S. Keckler, and D. Burger, "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures," in *Proceedings of the Annual Symposium on Computer Architecture (ISCA)*. New York, NY, USA: ACM Press, July 2000, pp. 248–259.

[34] R. Ho, K. Mai, and M. Horowitz, "The Future of Wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[35] T. N. Vijaykumar and Z. Chishti, "Wire Delay is Not a Problem for SMT (In the Near Future)," in *Proceedings of the Annual Symposium on Computer Architecture (ISCA)*. Washington, DC, USA: IEEE Computer Society, July 2004, pp. 40–50.

[36] J. Meindl, J. Davis, P. Zarkesh-Ha, C. Patel, K. Martin, and P. Kohl, "Interconnect Opportunities for Gigascale Integration," *IBM Journal of Research and Development*, vol. 46, no. 2, pp. 245–263, Mar. 2002.

[37] K. Banerjee and A. Mehrotra, "Global Interconnect Warming," *IEEE Circuits and Devices*, vol. 17, pp. 16–32, Sept. 2001.

[38] A. Ajami, K. Banerjee, and M. Pedram, "Modeling and Analysis of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects," *IEEE Transactions on Computer Aided Design of Integrated Circuits and systems*, vol. 24, no. 6, pp. 849–860, June 2005.

[39] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach, Third Edition.* Morgan Kaufmann Publishers, 2003.

[40] J. Shen and M. Lipasti, *Modern Processor Design: Fundamentals of Superscalar Processors.* McGraw Hill, 2004.

[41] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[42] J. Davis, V. De, and J. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration–Part I: Derivation and Validation," *IEEE Transactions on Electron Devices*, vol. 45, no. 3, pp. 580–589, Mar. 1998.

[43] K. Banerjee and A. Mehrotra, "Analysis of On-Chip Inductive Effects for Distributed RLC Interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 8, pp. 904–915, Aug. 2002.

[44] R. Kumar, "Interconnect and Noise Immunity Design for the Pentium 4 Processor," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC).* New York, NY, USA: ACM Press, 2003, pp. 938–943.

[45] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits, Second Edition.* Prentice-Hall, Dec. 2002.

[46] A. Naeemi, R. Venkatesan, and J. D. Meindl, "Optimal Global Interconnects for GSI," *IEEE Transactions on Electron Devices*, vol. 50, no. 4, pp. 980–987, Apr. 2003.

[47] M. Stan and W. Burleson, "Low-Power Encodings for Global Communication in CMOS VLSI," *IEEE Transactions on VLSI Systems*, vol. 5, no. 4, pp. 444–455, Dec. 1997.

[48] J. Liu, N. Mahapatra, and K. Sundaresan, "Hardware-Only Compression to Reduce Cost and Improve Utilization of Address Buses," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI).* Los Alamitos, CA, USA: IEEE Computer Society, Feb. 2003, pp. 220–221.

[49] J. Liu, K. Sundaresan, and N. Mahapatra, "Energy-Efficient Compressed Address Transmission," in *Proceedings of the 18th International Conference on VLSI Design (VLSID).* Washington, DC, USA: IEEE Computer Society, Jan. 2005, pp. 592–597.

[50] ——, "Fast Perfomance-Optimized Partial Match Compression for Low-Latency On-Chip Address Buses," in *Proceedings of International Conference on Computer Design (ICCD).* Piscataway, NJ, USA: IEEE Press, Oct. 2006.

[51] M. Stan and W. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[52] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization," in *Proceedings of Conference on Design Automation and Test in Europe (DATE)*. Washington, DC, USA: IEEE Computer Society, Feb. 1998.

[53] Y. Zhang, J. Lach, K. Skadron, and M. Stan, "Odd/Even Bus Invert with Two-Phase Transfer for Buses with Coupling," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*. New York, NY, USA: ACM Press, Aug. 2002, pp. 80–83.

[54] K. Kim, K. Back, N. Shanbhag, C. Liu, and S. Kang, "Coupling-Driven Signal Encoding Scheme for Low-Power Interface Design," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. Washington, DC, USA: IEEE Computer Society, Nov. 2000, pp. 318–321.

[55] B. Victor and K. Keutzer, "Bus Encoding to Prevent Crosstalk Delay," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. Piscataway, NJ, USA: IEEE Press, Nov. 2001, pp. 57–63.

[56] S. Khatri, A. Mehrotra, R. Brayton, R. Otten, and A. Sangiovanni-Vincentelli, "A Novel VLSI Layout Fabric for Deep Sub-Micron Applications," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC)*. New York, NY, USA: ACM Press, 1999, pp. 491–496.

[57] R. Arunachalam, E. Acar, and S. Nassif, "Optimal Shielding/Spacing Metrics for Low Power Design," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. Washington DC, USA: IEEE Computer Society, Feb. 2003, pp. 167–171.

[58] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proceedings of the Annual Symposium on Computer Architecture (ISCA)*. New York, NY, USA: ACM Press, 2000, pp. 83–94.

[59] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC)*. New York, NY, USA: ACM Press, June 2000, pp. 340–345.

[60] A. Dhodapkar, C. Lim, G. Cai, and W. Daasch, "TEM^{2}P^{2}EST: A Thermal Enabled Multi-model Power/Performance ESTimator," in *Lecture Notes In*

Computer Science, Proceedings of the First International Workshop on Power-Aware Computer Systems (PACS'00) held with ASPLOS-IX, November, 2000. Springer-Verlag, 2001, pp. 112–125.

[61] J. Smith, L. He, A. Dhodapkar, and N. Nidhi, "WArPE: Wisconsin Architecture Power Estimator," URL: http://eda.ee.ucla.edu/ntool/.

[62] The Sim-Panalyzer Team, "SimpleScalar-ARM Power Modeling Project," URL: http://www.eecs.umich.edu/~panalyzer/.

[63] D. Ponomarev, G. Kukuk, and K. Ghose, "AccuPower: An Accurate Power Estimation Tool for Superscalar Microprocessors," in *Proceedings of Conference on Design Automation and Test in Europe (DATE).* Washington, DC, USA: IEEE Computer Society, Mar. 2002, pp. 124–128.

[64] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," in *Proceedings of the Annual Symposium on Computer Architecture (ISCA).* New York, NY, USA: ACM Press, June 2003, pp. 2–13.

[65] Y. Zhang, R. Chen, W. Ye, and M. Irwin, "System Level Interconnect Power Modeling," in *Proceedings of the IEEE ASIC/SoC Conference.* Piscataway, NY, USA: IEEE Press, Sept. 1998, pp. 289–293.

[66] W. Huang, M. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy, "Compact Thermal Modeling for Temperature-Aware Design," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC).* New York, NY, USA: ACM Press, June 2004, pp. 878–883.

[67] D. Burger and T. Austin, "The SimpleScalar Tool Set, version 2.0," *Computer Architecture News*, vol. 25, no. 5, pp. 13–25, June 1997.

[68] Michigan State University High Performance Computing Center, "128 Node Opteron Cluster from Western Scientific," https://hpc.msu.edu/twiki/bin/view/Main/WesternScientificCluster.

[69] T.-Y. Chiang, K. Banerjee, and K. Saraswat, "Compact Modeling and SPICE-Based Simulation for Electrothermal Analysis of Multilevel ULSI Inteconnects," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD).* Washington, DC, USA: IEEE Computer Society, Nov. 2001, pp. 165–172.

[70] T.-Y. Wang and C.-P. Chen, "SPICE-Compatible Thermal Simulation with Lumped Circuit Modeling for Thermal Reliability Analysis based on Modeling

Order Reduction," in *Proceedings of International Symposium on Quality of Electronics Design (ISQED)*, 2004.

[71] L. Shang, L.-S. Peh, A. Kumar, and N. K. Jha, "Thermal Modeling, Characterization and Management of On-Chip Networks," in *Proceedings of the Annual ACM/IEEE International Symposium on Microarchitecture (MICRO).* Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2004, pp. 67–78.

[72] K. Banerjee, A. Mehrotra, A. Sangiovanni-Vincentelli, and C. Hu, "On Thermal Effects in Deep Sub-Micron VLSI Interconnects," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC).* New York, NY, USA: ACM Press, 1999, pp. 885–891.

[73] D. Chen, E. Li, E. Rosenbaum, and S. Kang, "Interconnect Thermal Modeling for Accurate Simulation of Circuit Timing and Reliability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 197–205, Feb. 2000.

[74] R. Desikan, D. Burger, S. Keckler, and T. Austin, "Sim-alpha: A Validated, Execution-Driven Alpha 21264 Simulator," The University of Texas at Austin, Department of Computer Sciences, Tech. Rep. TR-01-23, 2001.

[75] Standard Performance Evaluation Council, "CPU2000 Version 1.2," http://www.spec.org/cpu2000, 2001.

[76] SimpleScalar LLC, http://www.simplescalar.com.

[77] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Systems (ASPLOS).* New York, NY, USA: ACM Press, Oct. 2002, pp. 45–57.

[78] ——, "SimPoint Single Simulation Points for SPEC CPU 2000," URL: http://www.cse.ucsd.edu/~calder/simpoint/single-sim-pionts.htm.

[79] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "SimPoint 3.0: Faster and More Flexible Program Analysis," *The Journal of Instruction-Level Parallelism*, vol. 7, 2005, http://www.jilp.org/vol7/v7paper14.pdf.

[80] K. Sundaresan and N. Mahapatra, "An Accurate Energy and Thermal Model for Global Signal Buses," in *Proceedings of the 18th International Conference on VLSI Design.* Washington DC, USA: IEEE Computer Society, Jan. 2005, pp. 685–690.

[81] ——, "Accurate Energy Dissipation and Thermal Modeling for Nanometer-Scale Signal Buses," in *Proceedings of International Symposium on High Performance Computer Architecture (HPCA)*. Washington DC, USA: IEEE Computer Society, Feb. 2005, pp. 51–60.

[82] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul.–Aug. 1999.

[83] T.-Y. Chiang and K. Saraswat, "Closed-Form Analytical Thermal Model for Accurate Temperature Estimation of Multilevel ULSI Interconnects," in *2003 Symposium on VLSI Circuits Digest of Papers*. Piscataway, NJ, USA: IEEE Press, June 2003, pp. 275–279.

[84] K. Banerjee and A. Mehrotra, "Coupled Analysis of Electromigration Reliability and Performance in ULSI Signal Nets," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. Washington, DC, USA: IEEE Computer Society, Nov. 2001, pp. 158–164.

[85] P. Sotiriadis and A. Chandrakasan, "A Bus Energy Model for Deep Submicron Technology," *IEEE Transactions on VLSI Systems*, vol. 10, no. 3, pp. 341–350, June 2002.

[86] W.-C. Cheng and M. Pedram, "Memory Bus Encoding for Low-Power: A Tutorial," in *Proceedings of International Symposium on Quality of Electronics Design (ISQED)*. Washington, DC, USA: IEEE Computer Society, Mar. 2001.

[87] P. Sotiriadis and A. Chandrakasan, "Low Power Bus Coding Techniques Considering Inter-wire Capacitances," in *Proceedings of Custom Integrated Circuits Conference (CICC)*. Washington DC, USA: IEEE Computer Society, May 2000, pp. 414–419.

[88] H. Deogun, R. Rao, D. Sylvester, and D. Blaauw, "Leakage- and Crosstalk-Aware Bus Encoding for Total Power Reduction," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC)*. New York, NY, USA: ACM Press, June 2004, pp. 779–782.

[89] N. Menezes and L. Pillegi, "Analyzing On-Chip Interconnect Effects," in *Design of High-Performance Microprocessor Circuits*, A. Chandrakasan, W. Bowhill, and F. Fox, Eds. Piscataway, NJ, USA: IEEE Press, 2000, pp. 331–351.

[90] A. Kahng, K. Masuko, and S. Muddu, "Analytical Delay Models for VLSI Interconnects under Ramp Input," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. Washington, DC, USA: IEEE Computer Society, Nov. 1996, pp. 30–36.

[91] J. Srinivasan and S. Adve, "The Importance of Heat-Sink Modeling for DTM and a Correction to Predictive DTM for Multimedia Applications," In *Proceedings of the Fourth Annual Workshop on Duplicating, Deconstructing, and Debunking*, Madison, WI, USA, June 2005.

[92] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing.* New York, NY, USA: Cambridge University Press, 1992.

[93] R. Chandra, "Impact of Thermal Analysis on Large Chip Design," *Electronic Design Process Symposium (EPDS 2005)* talk slides, URL: http://www.gradient-da.com/pdf/EDP_for_website.pdf, 2005.

[94] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The Case for Lifetime Reliability-Aware Microprocessors," in *Proceedings of the Annual Symposium on Computer Architecture (ISCA).* Washington, DC, USA: IEEE Computer Society, June 2004, pp. 276–286.

[95] E. W. Weisstein, "Geometric Centroid," From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/GeometricCentroid.html.

[96] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula, "Variational Delay Metrics for Interconnect Timing Analysis," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC).* New York, NY, USA: ACM Press, 2004, pp. 381–384.

[97] P. Bose, "Power- and Reliability-Aware (Integrated) Design: Challenges and Opportunities," Talk slides URL: ee.usc.edu/news/dls/talks/bose_presentation.pdf, Oct. 2005.

[98] K. Sundaresan and N. Mahapatra, "Value-Based Bit Ordering for Energy Optimization of On-Chip Global Signal Buses," in *Proceedings of Conference on Design Automation and Test in Europe (DATE).* Leuven, Belgium: European Design and Automation Association, Mar. 2006, pp. 624–625.

[99] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, "Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD).* Washington, DC, USA: IEEE Computer Society, Nov. 2004, pp. 327–334.

[100] Q. Zhou and K. Mohanram, "Elmore Model for Energy Estimation in RC Trees," in *Proceedings of the Annual ACM/IEEE Design Automation Conference (DAC).* New York, NY, USA: ACM Press, July 2006, pp. 965–970.

[101] S. Ramprasad, N. Shanbhag, and I. Hajj, "Information-Theoretic Bounds on Average Signal Transition Activity," *IEEE Transactions on VLSI Systems*, vol. 7, no. 3, pp. 359–368, Sept. 1999.

[102] R.-B. Lin and C.-M. Tsai, "Theoretical Analysis of Bus-Invert Coding," *IEEE Transactions on VLSI Systems*, vol. 10, no. 6, pp. 929–935, Dec. 2002.

[103] Y. Shin and K. Choi, "Narrow Bus Encoding for Low Power Systems," in *Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC)*. New York, NY, USA: ACM Press, Jan. 2000, pp. 217–220.

[104] P. Sotiriadis and A. Chandrakasan, "Bus Energy Minimization by Transition Pattern Coding (TPC) Using a Detailed Deep Sub-Micron Bus Model," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. Washington, DC, USA: IEEE Computer Society, Nov. 2001, pp. 322–328.

[105] L. Macchiarulo, E. Macii, and M. Poncino, "Low-Energy Encoding for Deep-Submicron Address Buses," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*. New York, NY, USA: ACM Press, 2001, pp. 176–181.

[106] ——, "Wire Placement for Crosstalk Energy Minimization in Address Buses," in *Proceedings of Conference on Design Automation and Test in Europe (DATE)*. Washington, DC, USA: IEEE Computer Society, Mar. 2002, pp. 158–162.

[107] E. Macii, M. Poncino, and S. Salerno, "Combining Wire Swapping and Spacing for Low-Power Deep-Submicron Buses," in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*. New York, NY, USA: ACM Press, 2003, pp. 198–202.

[108] E. Naroska, S.-J. Ruan, and U. Schwiegelshohn, "An Efficient Algorithm for Simultaneous Wire Permutation, Inversion, and Spacing," in *Proceedings of International Symposium on Circuits and Systems (ISCAS)*. Piscataway, NJ, USA: IEEE Press, May 2005, pp. 109–112.

[109] L. Deng and M. Wong, "Energy Optimization in Memory Address Bus Structure for Application-Specific Systems," in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*. New York, NY, USA: ACM Press, Apr. 2005, pp. 232–237.

[110] F.Wang, Y. Xie, N. Vijaykrishnan, and M. Irwin, "On-Chip Bus Analysis and Optimization," in *Proceedings of Conference on Design Automation and Test in Europe (DATE)*. Leuven, Belgium: European Design and Automation Association, Mar. 2006, pp. 850–855.

[111] ILOG, Inc., "CPLEX 9.0 ," http://www.ilog.com/products/cplex, 2003.

[112] R. Kumar, “Interconnect and Noise Immunity Design for the Pentium 4 Processor,” *Intel Technology Journal, 1st Quarter*, vol. Q1, 2001.

[113] Berkeley Espresso minimization tool, “Web version,” http://embsys.technikum-wien.at/espresso/html/espresso.html.

[114] P. Groeneveld, “Wire Ordering for Detailed Routing,” *IEEE Design and Test*, vol. 6, no. 6, pp. 6–17, 1989.

[115] M. Marek-Sadowska and M. Sarrafzadeh, “The Crossing Distribution Problem,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 4, pp. 423–433, Apr. 1995.

[116] X. Song and Y. Wang, “On the Crossing Distribution Problem,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 4, no. 1, pp. 39–51, 1999.

[117] D. Knuth, *The Art of Computer Programming.* Reading, MA: Addison-Wesley Longman, 1973.

[118] L. He and K. Lepak, “Simultaneous Shield Insertion and Net Ordering for Capacitive and Inductive Coupling Minimization,” in *Proceedings of the International Conference on Computer-Aided Design (ICCAD).* Los Alamitos, CA, USA: IEEE Computer Society Press, 2000, pp. 55–60.

[119] J. Ma and L. He, “Formulae and Applications of Interconnect Estimation Considering Shield Insertion and Net Ordering,” in *Proceedings of the International Conference on Computer-Aided Design (ICCAD).* Piscataway, NJ, USA: IEEE Press, 2001, pp. 327–332.

[120] P. Sotiriadis and A. Chandrakasan, “Reducing Bus Delay in Sub-Micron Technology Using Coding,” in *Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC).* New York, NY, USA: ACM Press, Jan. 2001, pp. 109–114.

[121] S. Sridhara, A. Ahmed, and N. Shanbhag, “Area and Energy-Efficient Crosstalk Avoidance Codes for On-Chip Buses,” in *Proceedings of International Conference on Computer Design (ICCD).* Washington, DC, USA: IEEE Computer Society, Oct. 2004, pp. 12–17.

[122] C. Duan and S. Khatri, “Exploiting Crosstalk to Speed up On-Chip Buses,” in *Proceedings of Conference on Design Automation and Test in Europe (DATE).* Washington, DC, USA: IEEE Computer Society, 2004, pp. 20 778–20 782.

[123] L. Li, N. Vijaykrishnan, M. Kandemir, and M. Irwin, “A Crosstalk Aware Interconnect with Variable Cycle Transmission,” in *Proceedings of Conference on Design Automation and Test in Europe (DATE)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 10 102–10 106.