

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/267775712>

Low Power Design Automation

Thesis · December 2006

CITATIONS

7

READS

107

1 author:



David Chinnery

Mentor Graphics, Fremont

32 PUBLICATIONS 628 CITATIONS

SEE PROFILE

Low Power Design Automation

by

David Graeme Chinnery

Bachelor of Science with First Class Honours
(University of Western Australia) 1998

Bachelor of Engineering with First Class Honours
(University of Western Australia) 1998

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Kurt Keutzer, Chair
Professor Borivoje Nikolić
Professor Alper Atamturk

Fall 2006

The dissertation of David Graeme Chinnery is approved:

Chair

Date

Date

Date

University of California, Berkeley

Fall 2006

Low Power Design Automation

Copyright 2006

by David Graeme Chinnery

Abstract

Low Power Design Automation

by

David Graeme Chinnery

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kurt Keutzer, Chair

We investigate the differences in power between application-specific integrated circuits (ASICs), designed in an automated design methodology, and custom integrated circuits with examples from 0.6um to 0.13um CMOS. The ASICs dissipate 3 to 7 \times more power than custom integrated circuits. We quantify factors contributing to the power gap with analytical models and empirical data. We discuss the shortcomings of typical synthesis flows, and the changes to tools and standard cell libraries that are necessary to reduce power.

The most significant opportunity for power reduction in ASICs is using microarchitectural techniques to maintain performance while reducing power via voltage scaling. By using high performance and low power techniques that can be integrated within an EDA methodology, we believe that the power gap between ASIC and custom designs may be closed to within 2.6 \times . One such approach is to use high Vdd and low Vth on critical paths, and low Vdd and high Vth elsewhere to reduce the power.

We formulate optimization of Vdd, Vth, and gate sizes as a convex geometric program with posynomial models of gate delay and power. Dual Vdd and dual Vth provide up to 18% power saving versus optimal voltage scaling with single Vdd and single Vth. The geometric programming runtime grows cubically with circuit size, making it computationally infeasible for circuits of an interesting size to designers.

Existing circuit sizing heuristics that proceed in the manner of TILOS [64] are fast, having quadratic runtime growth, but can be suboptimal. We develop a gate sizing approach using linear programming (LP) which achieves on average 16.3% power savings versus Design Compiler, a commercial synthesis tool. Our approach has up to quadratic runtime growth, making it scalable to larger circuit sizes.

The dual-Vdd/dual-Vth/sizing results with the LP approach average 5% to 13% lower power than the best alternate approaches that we know of for this problem [124][125][188]. We find that dual Vth reduces power by 16% on average and up to 26% versus using a single Vth; whereas dual Vdd only reduces power by 4% on average and up to 14% versus using a single Vdd.

Professor Kurt Keutzer, Chair of Committee

Contents

<i>Acknowledgements</i>	<i>vii</i>
<i>Chapter 1. Introduction</i>	<i>I</i>
1.1 Definitions: ASIC and custom	1
1.2 What is a standard cell ASIC methodology?.....	4
1.3 Who should care?.....	8
1.3.1 ASIC and ASSP designers seeking high performance.....	8
1.3.2 ASIC and ASSP designers seeking lower power.....	10
1.3.3 Custom designers seeking higher productivity	11
1.4 Organization of the rest of the thesis.....	12
1.5 What's not in this thesis	13
<i>Chapter 2. Overview of the Factors Affecting the Power Consumption</i>	<i>14</i>
2.1 Introduction.....	14
2.2 Process technology independent FO4 delay metric	15
2.3 Components of power consumption	17
2.3.1 Dynamic power	18
2.3.2 Leakage power	18
2.4 ASIC and custom power comparison.....	19
2.4.1 ARM processors from 0.6 to 0.13um.....	21
2.4.2 Comparison of IDCT/DCT cores	23
2.5 Factors contributing to ASICs being higher power than custom.....	24
2.5.1 Combined impact of the contributing factors.....	26
2.5.2 Microarchitecture	28
2.5.3 Clock gating	31
2.5.4 Power gating and other techniques to reduce leakage in standby	34
2.5.5 Logic style	39
2.5.6 Logic design	42
2.5.7 Technology mapping.....	43
2.5.8 Gate sizing and wire sizing	45
2.5.9 Voltage scaling.....	49
2.5.10 Floorplanning, cell placement and wire routing.....	53
2.5.11 Process technology.....	55
2.5.12 Process variation	60
2.6 Conclusions.....	64
<i>Chapter 3. Pipelining to Reduce Power</i>	<i>66</i>
3.1 Introduction.....	68
3.1.1 Power and performance metrics.....	70
3.1.2 Parallel datapath model.....	72
3.1.3 Pipeline model.....	73

3.2 Pipelining overheads.....	76
3.2.1 Timing overhead per pipeline stage for ASIC and custom designs	79
3.2.2 Pipeline imbalance in ASIC and custom designs.....	80
3.2.3 Instructions per cycle versus number of pipeline stages.....	81
3.2.4 Power overheads for pipelining	83
3.3 Pipelining power and delay model.....	84
3.3.1 Pipeline stage delay.....	84
3.3.2 Utilizing slack for voltage scaling and downsizing to reduce power.....	85
3.3.3 Power consumption of the registers and clock tree.....	88
3.3.4 The pipeline power and delay model for optimization	91
3.4 ASIC versus custom pipelining.....	95
3.4.1 Maximum performance (minimum delay/instruction).....	98
3.4.2 Maximum BIPS ^m /W with voltage scaling and gate sizing	100
3.4.3 Minimum power for a given performance constraint.....	102
3.5 Other factors affecting the power gap between ASIC and custom	106
3.5.1 Combinational logic delay	106
3.5.2 Clock gating	107
3.5.3 CPI penalty.....	108
3.5.4 Leakage power	108
3.5.5 Register and clock tree power	109
3.5.6 Gate sizing and voltage scaling.....	110
3.6 Other factors affecting the minimum energy per operation.....	113
3.6.1 Glitching in complementary static CMOS logic	113
3.6.2 Additional power overheads	121
3.7 Summary.....	125
3.7.1 Comparison to related research.....	127
Chapter 4. Linear Programming for Gate Sizing.....	128
4.1 Introduction.....	129
4.1.1 Gate sizing approaches.....	130
4.2 Overview of TILOS gate sizing.....	133
4.3 Initial work on a linear programming formulation.....	136
4.3.1 The initial version of the linear program.....	137
4.3.2 Delay and power inaccuracy	141
4.4 A power and delay accurate linear programming formulation.....	142
4.4.1 What variables must be modeled in the linear program for accuracy?	144
4.4.2 Modeling the impact of signal slew propagation	146
4.4.3 Formulating cell changes	149
4.4.4 Input drivers	153
4.4.5 The linear program.....	154
4.5 Optimization Flow	158
4.5.1 The importance of the delay reduction phase	160
4.6 Comparison of gate sizing power minimization results.....	163
4.6.1 Benchmarks.....	165

4.6.2	Comparison versus Design Compiler.....	166
4.6.3	Post-pass cleanup with Design Compiler.....	168
4.6.4	Comparison versus integer linear programming	169
4.7	Computational runtime.....	171
4.7.1	Theoretical runtime complexity and analysis of actual runtimes.....	171
4.7.2	Runtime comparison versus other optimization approaches.....	177
4.7.3	Memory requirements	178
4.8	Summary.....	178
<i>Chapter 5. Voltage Scaling.....</i>		181
5.1	Effect of supply voltage, threshold voltage and gate size on power and delay	181
5.1.1	Switching power	182
5.1.2	Short circuit power.....	183
5.1.3	Leakage power	185
5.1.4	Delay	186
5.2	Delay and power for 0.13um libraries versus supply and threshold voltage.....	189
5.2.1	Delay dependence on Vdd and Vth for the 0.13um library	190
5.2.2	Switching power dependence on Vdd and Vth for the 0.13um library	193
5.2.3	Internal power dependence on Vdd and Vth for the 0.13um library	194
5.2.4	Leakage dependence on Vdd and Vth for the 0.13um library	197
5.2.5	Overall dependence of power on Vdd and Vth for the 0.13um library	199
5.2.6	Optimal supply and threshold voltages to minimize total power.....	200
5.3	Summary.....	205
<i>Chapter 6. Geometric Programming for Optimal Assignment of PMOS and NMOS Widths, Threshold Voltages, and Gate Supply Voltage.....</i>		208
6.1	Introduction.....	209
6.2	Posynomial Models	212
6.2.1	Data Sampling.....	215
6.2.2	Posynomial Fits.....	220
6.3	Constraints	227
6.3.1	Process Technology Constraints	227
6.3.2	Graph Edge Constraints	230
6.3.3	Maximum delay constraint and total power.....	233
6.4	Issues with Discrete Voltages	234
6.4.1	A Discretization Heuristic.....	236
6.5	Results	243
6.5.1	Benchmarks.....	243
6.5.2	Single voltage, dual voltage, and continuous voltage solutions.....	246
6.6	Computational Runtime.....	257
6.7	Conclusions.....	261
<i>Chapter 7. Linear Programming for Threshold Voltage and Supply Voltage Assignment with Gate Sizing</i>		267

7.1	Introduction.....	267
7.2	Voltage level restoration for multi-Vdd.....	272
7.3	Summary of previous research on optimization with multi-Vdd and multi-Vth....	275
7.3.1	Summary of papers on optimization with multi-Vth	276
7.3.2	Summary of papers on optimization with multi-Vdd.....	278
7.3.3	Summary of papers on optimization with multi-Vth and multi-Vdd	279
7.4	Optimizing with multiple supply voltages and multiple threshold voltages.....	281
7.4.1	The linear program formulation with multi-Vth and multi-Vdd.....	282
7.4.2	Voltage level converter power and delay overheads.....	283
7.4.3	Climbing the optimization barrier posed by level converter power overheads.....	287
7.4.4	Climbing the optimization barrier posed by level converter delay overheads	290
7.5	Comparison of multi-Vdd and multi-Vth results versus University of Michigan...	292
7.6	Analysis of power savings with multiple Vth and multiple Vdd.....	297
7.6.1	General experimental conditions.....	297
7.6.2	Experimental conditions for multi-Vth comparison.....	298
7.6.3	Experimental conditions for multi-Vdd comparison.....	300
7.6.4	Results with 80ps level converter flip-flop delay overhead	302
7.6.5	Results with 0ps level converter flip-flop delay overhead	310
7.7	Computational runtimes with multi-Vdd and multi-Vth.....	318
7.8	Summary.....	320
7.8.1	Detailed commentary	321
Chapter 8.	Conclusions	326
8.1	Analysis of the power gap between ASIC and custom	326
8.1.1	Power and delay pipeline model results.....	326
8.1.2	Summary of methods to close the power gap	327
8.2	Optimization of supply voltage, threshold voltage, and gate sizes	328
8.2.1	Circuit power and delay model accuracy	329
8.2.2	Geometric program optimization of Vdd, Vth, and gate sizes	330
8.2.3	Linear program assignment for Vdd, Vth, and gate sizes	332
8.3	Summary.....	334
Appendix A	Pipeline model details	336
A.1	Characteristics of ASIC and custom processors	336
A.2	Modeling register and clock tree power in different processors.....	340
A.3	Correcting Harstein and Puzak's pipeline power model	341
A.4	Finding T/T_{\min} for minimum energy per operation.....	345
A.5	Graphs of BIPS^m/W with voltage scaling and gate sizing.....	347
A.6	Dynamic and leakage power fits for the pipeline model.....	347
A.6.1	Fits for voltage scaling with T_{\min} of 0.9	347
A.6.2	Fits for gate sizing.....	348
A.7	Power consumption due to glitching	349

A.7.1	Glitching model.....	350
A.7.2	Glitching portion of dynamic power fit for a FPGA 32-bit multiplier.....	351
A.7.3	Dynamic power including glitching fit for a FPGA 64-bit multiplier	353
A.7.4	Comments on glitching power versus pipeline depth in FPGAs	355
A.7.5	Incorporating glitching power into the pipeline power model	356
A.8	Graphs of optimal clock period with additional power overheads	357
<i>Appendix B Checking the Delay and Power Characterization of the 0.13um Libraries</i>		359
B.1	Delay dependence on Vdd and Vth for the 0.13um library	359
B.2	Switching power dependence on Vdd and Vth for the 0.13um library.....	362
B.3	Internal power dependence on Vdd and Vth for the 0.13um library.....	364
B.4	Leakage dependence on Vdd and Vth for the 0.13um library.....	365
B.5	Summary of corrections to the 0.13um library	369
<i>Appendix C Complexity of Power Minimization with Gate Sizing</i>		372
C.1	Proof of NP-Completeness of Power Minimization with Gate Sizing.....	373
<i>Appendix D Geometric Programming</i>		380
D.1	Additional details for the data sampling.....	380
D.2	Simplifying inverter delay fits vs. input slew and input voltage.....	385
D.3	Posynomial fits	390
D.3.1	Posynomial fits with a reduced number of parameters	391
D.3.2	Conservative posynomial fits	394
D.4	Formulating constraints for the geometric program solver.....	395
D.5	Voltage discretization for c432 at a delay constraint of 2.12ns.....	396
D.6	Geometric programming result details.....	400
D.6.1	c17	401
D.6.2	c432nr	406
D.6.3	Power versus delay curves for c17 and c432nr with fixed global voltage values.	411
D.6.4	Power versus delay curves for c17 and c432nr without wire loads	411
D.6.5	c880.....	411
<i>Appendix E Static timing and power analysis</i>		416
E.1	Static Timing Analysis.....	416
E.2	Nonlinear delay interpolation between table lookup grid points.....	418
E.3	Calculating the timing envelopes on outputs.....	421
E.4	Static power analysis	424
E.5	Wire delays and wire capacitance (wire load model)	426
<i>Appendix F Design Compiler and VCS Scripts.....</i>		428
<i>Appendix G Details of the Linear Programming Approach</i>		431

G.1	Pseudo-code for trying different optimization parameters.....	431
G.2	Impact of excluding slew analysis on linear programming results.....	433
G.3	Problems with delay reduction at a tight delay constraint.....	434
G.4	Comparison versus TILOS	435
G.4.1	Accuracy of University of Michigan static timing and power analysis	437
G.4.2	Linear programming sizing results versus TILOS-like optimizer.....	441
G.5	Linear programming sizing results with different parameter settings.....	442
<i>Appendix H Survey of previous research on optimization with multi-Vdd and multi-Vth</i>		448
H.1	Papers on optimization with multi-Vth.....	448
H.2	Papers on optimization with multi-Vdd.....	460
H.3	Papers on optimization with multi-Vth and multi-Vdd	470
<i>Appendix I Multi-Vdd and multi-Vth with gate sizing results vs. University of Michigan...</i>		477
I.1	Results at $1.1 \times T_{\min}$, delay scaling by Equation (5.10) and α of 1.101	478
I.2	Results at $1.1 \times$, $1.2 \times$ and $1.5 \times T_{\min}$, delay scaling by Equation (5.9) and α of 1.3	480
<i>Appendix J Results for Multi-Vth and Multi-Vdd Analysis</i>		486
J.1	Multi-Vth/Vdd=1.2V with 1% leakage at $1.0 \times T_{\min}$.....	487
J.2	Multi-Vth/Vdd=1.2V with 8% leakage at $1.0 \times T_{\min}$	487
J.3	Multi-Vth/single Vdd with 1% leakage at $1.2 \times T_{\min}$ with 80ps LCFFs	488
J.4	Multi-Vdd/single Vth with 1% leakage at $1.2 \times T_{\min}$ with 80ps LCFFs	490
J.5	Multi-Vdd/multi-Vth with 1% leakage at $1.2 \times T_{\min}$ with 80ps LCFFs	492
J.6	Vdd=0.8V/multi-Vth results with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs.....	494
J.7	Multi-Vdd/single Vth with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs	494
J.8	Multi-Vdd/multi-Vth with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs	496
J.9	Computation runtimes with multi-Vdd and multi-Vth	504
<i>References</i>		506

Acknowledgements

Many people have given me advice, feedback and support over the years. I will endeavor to acknowledge the majority of those people here, but there are also numerous others with whom I have discussed research and who have made helpful suggestions. Foremost, I am grateful for the help and guidance from my advisor, Kurt Keutzer.

The Semiconductor Research Corporation supported this research on low power. For those searching the SRC website for details, the task title was “Algorithmic and Circuit-Level Approaches to Leveraging Multiple Threshold Voltage Processes” and the task identification number was 915.001. This research was expanded to include consideration of multiple supply voltages. My thanks to STMicroelectronics for access to their 0.13um process technology and to the contacts at STMicroelectronics, Bhusan Gupta and Ernesto Perea. For the algorithmic portion of this research, I collaborated extensively with David Blaauw, Sarvesh Kulkarni, Ashish Srivastava, and Dennis Sylvester. Sarvesh Kulkarni and Ashish Srivastava provided two characterized drive strengths of the strength 5 asynchronous level converter [123] and Synopsys PowerArc characterized libraries for STMicroelectronics 0.13um process. I would like to thank the Intel industrial liaisons, in particular Vijay Pitchumani and Desmond Kirkpatrick, for their advice.

I would like to thank researchers at the Berkeley Wireless Research Center: Stephanie Augsburger, Rhett Davis, Sohrab Emami-Neyestanak, Borivoje Nikolić, Fujio Ishihara, Dejan Markovic, Brian Richards, Farhana Sheikh, and Radu Zlatanovici. Laurent El Ghaoui also helped with convex optimization research.

I would like to acknowledge the contributors to sessions on Closing the Gap between ASIC and Custom and the two books on the topic. Fruitful discussions with them have helped me clarify some of my assumptions and delve in the details of the subject: Ameya Agnihotri, Debashis

Bhattacharya, Subhrajit Bhattacharya, Vamsi Boppana, Andrew Chang, Pinhong Chen, John Cohn, Michel Cote, Michel Courtoy, Wayne Dai, William Dally, David Flynn, Jerry Frenkil, Eliot Gerstner, Ricardo Gonzalez, Razak Hossain, Lun Bin Huang, Bill Huffman, Philippe Hurat, Anand Iyer, Srikanth Jadcherla, Michael Keating, Earl Killian, George Kuo, Yuji Kukimoto, Julian Lewis, Pong-Fei Lu, Patrick Madden, Murari Mani, Borivoje Nikolić, Greg Northrop, Satoshi Ono, Michael Orshansky, Barry Pangrle, Matthew Parker, Ruchir Puri, Stephen Rich, Nick Richardson, Jagesh Sanghavi, Kaushik Sheth, Jim Schwartz, Naresh Soni, David Staepelaere, Leon Stok, Xiaoping Tang, Chin-Chi Teng, Srini Venkatraman, Radu Zlatanovici, and Tommy Zounes.

I would also like to thank authors of other publications that I have been involved in: Abhijit Davare, Farzan Fallah, David Hathaway, Masayuki Ito, Matthew Moskewicz, David Nguyen, Kaushik Ravindran, Nadathur Satish, Serdar Tasiran, Brandon Thompson, and Scott Weber.

A number of other people within the UC Berkeley EECS CAD group have helped with research, editing, and camaraderie over the years: Luca Carloni, Matthias Gries, Benjamin Horowitz, Heloise Hse, Yujia Jin, Chidamber Kulkarni, Trevor Meyerowitz, Andrew Mihal, Alessandra Nardi, Will Plishker, Christian Sauer, Niraj Shah, Michael Shilman, Mark Spiller, and Kees Vissers.

Lastly, I would like to thank my friends and family for their support. Special thanks to my wife, Eleyda Negron, for her help editing my dissertation, for her patience and for her support.

Chapter 1. Introduction

This dissertation examines the power consumption of ASIC and custom integrated-circuits. In particular, we examine the relationship between custom circuits designed without any significant restriction in design methodology and ASIC circuits designed in a high-productivity EDA tool methodology. From analysis of similar ASIC and custom designs, we estimate that the power consumption of typical ASICs may be $\times 3$ to $\times 7$ that of custom ICs fabricated in process technology of the same generation. We consider ways to augment and enhance an ASIC methodology to bridge the power gap between ASIC and custom.

Reducing circuit power consumption has been a hot topic for some time; however, there has not been detailed analysis of the power gap between an automated design methodology and custom design, where manual design at a lower design level can achieve lower power. This work gives a quantitative analysis of the factors contributing to the power gap. By identifying the largest contributing factors, and which of these can be automated, we aim to help close the power gap. Voltage scaling is potentially one of the most significant factors contributing to the power gap. Thus in the latter portion of this thesis, we focus on automating the use of lower supply voltages and higher threshold voltages.

1.1 Definitions: ASIC and custom

The term *application-specific integrated-circuit* (ASIC), has a wide variety of associations. Strictly speaking, it simply refers to an *integrated circuit* (IC) that has been designed for a particular *application*. This defines a portion of the semiconductor market. Other market segments include memories, microprocessors, and field programmable gate arrays (FPGAs).

Two industries grew to support the development of ASICs: vendors fabricating chips, and companies offering electronic design automation (EDA) software. The ASIC semiconductor-vendor industry, established by companies such as LSI Logic, provides the service of fabricating

ASICs designed by other independent design groups. EDA companies such as Cadence and Synopsys provide commercial tools for designing these ASICs. Another key element of the ASIC design process is *ASIC libraries*. ASIC libraries are carefully characterized descriptions of the primitive logic-level building blocks provided by the ASIC vendors. Initially these libraries targeted gate-array implementations, but in time the higher-performance standard-cell targets became more popular.

ASIC vendors then offered complete design flows for their fabrication process. These consisted of ASIC tools, ASIC libraries for the process, and a particular design methodology. These embodied an *ASIC methodology* and were known as *ASIC design kits*. Smith's book on ASICs [185] is a great one-stop reference for ASICs.

Generally, ASICs are designed at the register-transfer level (RTL) in Verilog or VHDL, specifying the flow of data between registers and the state to store in registers. Commercial EDA tools are used to map the higher level RTL description to standard cells in an ASIC library, and then place the cells and route wires. It is much easier to migrate ASIC designs to a new process technology, compared to custom designs which have been optimized for a specific process at the gate or transistor-level. ASIC designers generally focus on high level designs choices, at the microarchitectural level for example.

With this broader context, let us pause to note that the use of the term ASIC can be misleading: it most often refers to an IC produced through a standard cell ASIC methodology and fabricated by an ASIC vendor. That IC may belong to the application-specific standard product (ASSP) portion of the semiconductor market. ASSPs are sold to many different system vendors [185], and often may be purchased as standard parts from a catalog, unlike ASICs.

The term *custom integrated-circuit*, or *custom IC*, also has a variety of associations, but it principally means a circuit produced through a custom-design methodology. More generally,

custom IC is used synonymously with the semiconductor market segments of high-performance microprocessors and digital signal processors.

Custom ICs are typically optimized for a specific process technology and take significantly more time to design than ASICs, but can achieve higher performance and lower power by higher quality design and use of techniques that are not generally available to ASICs. For example, custom designers may design logic gates at the transistor-level to provide implementations that are optimal for that specific design; whereas an ASIC designer is limited by what is available in the standard cell library.

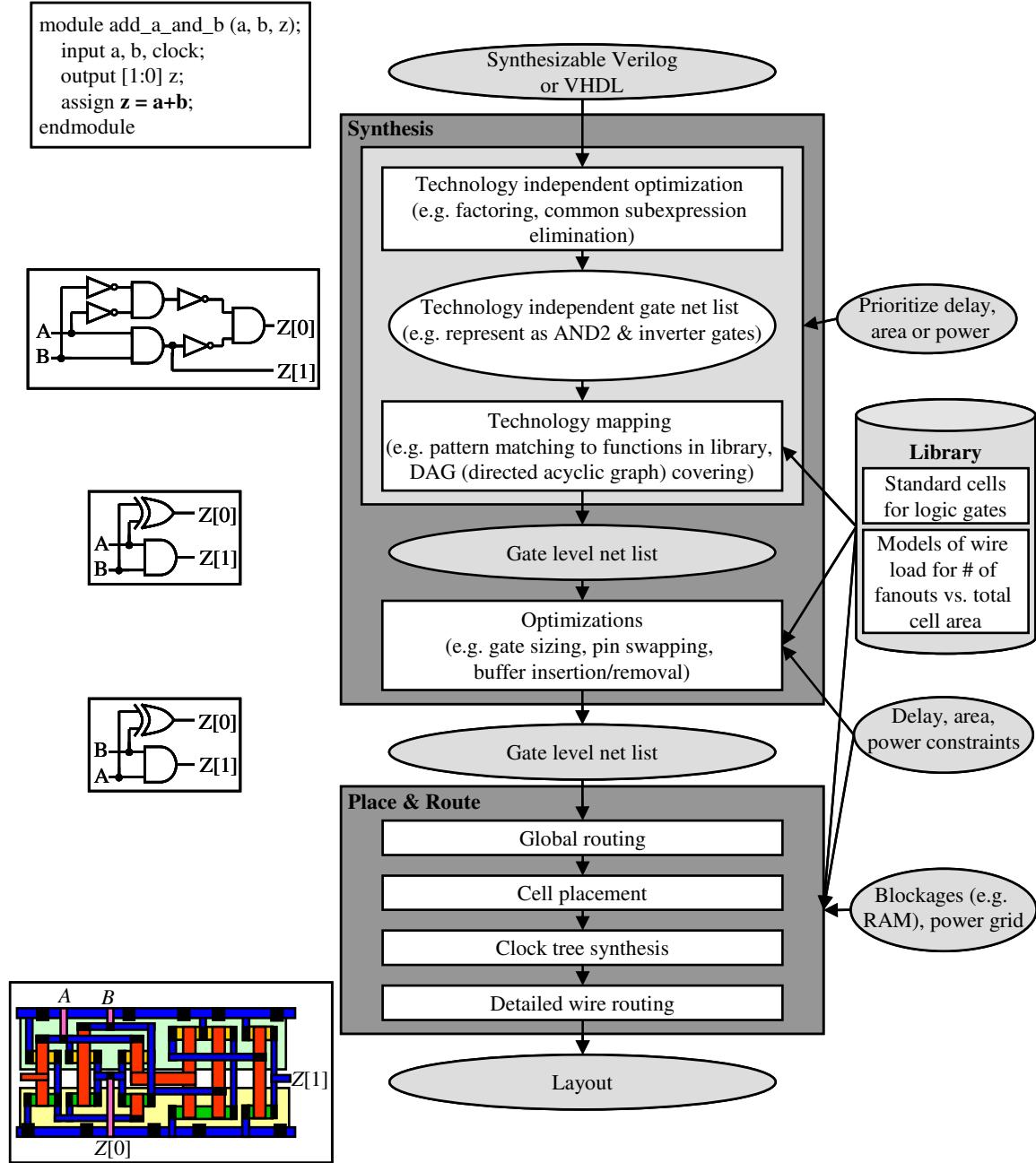


Figure 1.1 A typical EDA flow from a high level hardware design language (HDL) description through to layout. The diagrams on the left show, from top to bottom: typical Verilog code; technology-independent gate level net list; gate level net list; optimized gate level net list with pins switched to reduce the delay of a late arriving signal; and layout. The XOR2 layout to compute Z[0] is composed per the six transistor XOR2 on page 525 of [236], and the AND2 is composed of a NAND2 and inverter shown in Figure 1.2.

1.2 What is a standard cell ASIC methodology?

A standard cell ASIC methodology incorporates a standard cell library and automated design tools to utilize this, to achieve higher designer productivity. The designer specifies the circuit behavior in a hardware description language (HDL) such as Verilog or VHDL [184]. This high

level description is then mapped to a library of standard cells that implement various logic functions, as shown in Figure 1.1. Various optimizations are performed to try and meet delay, power, or area constraints specified by the designer. The final layout of the chip is not known at this stage, so the wire capacitances are estimated using a wire load model. Then the standard cells are placed, wires are routed between them, and a clock tree network is inserted to distribute the clock signal to all the clock-triggered registers that store results from combinational logic.

The EDA flow may be iterated through many times as a design is changed to meet performance constraints. Small changes may be made at the layout level, but significant changes like resizing gates on a delay-critical path usually require redoing at least place and route. After place and route, wire load models for later iterations may be updated based on the resulting layout.

There are also verification steps to try to ensure that the final circuit that is fabricated performs correctly. These include verifying that the gate level logic corresponds to the HDL description; gate level simulation to check correct functional behavior; verifying the layout meets design rules; checking that supply and ground voltage (IR) drops are within tolerances for the standard cell library or design; cross-talk analysis to check signal interference between wires on the chip; and electromagnetic interference analysis to check signal interference with the surrounding environment.

Custom designers sometimes use an ASIC methodology, in particular for portions of the chip that are not timing critical, such as control logic. In contrast, for performance-critical datapath logic, it is highly advantageous in terms of speed, power, and area to manually lay out the semi-regular logic. As their position is known, logic cells may be custom-designed to have less guard banding, or input and output ports in a particular place to reduce wire lengths, and so forth. Custom design of individual logic gates and manual cell placement is laborious, increasing the time-to-market and requiring much larger design teams. In addition, such design-specific optimizations are

seldom useful on other designs except for commonly used structures such as memory, and also may not be usable if the technology for the design changes. There have been several attempts by EDA companies to sell datapath synthesis tools, but they have not been successful. It is very difficult for tools to identify the appropriate layout, as a datapath does not usually a regular structure that can be identified by a general purpose tool, though some design companies do have in-house datapath generation tools.

Using a standard cell library that is provided by a vendor for a given fabrication process technology improves designer productivity. Lower transistor-level circuit design issues are abstracted to gate-level power and delay characteristics, and standard cells are designed robustly with guard-banding to ensure correct behavior in the process technology. For each logic gate, a library typically has several drive strengths of standards cells that implement that logic function. These drive strengths correspond to the capacitive load that a cell can drive without excessive delay and with acceptable signal characteristics. Cell placement is simplified by using a fixed height for all the cells. Rows of cells are placed on the chip, with contiguous supply voltage (V_{dd}) rails and ground voltage (GND) rails at the top and bottom of the rows as shown in Figure 1.2. This makes it possible for automated placement of standard cells in the manner shown in Figure 1.3. A new standard cell library can be used by iterating an RTL design through the design flow again. This makes it much easier to migrate between process technologies.

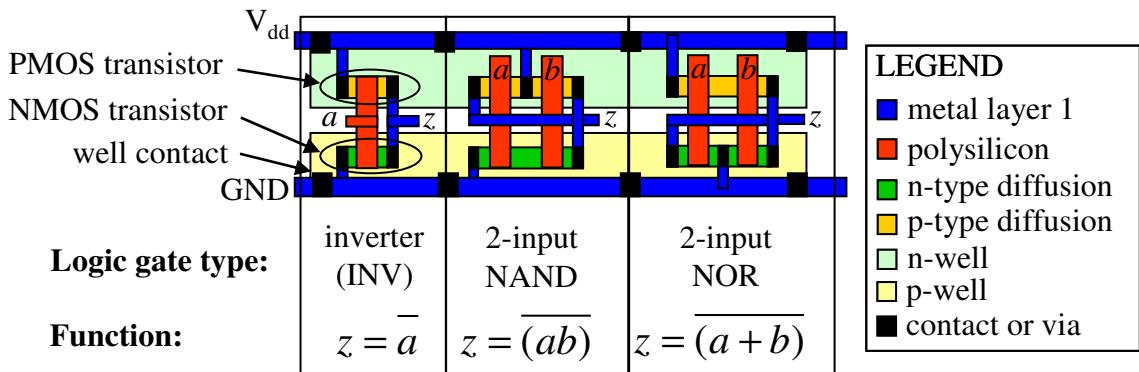


Figure 1.2 This figure shows a “stick” diagram of the layout of three simple combinational logic gates on the same standard cell row. Wires are generally routed over the top of the standard cells, with the standard cells in this case posing a barrier to routing with metal layer 1 and below.

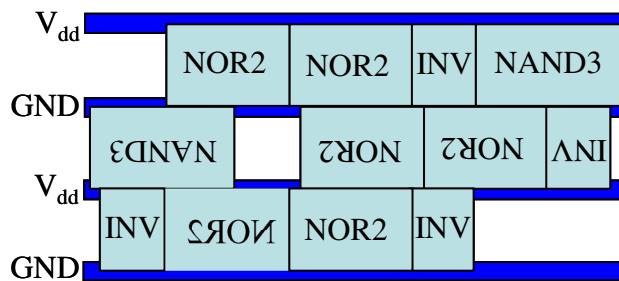


Figure 1.3 Placement of standard cells on standard cell rows are shown, with cells on alternate rows rotated 180° to share power rails. Standard cell height is fixed, but width and placement along a row may vary, and cells may also be mirrored horizontally.

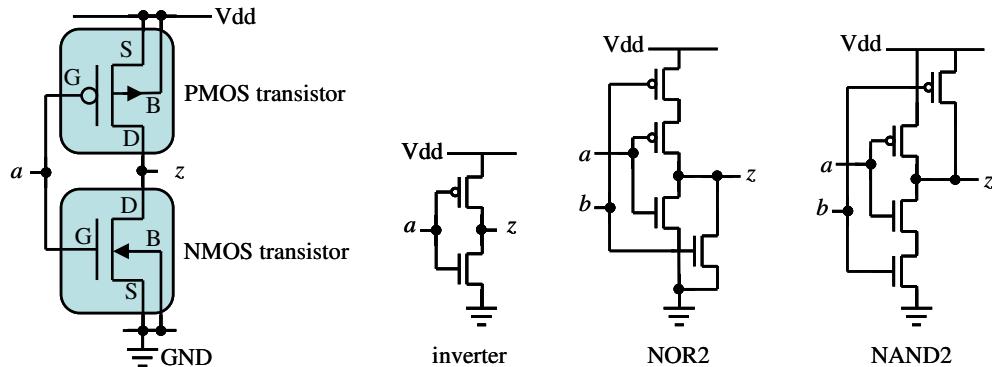


Figure 1.4 On the left is shown a detailed circuit schematic for an inverter. Transistor gate G , source S , drain D and bulk B (also referred to as substrate) nodes are noted. Connections to the substrate are generally omitted, in which case it is assumed that the NMOS p-well connects to ground (0V), and the PMOS n-well connects to the supply voltage (V_{dd}). On the right are shown the circuit schematics for three logic gates.

We will discuss later optimizing the drive strength of logic gates in a circuit and similar gate-level and transistor-level issues. Thus it is useful to briefly examine transistor-level layouts. The left of Figure 1.4 shows a detailed circuit schematic for an inverter, corresponding to the inverter layout in Figure 1.2. There is some ambiguity when we refer to a “gate”, whether it is a logic gate such

as an inverter, or the transistor gate shown labeled G – this will be clarified where appropriate in the text.

In Figure 1.4, note that the NOR2 gate has two PMOS transistors in series in the pull-up portion, whereas the NAND2 gate has two NMOS transistors in series in the pull-down portion. The more transistors in series, the slower the logic gate is due to increased series resistance. PMOS transistors are slower than NMOS transistors, so the NOR2 is slower than a NAND2, assuming the same transistor sizes. Wider transistors may be used to reduce the delay. A typical inverter PMOS to NMOS ratio to have equal pull-up and pull-drive strengths is 2:1. To reduce the additional delay of transistors in series, for a NOR2 gate this becomes 4:1, and for a NAND2 gate this is 2:2. Skewed P to N ratios, substrate biasing, and other circuit issues will be discussed briefly in later chapters. However, increasing the transistor widths increases the power used in charging and discharging the logic gate.

This is an example of low-level power-performance trade-offs that would be considered by a custom circuit designer. To reduce the time to design a circuit, an ASIC circuit designer typically avoids such issues by using a fixed library of standard cells. It is assumed that the library has appropriate PMOS to NMOS ratios and a range of sizes for logic gates to optimally drive the circuit in different load capacitance conditions. However, this assumption is not necessarily true. Such factors may contribute to suboptimal ASIC designs. This and other power-performance trade-offs are examined in detail in this thesis.

1.3 Who should care?

1.3.1 ASIC and ASSP designers seeking high performance

Power consumption has become a major design constraint for high performance circuits and limits performance for high end microprocessor chips in today's technologies. Our book titled *Closing the Gap between ASIC & Custom* [38] detailed how to achieve high performance for

ASICs in an EDA design flow, but we did not focus on the limitations imposed by power consumption. Some of the techniques used to achieve lower power in high performance custom designs can be automated for use in an ASIC design methodology.

While many ASIC designers may be power-budget limited when seeking higher performance, we quickly acknowledge that not all ASIC designers are seeking higher performance. Many ASIC designs need only to be cheaper than FPGAs (field programmable gate arrays) or faster than general-purpose processor solutions to be viable. For these designs, the desire for higher performance is dominated by final part cost, low non-recurring engineering cost, and time-to-market concerns. Non-recurring engineering costs for ASICs have grown substantially with increased transistor density and deep-submicron design issues. Mask-set costs are now exceeding one million dollars. Both the number and cost of tools required to do ASIC design are rising. In order to recoup their massive investment in fabrication facilities, semiconductor vendors for ASICs are “raising the bar” for incoming ASIC designs. Specifically, ASIC semiconductor vendors are raising the minimum volumes and expected revenues required to enter into contract for fabricating ASICs. These different factors are causing more ASIC design groups to rethink their approach. Some groups are migrating to using FPGA solutions. Some groups are migrating to application-specific standard parts (ASSPs) that can be configured or programmed for their target application.

Those groups that retain their resolve to design ASICs have a few common characteristics. First, these groups aim to amortize increasing non-recurring engineering costs for ASIC designs by reusing a design across multiple applications. Thus they are no longer designing “point solution” ASICs, but are tending toward more sustainable IC *platforms* with software that can be updated as application requirements change [119]. Secondly, as transistor density increases with Moore’s law [145], more and more devices are integrated onto a single chip to reduce the net production cost. Multiple processor cores are integrated onto a chip to increase performance or allow for

more programmability to achieve retargetable IC platforms. However, the power consumption also increases with more devices on a chip. Finally, given the effort and attention required to design a highly complex ASIC, design groups are demanding more out of their investment. In short, we contend that there are an increasing number of ASIC and ASSP designers who require both a high performance and low power design flow.

In short, the first portion of this thesis targets ASIC and ASSP designers seeking high-performance within an automated design methodology, and we contend that this number is *increasing* over time.

1.3.2 ASIC and ASSP designers seeking lower power

Power consumption is of primary importance in chips designed for embedded and battery powered applications. To reduce part costs, cheap plastic packaging is preferred, which limits the maximum heat dissipation. For many applications such as mobile phones, a long battery lifetime is desirable, so low power is important. ASIC implementations are often chosen for low power, as they can be an order of magnitude or more lower power than applications implemented on an FPGA [126] or in software running on a general purpose processor.

The main approaches to reducing power consumption are scaling down supply voltage and using smaller gate sizes to reduce dynamic power, and increasing threshold voltage to reduce static leakage power; however, these techniques to reduce power also substantially slow down a circuit. Thus we focus on reducing the power gap between ASIC and custom designs subject to some performance constraint. In ASIC designs with tight performance constraints and a tight power budget, ASIC designers must use high performance techniques to create some timing slack for power minimization.

1.3.3 Custom designers seeking higher productivity

An equally important audience for this thesis is custom designers seeking low power ICs in a design methodology that uses less human resources, such as an ASIC design methodology. Without methodological improvements, custom design teams can grow as fast as Moore's Law to design the most complex custom ICs. Even the design teams of the most commercially successful microprocessors cannot afford to grow at that rate.

We hope to serve this audience in two ways. First, we account for the relative power impact of different elements of a custom-design methodology. Projects have limited design resources and must be used judiciously. Therefore, design effort should be applied where it offers the greatest benefit. We believe that our analysis should help to determine where limited design resources are best spent.

Secondly, specific tools targeted to reduce the power of ASICs can be applied to custom design. The custom designer has always lacked adequate tool support. Electronic Design Automation (EDA) companies have never successfully found a way to tie their revenues to the revenues of the devices they help design. Instead, EDA tool vendors get their revenues from licensing design tools for each designer, known as a "design seat". It doesn't matter if the chip designed with an EDA tool sells in volumes of ten million parts or one, the revenue to the EDA company is the same. It has been estimated that there are more than ten times as many ASIC designers (50,000 – 100,000 worldwide) as custom designers (3,000 – 5,000 worldwide). As a result EDA tool vendors naturally "follow the seats" and therefore have focused on tools to support ASIC designers rather than custom designers. Companies using custom design augment tools from EDA vendors with their own in-house tools. These in-house tools can also be improved by identifying where gaps exist in the standard approaches that have been used for circuit design, or replaced in cases where EDA tools perform sufficiently well.

1.4 Organization of the rest of the thesis

This thesis examines the power gap between ASIC and custom design methodologies, and techniques to reduce the power gap while being subject to a performance constraint. Examining the power consumption of similar ASIC and custom designs in Chapter 2, we estimate the range of the power gap. Then based on our design and EDA tool experiences, and a careful study of different design factors, we quantitatively estimate the importance of the major factors contributing to the power gap between ASIC and custom designs.

Having identified voltage scaling as potentially the largest factor for reducing power, we study voltage scaling in conjunction with other factors. Chapter 3 looks at a simple model to examine the microarchitectural power-performance trade-off of pipelining in the context of gate sizing and voltage scaling. Chapter 5 details power-performance trade-offs with voltage scaling.

Chapter 6 discusses circuit optimization using gate sizing with voltage scaling, multiple threshold voltages and multiple supply voltages. Using convex models, it is possible to formulate the optimization problem as a geometric program. However, geometric programming is limited by large runtimes and the accuracy of the convex models.

A linear programming optimization approach for gate sizing with more accurate analysis and faster run times is detailed in Chapter 4. The linear programming approach achieves lower power consumption than existing circuit sizing approaches using commercial software. With this power minimization approach, Chapter 7 explores the use of multiple threshold voltages and multiple supply voltages versus voltage scaling and gate sizing.

Chapter 8 provides a short summary of this work, in the general context of circuit design subject to power and performance constraints.

1.5 What's not in this thesis

This thesis focuses on power consumption of integrated circuits and the tools and techniques by which lower power can be achieved. ASIC and custom performance and approaches to increase circuit speed were discussed extensively in our book on the topic [38]. Area minimization is also not a direct focus, as that is a less critical design constraint compared to speed and power in today's technologies that allow billions of transistors on a chip. Notably absent from this thesis are discussions of functional verification, manufacturing test, manufacturing yield, and the impact of deep-submicron effects on integrated circuit design. These topics are important ones but are, unless mentioned, orthogonal issues to those that we discuss here. Where the techniques that we suggest here negatively impact one or more of these issues we have made every effort to point that out.

Chapter 2. Overview of the Factors Affecting the Power Consumption

We investigate differences in power between application-specific integrated circuits (ASICs) and custom integrated circuits, with examples from 0.6um to 0.13um CMOS. A variety of factors cause synthesizable designs to consume 3 to 7 \times more power. We discuss the shortcomings of typical synthesis flows, and changes to tools and standard cell libraries needed to reduce power. Using these methods, we believe that the power gap between ASICs and custom circuits can be closed to within 2.6 \times at a tight performance constraint for a typical ASIC design.

2.1 Introduction

In the same technology generation, custom designs can achieve 3 to 8 \times higher clock frequency than ASICs [38]. Custom techniques that are used to achieve high speed can also be used to achieve low power [144]. Custom designers can optimize the individual logic cells, the layout and wiring between the cells, and other aspects of the design. In contrast, ASIC designers generally focus on optimization at the RTL level, relying on EDA tools to map RTL to cells in a standard cell library and then automatically place and route the design. Automation reduces the design time, but the resulting circuitry may not be optimal.

Low power consumption is essential for embedded applications. Power affects battery life and the heat dissipated by hand-held applications must be limited. Passive cooling is often required, as using a heat sink and/or fan is larger and more expensive.

Power is also becoming a design constraint for high-end applications due to reliability, and costs for electricity usage and cooling. As technology scales, power density has increased with transistor density, and leakage power is becoming a significant issue even for high end processors. Power consumption is now a major problem even for high end microprocessors. Intel canceled the next generation Tejas Pentium 4 chips due to power consumption issues [238].

In this chapter, we will discuss the impact of manual and automated design on the power consumption, and also the impact of process technology and process variation. Our aim is to quantify the influence of individual design factors on the power gap. Thus, we begin by discussing a process technology independent delay metric in Section 2.2. Section 2.3 discusses the contribution to a chip’s power consumption from memory, control and datapath logic, and clocking, and also provides an overview of dynamic and leakage power.

In Section 2.4, we compare full custom and synthesizable ARM processors and a digital signal processor (DSP) functional unit. We show that ASICs range from 3 to 7 \times higher power than custom designs for a similar performance target. To date the contribution of various factors to this gap has been unclear. While automated design flows are often blamed for poor performance and poor energy efficiency, process technology is also significant. Section 2.5 outlines factors contributing to the power gap. We then examine each factor, describing the differences between custom and ASIC design methodologies, and account for its impact on the power gap. Finally, we detail approaches that can reduce this power gap. We summarize our analysis in Section 2.6.

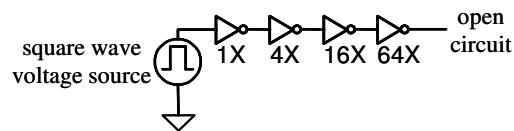


Figure 2.1 This illustrates a circuit to measure FO4 delays. The delay of the 4X drive strength inverter gives the FO4 delay. The other inverters are required to appropriately shape the input waveform to the 4X inverter, and reduce the switching time of the 16X inverter, which affect the delay of the 4X inverter [88].

2.2 Process technology independent FO4 delay metric

At times we will discuss delay in terms of FO4 delays. It is a useful metric for normalizing out process technology dependent scaling of the delay of circuit elements.

The fanout-of-4 inverter delay is the delay of an inverter driving a load capacitance that has four times the inverter’s input capacitance [88]. This is shown in Figure 2.1. The FO4 metric is not substantially changed by process technology or operating conditions. In terms of FO4 delays,

other fanout-of-4 gates have at most 30% range in delay over a wide variety of process and operating conditions, for both static logic and domino logic [88].

If it has not been simulated in SPICE or tested silicon, the FO4 delay in a given process technology can be estimated from the channel length. Based on the effective gate length L_{eff} , the rule of thumb for FO4 delay is [95]

$$360 \times L_{eff} \text{ ps for typical operating and typical process conditions} \quad (2.1)$$

$$500 \times L_{eff} \text{ ps for worst case operating and typical process conditions} \quad (2.2)$$

where the effective gate length L_{eff} has units of micrometers. Typical process conditions give high yield, but are not overly pessimistic. Worst case operating conditions are lower supply voltage and higher temperature than typical operating conditions. Typical operating conditions for ASICs may assume a temperature of 25°C, which is optimistic for most applications. Equation (2.2) can be used to estimate the FO4 delay in silicon for realistic operating conditions [95].

L_{eff} is often assumed to be about 0.7 of the drawn gate length for a process technology – for example, 0.13um for a 0.18um process technology. However, many foundries are aggressively scaling the channel length to increase the speed. Thus, the FO4 delay should be calculated from the effective gate length, if it is known, rather than from the process technology generation.

From previous analysis [38], typical process conditions are between 17% and 28% faster than worst case process conditions. Derating worst case process conditions by a factor of $1.2 \times$ gives

$$600 \times L_{eff} \text{ ps for worst case operating and worst case process conditions} \quad (2.3)$$

Equation (2.3) was used for estimating the FO4 delays of synthesized ASICs, which have been characterized for worst case operating and worst case process conditions. This allows analysis of

the delay per pipeline stage, independent of the process technology, and independent of the process and operating conditions.

Note: these rules of thumb give approximate values for the FO4 delay in a technology. They may be inaccurate by as much as 50% compared to simulated or measured FO4 delays in silicon. These equations do not accurately account for operating conditions. Speed-binning and process improvements that do not affect the effective channel length are not accounted for. Accurate analysis with FO4 delays requires proper calibration of the metric: simulating or measuring the actual FO4 delays for the given process and operating conditions.

2.3 Components of power consumption

Designers typically focus on reducing both the total power when a circuit is active and its standby power. There is usually a minimum performance target, for example 30 frames/s for MPEG. When performance is less important, the energy per operation to perform a given task can be minimized.

Active power includes both dynamic power consumption, when the logic evaluates or the clock transitions, and current leakage when logic is not switching. There is no computation in logic in standby, the clock must be gated to prevent it switching, and leakage is the dominant source of power consumption in standby.

The major sources of power consumption in circuitry are the clock tree and registers, control and datapath logic, and memory. The breakdown of power consumption between these is very application and design dependent. The power consumption of the clock tree and registers ranged from 18% to 36% of the total power for some typical embedded processors and microprocessors (see Section 3.2.4). In custom cores for discrete cosine transform (DCT) and its inverse (IDCT), contributions to the total power were 5% to 10% from control logic, about 40% from the clock tree and clock buffers, and about 40% from datapath logic [239][240]. Memory can also account

for a substantial portion of the power consumption. For example, in the StrongARM caches consume 43% of the power [144].

2.3.1 Dynamic power

Dynamic power is due to switching capacitances and short circuit power when there is a current path from supply to ground.

The switching power is proportional to αfCV_{dd}^2 , where α is the switching activity per clock cycle, f is the clock frequency, C is the capacitance that is (dis)charged, and V_{dd} is the voltage swing. The switching activity is increased by glitches, which typically cause 15% to 20% of the activity in complementary static CMOS logic [182].

Short circuit power typically contributes less than 10% of the total dynamic power [32], and increases with increasing Vdd, and with decreasing Vth. Short circuit power can be reduced by matching input and output rise and fall times [227].

As the dynamic power depends quadratically on Vdd, methods for reducing active power often focus on reducing Vdd. Reducing the capacitance by downsizing gates and reducing wire lengths is also important.

2.3.2 Leakage power

In today's processes, leakage can account for 10% to 30% of the total power when a chip is active. Leakage can contribute a large portion of the average power consumption for low performance applications, particularly when a chip has long idle modes without being fully off.

Optimally choosing Vdd and Vth to minimize the total power consumption for a range of delay constraints in 0.13um technology, the leakage varied from 8% to 21% of the total power consumption in combinational logic, as discussed later in Section 5.2.6. However, the possible Vdd and Vth values depend on the particular process technology and standard cell libraries

available. For example for a delay constraint of $1.2 \times$ the minimum delay, the best library choice had Vdd of 0.8V and Vth of 0.08V (see Table 7.7 with 0.8V input drivers), and leakage contributed on average 40% of total power.

Leakage power in complementary static CMOS logic in bulk CMOS is primarily due to subthreshold leakage. Subthreshold leakage increases exponentially with decrease in Vth and increase in temperature. It can also be strongly dependent on transistor channel length in short channel devices. Gate tunneling leakage is becoming significant as gate oxide thickness reduces with device dimensions. There is also substrate leakage. Leakage has become increasingly significant in deep submicron process technologies.

2.4 ASIC and custom power comparison

To illustrate the power gap, we examine custom and ASIC implementations of ARM processors and dedicated hardware to implement discrete cosine transform (DCT) and its inverse (IDCT). ARM processors are general purpose processors for embedded applications. ASICs often have dedicated functional blocks to achieve low power and high performance on specific applications. Media processing is a typical example where high speed and low power is required. JPEG and MPEG compression and decompression of pictures and video use DCT and IDCT. There is a similar power gap between ASIC and custom for the ARM processors and for DCT and IDCT blocks.

Table 2.1 Full custom and hard macro ARMs [27][68][69][106][166]. The highlighted full custom chips have 2 to 3x MIPS/mW.

ARM Processor	Technology (um)	Voltage (V)	Frequency (MHz)	MIPS	Power (mW)	MIPS/mW
ARM710	0.60	5.0	40	36	424	0.08
Burd	0.60	1.2	5	6	3	1.85
Burd	0.60	3.8	80	85	476	0.18
ARM810	0.50	3.3	72	86	500	0.17
ARM910T	0.35	3.3	120	133	600	0.22
StrongARM	0.35	1.5	175	210	334	0.63
StrongARM	0.35	2.0	233	360	950	0.38
ARM920T	0.25	2.5	200	220	560	0.39
ARM1020E	0.18	1.5	400	500	400	1.25
XScale	0.18	1.0	400	510	150	3.40
XScale	0.18	1.8	1000	1250	1600	0.78
ARM1020E	0.13	1.1	400	500	240	2.08

Table 2.2 The highlighted ARM7TDMI hard macros have 1.3 to 1.4x MIPS/mW versus the synthesizable ARM7TDMI-S cores [12].

ARM Core	Technology (um)	Frequency (MHz)	Power (mW)	MIPS/mW
ARM7TDMI	0.25	66	51	1.17
ARM7TDMI-S	0.25	60	66	0.83
ARM7TDMI	0.18	100	30	3.00
ARM7TDMI-S	0.18	90	35	2.28
ARM7TDMI	0.13	130	10	11.06
ARM7TDMI-S	0.13	120	13	8.33

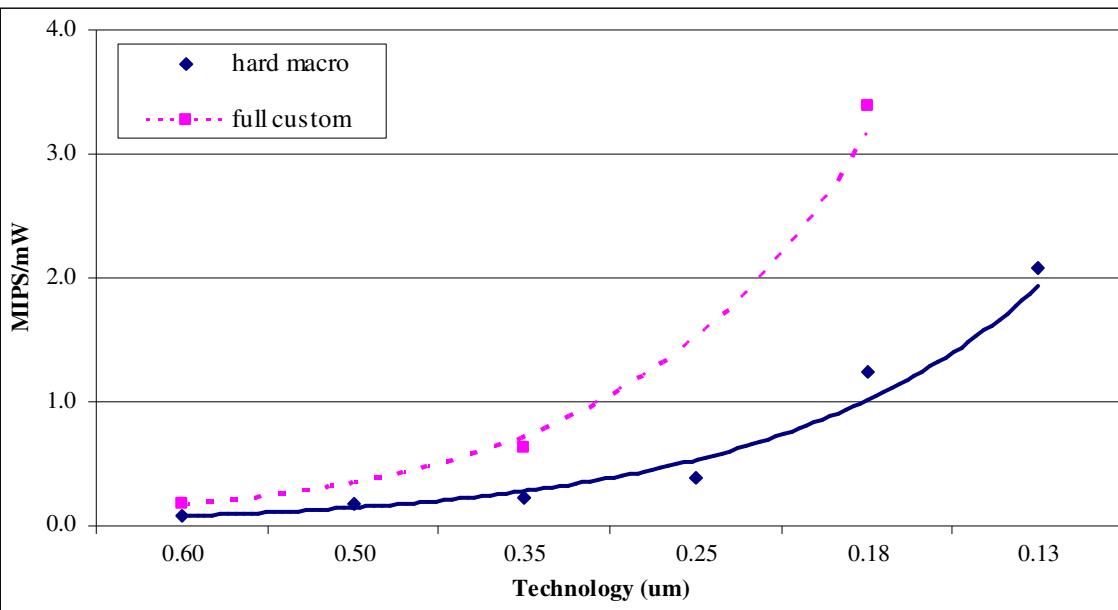


Figure 2.2 This graph compares MIPS/mW of the custom and hard macro ARMs in Table 2.1.

2.4.1 ARM processors from 0.6 to 0.13um

We compare chips with full custom ARM processors, soft, and hard ARM cores. Soft macros of RTL code may be sold as individual IP (intellectual property) blocks and are portable between fabrication processes. A hard macro is a design where the standard cell logic used, layout and wiring have been specified and optimized then fixed for a particular fabrication process. A hard macro may be custom, or it may be “hardened” from a soft core. A complete chip includes additional memory, I/O logic, and so forth.

To quantify the differences between ASIC and custom power consumption, we first examined hard macro and full custom ARMs, listed in Table 2.1. Compared to the other designs, the three full custom chips in bold achieved 2 to 3 \times millions of instructions per second per milliwatt (MIPS/mW) at similar MIPS, as shown in Figure 2.2. The inverse of this metric, mW/MIPS, is the energy per operation. The Dhrystone 2.1 MIPS benchmark is the performance metric [234]. It fits in the cache of these designs, so there are no performance hits for cache misses or additional power to read off-chip memory.

Lower power was achieved in several ways. The DEC StrongARM used clock-gating and cache sub-banking to substantially reduce the dynamic power [144]. The Intel XScale and DEC StrongARM used high speed logic styles to reduce critical path delay, at the price of higher power consumption on these paths. To reduce pipeline register delay, the StrongARM used pulse-triggered flip-flops [144] and the XScale used clock pulsed latches [45]. Shorter critical paths allow the same performance to be achieved with a lower supply voltage (Vdd), which can lower the total power consumption. Longer channel lengths were used in the StrongARM caches to reduce the leakage power, as the two 16kB caches occupy 90% of the chip area [144]. The XScale used substrate biasing to reduce the leakage [47].

For the same technology and similar performance (MIPS), the Vdd of the full custom chips is lower than that of the hard macros – reducing Vdd gives a quadratic reduction in dynamic power. The StrongARM can operate at up to 233MHz at 2.0V and the XScale can operate at up to 1GHz at 1.65V [106]. If operating at higher performance was not required, it is likely that even higher MIPS/mW could have been achieved.

Energy efficiency can be improved substantially if performance is sacrificed. Burd's 0.6um ARM8 had software controlled dynamic voltage scaling based on the processor load. It scaled from 0.18MIPS/mW at 80MHz and 3.8V, to 2.14MIPS/mW at 5MHz and 1.2V [27]. Voltage scaling increased the energy efficiency by 1.1 \times for MPEG decompression which required an average clock frequency of 50MHz, and increased the energy efficiency by 4.5 \times for audio processing which required a clock frequency of only 17MHz [28].

There is an additional factor of 1.3 to 1.4 \times between hard macro and synthesizable ARM7 soft cores, as shown in Table 2.2. These MIPS/mW are higher than those in Table 2.1, as they exclude caches and other essential units. The ARM7TDMI cores are also lower performance, and thus can achieve higher energy efficiency.

Overall, there is a factor of 3 to 4 \times between synthesizable ARMs and the best full custom ARM implementations.

2.4.1.1 Other full custom ARM implementations

There are two other higher performance full custom ARM implementations that are noteworthy, though they are less energy efficient than the 0.18um XScale.

Samsung's Halla is a full custom 0.13um implementation of the ARM1020E with power consumption from 0.26W at 400MHz and Vdd of 0.7V to 1.8W at 1200MHz and Vdd of 1.1V [120]. Achieving 1480MIPS at 1200MHz clock frequency, the energy efficiency ranged from

0.82MIPS/mW at 1200MHz to 1.90MIPS/mW at 400MHz. Differential cascode voltage switch logic (DCVSL) was used for high performance, but DCVSL has substantial power consumption compared to complementary static CMOS logic that is used in ASICs. Sense amplifiers were used with the low voltage swing dual rail bus to detect voltage swings of less than 200mV, achieving high bus speeds at lower power consumption [136]. The die area of the Halla was 74% more than ARM's 0.13um ARM1020E.

Intel's 90nm implementation of the XScale, codenamed Monahans, has 770mW dynamic power consumption at 1500MHz and Vdd of 1.5V with performance of 1200MIPS at this point [169]. The energy efficiency of Monahans is 1.56MIPS/mW at 1500MHz – data for improved energy efficiencies at lower Vdd has not been published. Clock pulsed latches were also used in this implementation of the XScale. The hold time for the clock gating enable signal was the duration of the clock pulse, and thus did not require latching. Domino logic was used for high performance in the shifter and cache tag NOR comparators. 75% of instruction cache tag accesses were avoided by checking if the instruction cache request line was the same as the previous one. Selective accesses and avoiding repeated accesses reduced power by 42% in the dynamic memory management unit [44].

Table 2.3 Comparison of ASIC and custom DCT/IDCT core power consumption at 30 frames/s for MPEG2 [61][239][240].

Design	Technology (um)	Voltage (V)	DCT (mW)	IDCT (mW)
ASIC	0.18	1.60	8.70	7.20
custom DCT	0.6 (L_{eff} 0.6)	1.56	4.38	
custom IDCT	0.7 (L_{eff} 0.5)	1.32		4.65

2.4.2 Comparison of IDCT/DCT cores

Application-specific circuits can reduce power by an order of magnitude compared to using general purpose hardware [182]. Two 0.18um ARM9 cores were required to decode 30 frames/s for MPEG2. They consumed 15× the power of a synthesizable DCT/IDCT design [61]. However, the synthesizable DCT/IDCT significantly lags its custom counterparts in energy efficiency.

Fanucci and Saponara designed a low power synthesizable DCT/IDCT core, using similar techniques to prior custom designs. Despite being three technology generations ahead, the synthesizable core was 1.5 to 2.0 \times higher power [61]. Accounting for the technology difference by conservatively assuming power scales linearly with device dimensions [167], the gap is a factor of 4.3 to 6.6 \times . The data is shown in Table 2.3.

Table 2.4 Factors contributing to ASICs being higher power than custom. The excellent column is what ASICs may achieve using low power and high performance techniques. This table focuses on the total power when a circuit is active, so power gating and other standby leakage reduction techniques are omitted. The combined impact of these factors is not multiplicative – see discussion in Section 2.5.1.

Contributing Factor	Typical ASIC	Excellent ASIC
microarchitecture	5.1 \times	1.9 \times
clock gating	1.6 \times	1.0 \times
logic style	2.0 \times	2.0 \times
logic design	1.2 \times	1.0 \times
technology mapping	1.4 \times	1.0 \times
cell and wire sizing	1.6 \times	1.1 \times
voltage scaling	4.0 \times	1.0 \times
floorplanning and placement	1.5 \times	1.1 \times
process technology	1.6 \times	1.0 \times
process variation	2.0 \times	1.3 \times

2.5 Factors contributing to ASICs being higher power than custom

Various parts of the circuit design and fabrication process contribute to the gap between ASIC and custom power. Our analysis of the most significant design factors and their impact on the total power when a chip is active is outlined in Table 2.4. The “typical” column shows the maximum contribution of individual factors comparing a typical ASIC to a custom design. In total these factors can make power an order of magnitude worse. In practice, even the best custom designs can't fully exploit all these factors simultaneously. Low power design techniques that can be incorporated within an EDA flow can reduce the impact of these factors in a carefully designed ASIC as per the “excellent” column in Table 2.4.

Most low power EDA tools focus on reducing the dynamic power in control logic, datapath logic, and the clock tree. The design cost for custom memory is low, because of the high regularity. Several companies provide custom memory for ASIC processes. Optimization of memory

hierarchy, memory size, caching policies, and so forth is application dependent and beyond the scope of this thesis, though they have a substantial impact on the system-level performance and power consumption. We will focus on the power consumption in a processor core.

Microarchitectural techniques such as pipelining and parallelism increase throughput, allowing timing slack for gate downsizing and voltage scaling. The microarchitecture also affects the average instructions per cycle (IPC), and hence energy efficiency. The power and delay overheads for microarchitectural techniques must be considered. With sufficient timing slack, reducing the supply voltage can greatly increase the energy efficiency. For example in Table 2.1, scaling the XScale from Vdd of 1.8V to 1.0V increases the efficiency from 0.78MIPS/mW to 3.40MIPS/mW, a factor of 4.4 \times , but the performance decreases from 1250MIPS to 510MIPS.

Process technology can reduce leakage by more than an order of magnitude. It also has a large impact on dynamic power. Process variation results in a wide range of the leakage power for chips and some variation in the maximum operating clock frequency for a given supply voltage. For high yield, a higher supply voltage may be needed to ensure parts meet the desired performance target, resulting in a significant spread in power consumption. Limiting process variation and guard-banding for it without being overly conservative help reduce the power consumption.

Using a high speed logic style on critical paths can increase the speed by 1.5 \times [38]. Circuitry using only slower complementary static CMOS logic at a tight performance constraint may be 2.0 \times higher power than circuitry using a high speed logic style to provide timing slack for power reduction by voltage scaling and gate downsizing.

Other factors in Table 2.4 have smaller contributions to the power gap. We will discuss the combined impact of the factors and then look at the individual factors and low power techniques to reduce their impact.

2.5.1 Combined impact of the contributing factors

The combined impact of the contributing factors is complicated. The estimate of the contribution from voltage scaling assumes that timing slack is provided by pipelining, so this portion is double counted. The timing slack depends on the tightness of the performance constraint, which has a large impact on the power gap. We assumed a tight performance constraint for both the typical ASIC and excellent ASIC for the contributions from microarchitecture, logic style, and voltage scaling detailed in Table 2.4. If the performance constraint is relaxed, then the power gap is less. For example, from our model of pipelining to provide timing slack for voltage scaling and gate sizing, the power gap between a typical ASIC and custom decreases from $5.1\times$ at a tight performance constraint for the typical ASIC to $4.0\times$ if the constraint is relaxed by 7%.

Chapter 3 details our power and delay model that incorporates pipelining, logic delay, voltage scaling and gate sizing. The logic delay is determined by factors such as the logic style, wire lengths after layout, process technology, and process variation which affects the worse case delay.

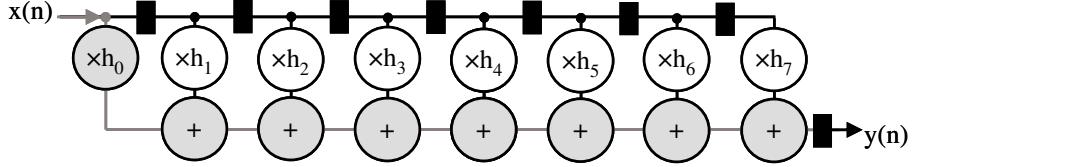
From analysis with this model¹, an excellent ASIC using the low power techniques that we recommend below may close the power gap to a factor of 2.2 at a tight performance constraint for a typical ASIC. If we require higher performance² beyond that of a typical ASIC to the maximum that can be achieved by an excellent ASIC, the power gap is $5.0\times$ between an excellent ASIC and a custom design taking full advantage of high performance design techniques to provide timing slack for voltage scaling and gate sizing.

¹ Pipeline model parameters, as detailed in Chapter 3: the pipeline stage delay overhead is assumed to be 20 FO4 delays in a typical ASIC, 5 FO4 delays in an excellent ASIC, and 3 FO4 delays in custom. The unpipelined combinational logic delay is assumed to be 180 FO4 delays for both a typical and excellent ASIC, but may be reduced to 104 FO4 delays in custom using a high speed logic style and adaptive body bias to compensate for process variation. At a tight performance constraint of 55.8 FO4 delays/instruction for a typical ASIC, there is no timing slack for voltage scaling or gate sizing, but there is timing slack in excellent ASIC and custom designs. Consequently, the typical ASIC's power may be $9.0\times$ worse than custom, but the excellent ASIC closes this gap to $2.2\times$.

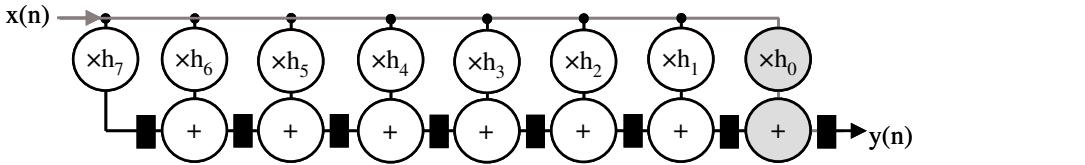
² A tight performance constraint for an excellent ASIC is 27.4 FO4 delays in our pipeline model.

Accounting for the additional factors of 1.1 \times for worse floorplanning and placement, and 1.1 \times for worse cell and wire sizing in an excellent ASIC, the power gap between an excellent ASIC and custom may be only 2.6 \times at a tight performance requirement for a typical ASIC¹, and 5.9 \times at a tight performance constraint for the excellent ASIC.

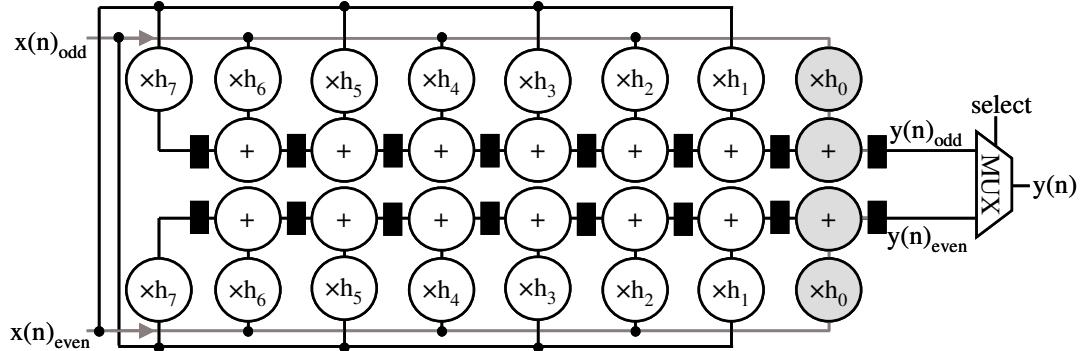
¹ In comparison, the power gap between a typical ASIC and custom at a tight performance requirement for the typical ASIC (55.8 FO4 delays/instruction) could be as large as 22 \times . This is calculated from the factor of 9.0 \times from the pipeline model, as described in the second last footnote, and the factors of 1.6 \times for worse gate sizing with a limited standard cell library, and 1.5 \times for worse floorplanning and placement comparing the typical ASIC to custom. In practice, even the best custom designs don't fully exploit all these factors simultaneously, so the actual power gap versus a typical ASIC is usually less.



(a) Direct form FIR filter



(b) Transpose form FIR filter



(c) Two-path parallel transpose FIR filter

Figure 2.3 This diagram shows pipelined (b) and parallel implementations (c) of the unpipelined direct form finite input response (FIR) filter in (a) [40][190]. The FIR filter calculates $y_n = h_0x_n + h_1x_{n-1} + \dots + h_7x_{n-7}$. The critical paths are shown in grey. The minimum clock period decreases as the registers break the critical path up into separate pipeline stages. Computation in each pipeline stage proceeds concurrently. The parallel implementation doubles the throughput at the expense of more than doubling the area. The multiplexer to select the odd or even result from the two parallel datapaths at each clock cycle is denoted by MUX.

2.5.2 Microarchitecture

Algorithmic and architectural choices can reduce the power by an order of magnitude [182]. We assume that ASIC and custom designers make similar algorithmic and architectural choices to find a low power implementation that is appropriate for the required performance and target application. Pipelining and parallelism are the two major microarchitectural techniques that can be used to maintain throughput (see Figure 2.3), when other power reduction techniques increase critical path delay. With similar microarchitectures, how do ASIC and custom pipelining and parallelism compare?

On their own, pipelining and parallelism do not reduce power. Pipelining reduces the critical path delay by inserting registers between combinational logic. Glitches may not propagate through pipeline registers, but the switching activity of the combinational logic is otherwise unchanged. Additional pipeline registers add to the leakage power and especially to the dynamic power, because the clock signal going to the registers has high activity. Pipelining may reduce the instructions per cycle (IPC) due to branch misprediction and other hazards; in turn this reduces the energy efficiency. Parallelism trades off area for increased throughput, with overheads for multiplexing and additional wiring [17]. Both techniques enable the same performance to be met at lower supply voltage with smaller gate sizes, which can provide a net reduction in power.

Bhavnagarwala et al. [17] predict a 2 to 4× reduction in power with voltage scaling by using 2 to 4 parallel datapaths. Generally, ASICs can make as full use of parallelism as custom designs, but careful layout is required to minimize additional wiring overheads.

Delay overheads for pipelining include: register delay; register setup time; clock skew; clock jitter; and any imbalance in pipeline stage delays that cannot be compensated for by slack passing or useful clock skew. For a given performance constraint, the pipelining delay overheads reduce the slack available to perform downsizing and voltage scaling.

In the IDCT, the cost of pipelining was about a 20% increase in total power, but pipelining reduced the critical path length by a factor of 4. For the same performance without pipelining, Vdd would have to be increased from 1.32V to 2.2V. Thus pipelining helped reduce power by 50% [240].

2.5.2.1 What's the problem?

The timing overhead per pipeline stage for a custom design is about 3 FO4 delays, but it may be 20 FO4 delays for an ASIC, substantially reducing the timing slack available for power reduction. For a typical ASIC, the budget for the register delay, register setup time, clock skew and clock

jitter is about 10 FO4 delays. Unbalanced critical path delays in different pipeline stages can contribute an additional 10 FO4 delays in ASICs. If the delay constraint is tight, a little extra timing slack can provide substantial power savings from downsizing gates – for example, a 3% increase in delay gave a 20% reduction in energy for a 64-bit adder [246].

For pipeline registers, most ASICs use slow edge-triggered D-type flip-flops that present a hard timing boundary between pipeline stages, preventing slack passing. The clock skew between clock signal arrivals at different points on the chip must be accounted for. Faster pulse-triggered flip-flops were used in the custom StrongARM [144]. Some pulse-triggered flip-flops have greater clock skew tolerance [194]. Custom designs may use level-sensitive latches to allow slack passing, and latches are also less sensitive to clock skew [40].

The custom XScale used clock-pulsed transparent latches [45]. A D-type flip-flop is composed of a master-slave latch pair. Thus a clock-pulsed latch has about half the delay of a D-type flip-flop and has a smaller clock load, which reduced the clock power by 33%. Clock-pulsed latches have increased hold time and thus more problems with races. The pulse width had to be carefully controlled and buffers were inserted to prevent races. The clock duty cycle also needs to be carefully balanced.

To estimate the impact of worse ASIC pipelining delay overhead, we developed a pipeline performance and power model, with power reduction from gate downsizing and voltage scaling versus timing slack (see Chapter 3). At a tight performance constraint for the ASIC design, we estimate that ASIC power consumption can be 5.1 \times that of custom, despite using a similar number of pipeline stages. While there is no timing slack available to the ASIC design, the lower custom pipeline delay overhead allows significant power reduction by gate downsizing and voltage scaling.

2.5.2.2 What can we do about it?

Latches are well-supported by synthesis tools [201], but are rarely used other than in custom designs. Scripts can be used to convert timing critical portions of an ASIC to use latches instead of flip-flops [37]. High-speed flip-flops are now available in some standard cell libraries and can be used in an automated design methodology to replace slower D-type flip-flops on critical paths [70]. Useful clock skew tailors the arrival time of the clock signal to different registers by adjusting buffers in the clock tree and can be used in ASIC designs for pipeline balancing [52]. With these methods, the pipeline delay overhead in ASICs can be reduced to as low as 5 FO4 delays [38]. This enables more slack to be used for downsizing, voltage scaling, or increasing the clock frequency. From our pipeline model, ASICs can close the gap for the microarchitecture and timing overhead factor to within 1.9× of custom.

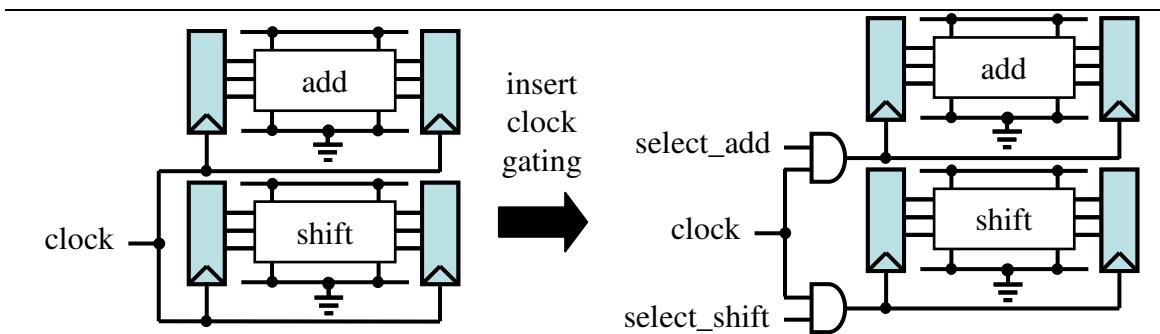


Figure 2.4 This is a simple illustration of clock gating. The clock signal to the registers is gated with a control signal that selects which functional unit is in use. A transparent low latch is usually inserted to de-glitch the enable signal [121].

2.5.3 Clock gating

In typical operation, pipeline stages and functional units are not always in use. For example, during a sequence of integer operations, the floating point unit may be idle. Providing the logical inputs to the idle unit are held constant, there are only two sources of power dissipation in the idle unit: static leakage; and switching activity at registers and any other clocked elements due to the clock signal – for example, precharge of domino logic.

Architectural or gate-level signals can turn off the clock to portions of the clock tree that go to idle units. This can be done with a clock gating control signal and clock signal at an AND gate, as illustrated in Figure 2.4. As the clock tree and registers can contribute 20% to 40% of the total power, this gives substantial dynamic power savings if units are often idle. The power overheads for logic to generate clock gating signals and the clock gating logic need to be compared versus the potential power savings. Usually clock gating signals can be generated within only one clock cycle, and there is only a small delay increase in the arrival of the gated clock signal at the register.

The StrongARM's total power when active would be about 1.5 \times worse without clock gating [144]. The StrongARM uses a 12 bit by 32 bit multiply-accumulate (MAC) unit. For some applications, one multiply operand will be 24-bit or less, or 12-bit or less, thus the number of cycles the 12 \times 32 MAC is required is less than the three cycles for a full 32 \times 32 multiply. This saves power by avoiding unnecessary computation. Typical code traces had shift operations of zero, so power could be saved by disabling the shifter in this case [144].

The custom DCT core uses clock gating techniques extensively. In typical operation, consecutive images are highly correlated. Calculations using the significant bits of pixels in common between consecutive images can be avoided. This reduced the number of additions required by 40%, and gave on average 22% power savings for typical images [239]. After the discrete cosine transform on a typical image, there are many coefficients of value zero. This was exploited in the custom IDCT to separately clock gate pipeline stages processing coefficients of zero [240].

We estimate that clock gating techniques can increase energy efficiency when the chip is active by up to 1.6 \times . Note that the power savings from clock gating vary substantially with the application.

2.5.3.1 What's the problem?

Clock gating requires knowledge of typical circuit operation over a variety of benchmarks. If a unit is seldom idle, clock gating would increase power consumption. Until recently, clock gating was not fully supported by commercial tools. Retiming to reposition the registers [179] can be essential to better balance the pipeline stages, but EDA tools did not support retiming of registers with clock gating.

Care must be taken with gated clock signals to ensure timing correct operation of the registers. Glitches in the enable signal must not propagate to the clock gate while the clock gate is high. This results in a long hold time for the enable signal, which may be avoided by inserting a transparent low latch to de-glitch the enable signal [121]. The transparent low latch prevents the signal that goes to the clock gate from changing while the clock is high. The setup time for the enable signal is longer to account for the clock gate and de-glitching latch. The clock signal arrives later to the register due to the delay of the clock gate, which increases the hold time for that register. The clock tree delay of the clock signal to the clock gate can be reduced to compensate for this, but that may require manual clock tree design.

2.5.3.2 What can we do about it?

An ASIC designer can make full use of clock gating techniques by carefully coding the RTL for the desired applications, or using automated clock-gating. The techniques used in custom DCT and IDCT designs were used in the synthesizable DCT/IDCT [61]. In the synthesizable DCT/IDCT, clock gating and data driven switching activity reduction increased the energy efficiency by 1.4 \times for DCT and 1.6 \times for IDCT [61].

In the last few years, commercial synthesis tools have become available to automate gate-level clock-gating, generating clock gating signals and inserting logic to gate the clock. There is now support for retiming of flip-flops with gated clock signals. Power Compiler was able to reduce the

power of the synthesizable ARM9S core by 41% at 5% worse clock frequency [65] – primarily by gate downsizing, pin reordering, and clock gating. Useful clock skew tools can compensate for the additional delay on the gated clock signal [52].

There are tools for analyzing the clock-tree power consumption and activity of functional units during benchmark tests. These tools help designers identify signals to cut off the clock signal to logic when it is not in use, and to group logic that is clock gated to move the clock gating closer to the root of the clock tree to save more power.

As ASICs can make effective use of clock gating, there should be no power gap due to clock gating in comparison with custom.

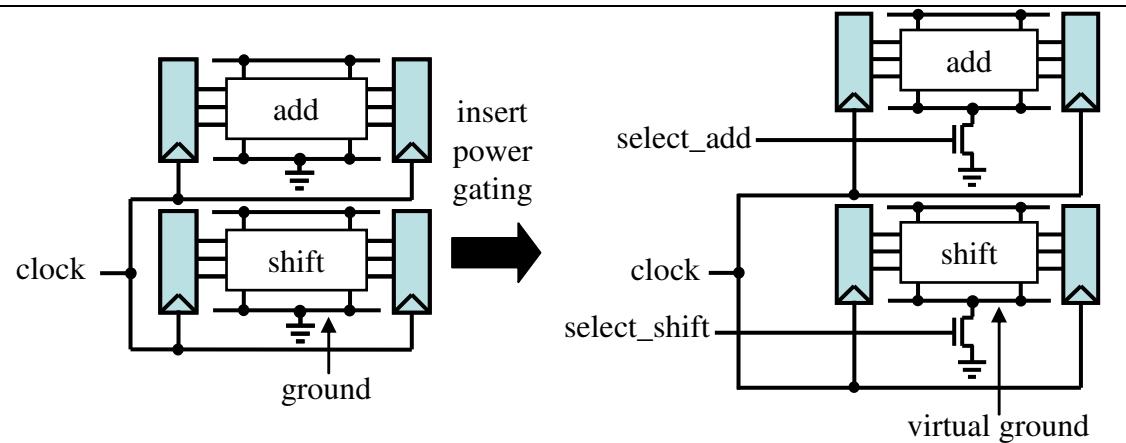


Figure 2.5 This is a simple illustration of power gating. The sleep transistors are turned on by a control signal that selects which functional unit is in use, reducing leakage from supply to ground. Registers may not be disconnected in this manner without losing state information.

2.5.4 Power gating and other techniques to reduce leakage in standby

With clock gating for idling units, only static leakage remains. The leakage can be substantially reduced by several methods: reducing the supply voltage (see Section 2.5.9); disconnecting the power rails with sleep transistors [150], known as *power gating*; increasing V_{th} via substrate biasing to reduce subthreshold leakage; and assigning logic gate inputs to a lower leakage state [130]. All of these methods take a significant amount of power and thus are only worthwhile when a unit will be idle for tens to thousands of clock cycles or more [58] – for example, when

most of a mobile phone's circuitry is idling while awaiting an incoming call. This requires architectural or software level signals to transition between normal operation and sleep mode.

Reducing the supply voltage reduces the subthreshold leakage current as there is less drain induced barrier lowering (DIBL), and also reduces the gate-oxide tunneling leakage current [131]. For example, leakage decreases by a factor of 3 \times when Vdd is reduced from 1.2V to 0.6V with our 0.13um libraries (see Section 5.2.4). Dynamic voltage scaling is discussed in more detail in Section 2.5.9.

Subthreshold leakage and gate leakage vary substantially depending on which transistors in a gate are off, which is determined by the inputs. Leakage in combinational logic can be reduced by a factor of 2 to 4 \times by assigning primary inputs to a lower leakage state [130][132]. Additional circuitry is required in the registers to store the correct state, while outputting the low leakage state; or state information may be copied from the registers and restored on resumption from standby. There is also dynamic power consumption in the combinational logic in the cycle that inputs are assigned to a low leakage state. Thus, units must be idle for on the order of ten cycles to justify going to a low leakage state.

Normally, the p-well of the NMOS transistors is connected to ground and the n-well of the PMOS transistors is connected to the supply. The subthreshold leakage can be reduced by increasing the threshold voltages by reverse biasing the substrate, connecting the n-well to more than 0V and connecting the p-well to less than Vdd. This requires charge pump circuitry to change the voltage, additional power rails, and a twin well or triple well process [47]. The advantage of reverse body bias is that the state is retained. Reverse body bias is less effective for reducing leakage in shorter channel transistors, for example providing a 4 \times reduction in leakage in 0.18um technology and 2.5 \times reduction in 0.13um [116], making it a less useful technique in deeper submicron technologies.

An alternate method of reverse body bias is to connect both the NMOS transistor source and well to a virtual ground V_{ss} (see Figure 2.5) which is raised to reduce leakage in standby. This avoids the need for charge pump circuitry and twin well or triple well process [47]. To avoid losing state information, the reduction in $V_{dd} - V_{ss}$ must be limited by circuitry to regulate the voltage [46]. Reducing $V_{dd} - V_{ss}$ also helps reduce the leakage. This reverse body bias and voltage collapse approach gave a 28 \times reduction in leakage in the 0.18um XScale with minimal area penalty [47]. Returning from “drowsy” mode took 20us, corresponding to 18,000 cycles at 800MHz, as the phase-locked loop (PLL) was also turned off to limit power consumption. In comparison, using sleep transistors in the XScale would have reduced leakage by only about 5 \times , if power gating was not applied to latches and other memory elements that need to retain state, as they comprise about a sixth of the total transistor width [47].

Power gating with sleep transistors to disconnect the supply and/or ground rail (Figure 2.5) can provide more than an order of magnitude leakage reduction in circuitry that uses leaky low V_{th} transistors on critical paths and high V_{th} sleep transistors [150]. This is often referred to as MTCMOS, multi-threshold voltage CMOS. The “virtual” supply and “virtual” ground rails, which are connected to the actual power rails via sleep transistors, may be shared between logic gates to reduce the area overhead for the sleep transistors. Disconnecting the power rails results in loss of state, unless registers contain a latch connected to the actual supply and ground rails [150]. Registers also have connections to the virtual supply and virtual ground rails to limit leakage.

Leakage was reduced by 37 \times in a 0.13um 32-bit arithmetic logic unit (ALU) using PMOS sleep transistors at the expense of a 6% area overhead and 2.3% speed decrease [214]. The leakage was reduced 64 \times by using reverse body bias in conjunction with PMOS sleep transistors in sleep mode, and forward body bias in active mode reduced the speed penalty to 1.8%. The total area overhead for the sleep transistors and the body bias circuitry was 8%. Using sleep transistors saved power if the ALU was idle for at least a hundred clock cycles. Two clock cycles were

required to turn the transistors back on from sleep mode, and four cycles were required to change from reverse body bias. With only forward body bias, Vdd could be reduced from 1.32V to 1.28V with no speed penalty, and leakage was reduced by 1.9 \times at zero bias in standby [214].

2.5.4.1 What's the problem?

Reducing leakage via state assignment, substrate biasing, reducing supply voltage, and sleep transistors requires architectural or software level signals to specify when units will be idle for many cycles. Benchmark applications may not have long idle periods. Without long idle times for units, these methods will not provide any power savings when the chip is active. These techniques cannot be automated at the gate level and require architectural level support for signals to enter and exit standby over multiple cycles.

For state assignment, registers that retain data instead output a 0 or 1 in sleep mode. For registers that don't retain data in standby, extra circuitry is required if the reset output differs from the low leakage state output. These registers are larger and consume more power than a standard register.

Substrate biasing and reducing the supply voltage require a variable supply voltage from a voltage regulator. The cell libraries need to have delay, dynamic power, leakage power and noise immunity characterized at the different supply and substrate voltages. If functional units enter standby at different times, additional power rails may be required and wells biased at different potentials must be spatially isolated. These techniques are often used in low power custom designs, but are complicated to implement in ASICs.

There is a voltage drop across sleep transistors when they are on, degrading the voltage swing for logic. Wider sleep transistors degrade the voltage swing less, but have higher capacitance. Power up of sleep transistors takes substantial energy due to their large capacitance [150]. Standard cells must be characterized for the degraded supply voltage. Layout tools must cluster the gates that connect to the same virtual power rail that is disconnected by a given sleep signal, as having

individual sleep transistors in each gate is too area expensive. As the registers that retain state connect to the virtual power rails and directly to the power rails, the standard cell rows on which registers are placed must be taller to accommodate the extra rails. The virtual and actual supply and ground voltages differ in standby. Thus, the substrates of the transistors connected to virtual power rails and those connected directly to the power rails are at different voltages and must be isolated spatially, increasing the area overhead. The floating output of a power-gated cell can cause large currents if it connects directly to a cell which is not power gated, so additional circuitry is required to drive the output of the power-gated cell [221].

2.5.4.2 What can we do about it?

ASICs seldom use standby power reduction techniques other than full power down, but there is now better tool support for power gating. An EDA flow with power gating can provide two orders of magnitude reduction in leakage if state is not retained, at the cost of 10% to 20% area overhead and 6% higher delay [67]. The ARM1176JZ-S synthesizable core supports dynamic voltage scaling, allowing the supply voltage to be scaled from 1.21V to 0.69V in the 0.13um process, but this requires additional hardware support [77].

To date state assignment and reverse substrate biasing have not been implemented in an EDA methodology. As state assignment cannot be effectively used with combinational logic that is power gated and provides far less leakage reduction than using sleep transistors, it is unlikely to be useful except for circuits that have only short idle periods, on the order of tens of clock cycles. Substrate biasing nicely complements power gating with forward body bias reducing the delay penalty for voltage drop across the sleep transistors, and with reverse body bias reducing the leakage in registers that are on to retain state information. As reverse substrate bias is less effective at shorter channel lengths, ASICs may have from 4 \times higher standby leakage than custom designs that use reverse body bias in 0.18um to 2 \times worse than custom in deeper submicron technologies.

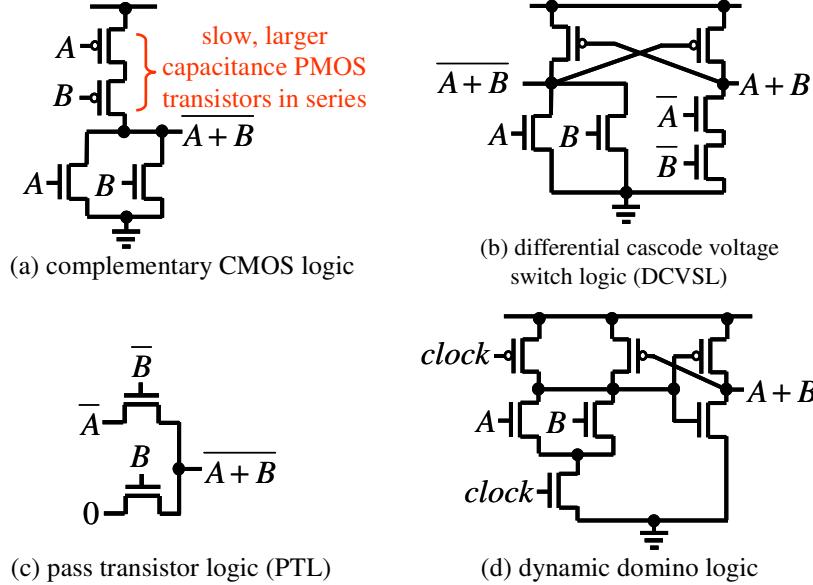


Figure 2.6 This figure shows NOR2 logic gate implementations in different logic styles. The domino logic output is inverted, so that after precharging the inputs to domino logic gates are low to avoid them being discharged until an input transition to high occurs [167].

2.5.5 Logic style

ASICs almost exclusively use complementary static CMOS logic for combinational logic, because it is more robust to noise and Vdd variation than other logic styles. Pass transistor logic (PTL), dynamic domino logic and differential cascode voltage switch logic (DCVSL) are faster than complementary static CMOS logic. These logic styles are illustrated in Figure 2.6. Complementary CMOS logic suffers because PMOS transistors are roughly 2× slower than NMOS transistors of the same width, which is particularly a problem for NOR gates. With the two PMOS transistors in series in Figure 2.6(a), the PMOS transistors must be sized about 4× larger for equal rise and fall delays, substantially increasing the load on the fanins. The high speed logic styles can be used to reduce the critical path delay, increasing performance. Alternatively, the additional timing slack can be used to achieve lower power at high performance targets. Complementary CMOS logic is lower power than other logic styles when high performance is not required. Hence, low power custom designs primarily use complementary CMOS, with faster logic only on critical paths. ASIC designs are mapped to slower, purely complementary CMOS logic standard cell libraries.

The StrongARM used primarily complementary CMOS, with static DCVSL to implement wide NOR gates [144]. In the custom IDCT multiplier, the carry and sum of the full adder cells are both on the critical path [240]. A complementary CMOS gate generated the carry out, and a static DCVSL gate generated the sum. This full adder was 37% faster than a purely complementary CMOS mirror adder.

The StrongARM and XScale used some dynamic logic. Dynamic DCVSL (dual rail domino logic) has twice the activity of single rail domino logic. The Samsung Halla used dynamic DCVSL and is higher power than the complementary CMOS ARM1020E at 400MHz. However, the Halla runs at up to 1.2GHz, while the ARM1020E is limited to 400MHz [136]. Zlatanovici [246] compared 0.13um single rail domino and complementary static CMOS 64-bit adders. Domino could achieve as low as 6.8 FO4 delays at 34pJ/cycle. The fastest static CMOS version was 12.5 FO4 delays, but only 18pJ/cycle.

PTL is a high speed and low energy logic style [18]. In a 0.6um study, a complementary CMOS carry-lookahead 32-bit adder was 20% slower than complementary PTL, but the complementary CMOS adder was 71% lower power [245]. At maximum frequency in 0.25um, a complementary CMOS 3-input XOR ring oscillator had 1.9 \times delay and 1.3 \times power compared to versions in PTL and DCVSL [122]. The XScale ALU bypass adder was implemented in PTL. At 1.1V, this was 14% slower than single rail domino, but it has no precharge and lower switching activity [45].

High speed logic styles can increase the speed of combinational logic by 1.5 \times [38]. We analyzed the potential power savings with reduced combinational logic delay using the pipeline model in Section 3.5.1. We optimistically assumed no extra power consumption for using a high speed logic style on critical paths. At a tight performance constraint, pipelines with only complementary static CMOS combinational logic had up to 2.0 \times higher energy per operation.

2.5.5.1 What's the problem?

PTL, DCVSL, and dynamic logic libraries are used as in-house aids to custom designers. Standard cell libraries with these logic styles are not available to ASIC designers. All of these logic styles are less robust than complementary CMOS logic, and have higher leakage power.

Differential cascode voltage switch logic is faster than complementary CMOS logic, but is higher energy [18][43]. DCVSL requires both input signal polarities and has higher switching activity than complementary CMOS logic. Static DCVSL has cross-coupled outputs, resulting in longer periods of time with a conducting path from supply to ground and larger short circuit current. The DCVSL inputs and their negations must arrive at the same time to limit the duration of the short circuit current, requiring tight control of the layout to ensure similar signal delays.

Dynamic logic is precharged on every clock cycle, increasing the clock load, activity, and dynamic power. The precharged node may only be discharged once, so glitches are not allowed. Shielding may be required to prevent electromagnetic noise due to capacitive cross-coupling discharging the precharged node. To avoid leakage through the NMOS transistors discharging the node, a weak PMOS transistor is required as a “keeper” [236]. There can be charge sharing between dynamic nodes or on PTL paths.

Pass transistor logic suffers a voltage drop of V_{th} across the NMOS pass transistor when the input voltage is high [167]. Consequently, the reduced voltage output from PTL may need to be restored to full voltage to improve the noise margin and to avoid large leakage currents in fanouts. The voltage drop can be avoided by using a complementary PMOS transistor in parallel with the NMOS transistor, but this increases the loading on the inputs, reducing the benefit of PTL. Buffering is needed if the fanins and fanouts are not near the PTL gates, and an inverter may be needed to generate a negated input.

Using these logic styles requires careful cell design and layout. A typical EDA flow gives poor control over the final layout, thus use of these logic styles would result in far more yield problems and chip failures.

2.5.5.2 What can we do about it?

The foundry requirement of high yield means that the only standard cell libraries available to ASIC designers will continue to be robust complementary static CMOS logic. Thus an EDA design flow cannot reduce the power gap for logic style.

An alternative is for designers to adopt a semi-custom design flow: high speed custom cells and manual layout can be used for timing critical logic; or custom macros can be used.

2.5.6 Logic design

Logic design refers to the topology and the logic structure used to implement datapath elements such as adders and multipliers. Arithmetic structures have different power and delay trade-offs for different logic styles, technologies, and input probabilities.

2.5.6.1 What's the problem?

Custom designers tend to pay more attention to delay critical datapaths. Specifying logic design requires carefully structured RTL and tight synthesis constraints. For example, we found that flat synthesis optimized out logic that reduced switching activity in multiplier partial products [111], so the scripts were written to maintain the multiplier hierarchy during synthesis. The reduced switching activity reduced the power-delay product by 1.1 \times for the 64-bit multiplier.

Careful analysis is needed to compare alternate algorithmic implementations for different speed constraints. For example, high-level logic transition analysis showed that a 32-bit carry lookahead adder had about 40% lower power-delay product than carry bypass or carry select adders [30]. There was also a 15% energy difference between 32-bit multipliers. Zlatanovici compared 64-bit

domino adders in 0.13um, and found that the radix-4 adders achieved smaller delay and about 25% lower energy than radix-2 [246].

We estimate that incomplete evaluation of logic design alternatives may result in 1.2 \times higher power for a typical ASIC.

2.5.6.2 What can we do about it?

Synthesis tools can compile to arithmetic modules. The resulting energy and delay is on par with tightly structured RTL. In general, ASIC designers should be able to fully exploit logic design.

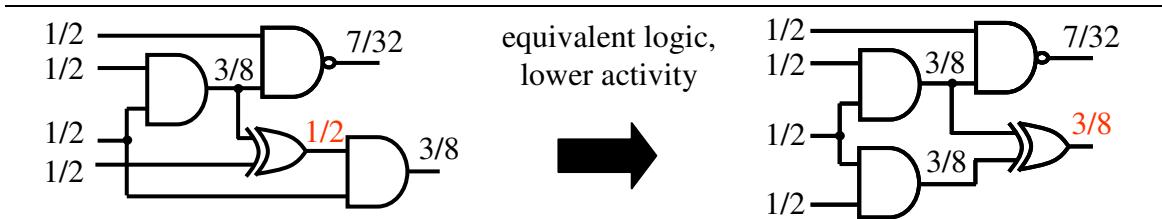


Figure 2.7 This figure illustrates how refactoring logic can reduce the switching activity while giving the same functional result. Switching activities are annotated on the diagram, as propagated from independent inputs that have equal probability of being zero or one.

2.5.7 Technology mapping

In technology mapping a logical netlist is mapped to a standard cell library in a given technology. Different combinations of cells can be used to implement a gate with different activity, capacitance, power and delay. For example to implement an XOR2, an AO22 with inverters¹ may be smaller and lower power, but slower. Refactoring can reduce switching activity (see Figure 2.7). Common sub-expression elimination reduces the number of operations. Balancing path delays and reducing the logic depth decreases glitch activity. High activity nets can be assigned to gate pins with lower input capacitance. [147][182]

¹ An AO22 logic gate computes $ab + cd$, where a, b, c and d are the four inputs. Thus, a two-input XOR may be implemented by $a\bar{b} + \bar{a}b$.

2.5.7.1 What's the problem?

While there are commercial tools for power minimization, power minimization subject to delay constraints is still not supported in the initial phase of technology mapping. Minimizing the total cell area minimizes circuit capacitance, but it can increase activity. For a 0.13um 32-bit multiplier after post-synthesis power minimization, the power was 32% higher when using minimum area technology mapping. This was due to more (small) cells being used, increasing activity. We had to use technology mapping combining delay and area minimization targets for different parts of the multiplier. Technology mapping for low power may improve results; without this and other low power technology mapping techniques, ASICs may have 1.4× higher power than custom.

2.5.7.2 What can we do about it?

Power minimization tools do limited remapping and pin reassignment, along with clock gating and gate sizing [203]. These optimizations are applied after technology mapping for minimum delay, or minimum area with delay constraints. EDA tools should support technology mapping for minimum power with delay constraints. This requires switching activity analysis, but it is not otherwise substantially more difficult than targeting minimum area.

For a given delay constraint, technology mapping can reduce the power by 10% to 20%, for about a 10% to 20% increase in area [147][182]. Low power encoding for state assignment can also give 10% to 20% power reduction [218]. Logic transformations based on logic controllability and observability relationships, common sub-expression elimination, and technology decomposition can give additional power savings of 10% to 20% [164]. Pin assignment can provide up to 10% dynamic power savings by connecting higher activity inputs to gate input pins with lower capacitance [178].

ASICs should not lag custom power consumption due to technology mapping, if better EDA tool support is provided.

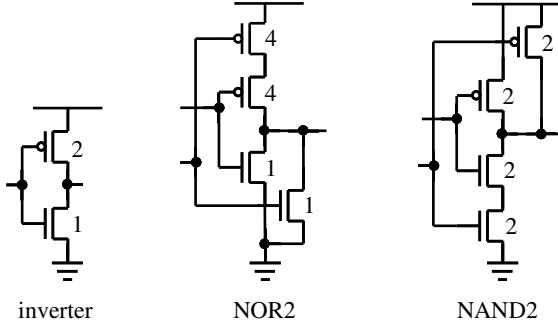


Figure 2.8 This figure shows the relative NMOS and PMOS transistor widths for equal rise and fall delays in different logic gates of equivalent drive strength.

2.5.8 Gate sizing and wire sizing

Wires and transistors should be sized to ensure correct circuit operation, meet timing constraints, and minimize power consumption. ASICs must choose cell sizes from the range of drive strengths provided in the library. ASIC wire widths are usually fixed. Downsizing transistors gives a linear reduction in their capacitance and thus dynamic power, and also gives a linear reduction in leakage. Reducing the wire width gives a linear reduction in wire capacitance but a linear increase in wire resistance, increasing signal delay on the wire.

2.5.8.1 What's the problem?

There is a trade-off between power and delay with gate sizing. To reduce delay, gates on critical paths are upsized, increasing their capacitance. In turn, their fanin gates must be upsized to drive the larger capacitance. This results in oversized gates and buffer insertion on the critical paths. Delay reductions come at the price of increasingly more power and worse energy efficiency.

To balance rise and fall delays, an inverter has PMOS to NMOS width ratio of about 2:1 as a PMOS transistor has about half the drain current of a NMOS transistor of the same width. Accounting for the number of transistors in series, other logic gates also have 2:1 P/N ratio to balance rise and fall delays for an inverter of equivalent drive strength, as illustrated in Figure 2.8. However, to minimize the average of the rise delay and fall delay, the P/N ratio for an inverter should be about 1.5:1 [86]. Reducing the P/N ratio provides a small reduction in delay and a substantial reduction in power consumption, by reducing the capacitance of the larger PMOS

transistors. The optimal P/N ratio to minimize the delay is larger for larger loads [170]. In addition, sometimes the rise and fall drive strengths needed are different – for example, the rising output transition from a cell may be on a critical path, but the falling transition may not be critical.

The ratio of pullup to pulldown drive strength determines at what input voltage a gate switches from low to high or high to low [236]. Equal rise and fall delays maximize the noise margin for a high or low input. Thus skewing the P/N ratio reduces the noise margin. Ideally, standard cell libraries should provide a range of drive strength skews and lower power cells with reduced P/N ratio, but often only cells with equal rise and fall drive strength are available to ensure high yield.

A design-specific standard cell library developed for the iCORE [170] gave a 20% speed increase by using reduced P/N width ratio, and by using larger transistor widths to increase drive strength instead of buffering. The larger transistor widths required increased cell height, but the net impact on layout area was minimal as they were only used in the most critical paths. However, the design time for this library was about two worker years.

Custom libraries may be finer grained, which avoids oversizing gates, and have skewed drive strengths. Cells in datapath libraries are denser and have smaller input capacitance [38]. Specific cell instances can be optimized. Cells that connect to nearby cells don't need guard-banding. This avoids the need for buffering to handle driving or being driven by long wires.

Wire widths can also be optimized in custom designs. Gong et al. [74] optimized global clock nets on a 1.2um chip. By simultaneously optimizing buffer and wire sizes, they reduced the clock net power by about 63%. This amounts to a 10% to 20% saving in total power.

The basic approach to gate sizing in commercial EDA software has changed little in the past 20 years. These gate sizers like TILOS [64] proceed in a greedy manner, picking the gate with the best power or area versus delay tradeoff to change, and iterating. There are known circuit examples where these approaches perform suboptimally, but it has not been clear how much of a

problem this is for typical circuits for real world applications. We found power savings of up to 32.3% versus gate sizing in Design Compiler, which is commonly used in EDA flows for circuit synthesis, and 16.3% savings on average across the ISCAS'85 benchmarks and three typical datapath circuits (see Section 4.6.3). Gate sizing is an NP-complete problem, but circuit sizes are large and optimization software must have runtimes of $O(n^2)$ or less to be of practical use [195], where n is the number of gates in the circuit. The TILOS-like greedy approaches are relatively fast, being $O(n^2)$, and other approaches that perform better with similar static timing analysis (STA) accuracy have had worse computational complexity.

Some commercial power minimization software has only recently provided the option of minimizing the total power. Previously, the user had to prioritize minimizing either the dynamic power or the leakage power, which can be suboptimal.

We estimate that these limitations in gate sizing and wire sizing for typical ASICs may lead to a power gap of 1.6 \times versus custom.

2.5.8.2 What can we do about it?

Gate downsizing to reduce power consumption is well supported by power minimization tools. Some commercial tools support clock tree wire sizing, but there are no commercial tools available for sizing other wires. Automated cell creation, characterization and in-place optimization tools are available. Standard cell libraries with finer grained drive strengths and lower power consumption are available, though users may be charged a premium.

We synthesized the base configuration of a Tensilica Xtensa processor in 0.13um. The power/MHz was 42% lower and the area was 20% less at 100MHz than at the maximum clock frequency of 389MHz, due to using smaller gates and less buffers. If delay constraints are not too tight, tools can reduce power by gate downsizing without impacting delay. At 325MHz, Power

Compiler was able to reduce the power consumption by 26% and reduce the area by 12% for no delay penalty.

Libraries with fine granularity help to reduce the power by avoiding use of oversized gates. In a 0.13um case study of digital signal processor (DSP) functional macros, using a fine grained library reduced power consumption by 13% [196].

After place and route when wire lengths and capacitive loads are accurately known, in place optimization can remove guard banding where it is unnecessary. ASIC designers have tended to distrust this approach, as the optimized cells without guard banding cannot be safely used at earlier stages in the EDA flow. Skewing the pullup to pulldown drive strength to optimize the different timing arcs through a gate can also improve energy efficiency. A prototype tool flow for in place cell optimization increased circuit speed by 13.5% and reduced power consumption by 18%, giving a 1.4 \times increase in energy efficiency for the 0.35um 12,000 gate bus controller [50]. 300 optimized cells were generated in addition to the original standard cell library that had 178 cells.

Our linear programming gate sizing approach discussed in Chapter 4 takes a global view of the circuit rather than performing greedy “peephole” optimization. We achieved up to 32.3% power savings and on average 16.3% power savings versus gate sizing in Design Compiler for the combinational netlists. Our optimization approach has between $O(n)$ and $O(n^2)$ runtime growth, making it scalable to large circuit sizes.

ASICs may have 1.1 \times worse power than custom due to gate and wire sizing, as wire sizing tools are not available other than for the clock tree, and some design-specific cell optimizations are not possible without custom cell design, beyond what is possible with automated cell creation.

2.5.9 Voltage scaling

Reducing the supply voltage Vdd quadratically reduces switching power. Short circuit power also decreases with Vdd. Reducing Vdd also reduces leakage. For example, with our 0.13um library leakage decreases by a factor of three as Vdd is decreased from 1.2V to 0.6V. As Vdd decreases, a gate's delay increases. To reduce delay, threshold voltage Vth must also be scaled down. As Vth decreases, leakage increases exponentially. Thus there is a tradeoff between performance, dynamic power and leakage power.

Ideally, we want to operate at as low Vdd as possible, with Vth high enough to ensure little leakage. For example, dynamic scaling of the supply voltage from 3.8V to 1.2V gives a 10× increase in energy efficiency at the price of decreasing performance by a factor of 14 for Burd's 0.6um ARM implementation [27]. Reducing the power consumption in this manner requires timing slack.

Power consumption may be reduced by using multiple supply voltages and multiple threshold voltages. High Vdd and low Vth can be used on critical paths to reduce their delay, while lower Vdd and higher Vth can be used elsewhere to reduce dynamic and leakage power.

2.5.9.1 What's the problem?

Custom designs can achieve at least twice the speed of ASICs with high performance design techniques [38]. At the same performance target as an ASIC, a custom design can reach lower Vdd using the additional timing slack. Compare Vdd of Burd, StrongARM and XScale to other ARMs in Table 2.1. With lower Vdd they save between 40% and 80% dynamic power versus other ARMs in the same technology. This is the primary reason for their higher energy efficiency. To use lower Vdd, ASICs must either settle for lower performance or use high speed techniques, such as deeper pipelining, to maintain performance.

Dynamically adjusting the supply voltage for the desired performance requires a variable voltage regulator and takes time, during which correct signals must be maintained to avoid transitioning into illegal states from which behavior is unknown. To change from 1.2V to 3.8V in Burd's ARM [27] required energy equal to that consumed in 712 cycles of peak operation, and there was a delay of 70us. Increasing or decreasing the supply voltage by 800mV took 50us in the XScale [45].

Several barriers remain to ASICs using low Vdd. Using lower Vdd requires lower Vth to avoid large increases in gate delay. Vth is determined by the process technology. A foundry typically provides two or three libraries with different Vth: high Vth for low power; and low Vth for high speed at the expense of significant leakage power. Most ASIC designers cannot ask a foundry to fine tune Vth for their particular design, even if an intermediate Vth might be preferable to reduce leakage. Vdd can be optimized for ASICs, but typical ASIC libraries are characterized at only two nominal supply voltages – say 1.2V and 0.9V in 0.13um. To use Vdd of 0.6V, the library must be re-characterized. There is also less noise immunity at lower Vdd.

Use of multiple supply voltages either requires that the wells of PMOS transistors in low Vdd gates are reverse biased by connecting them to high Vdd, or spatial isolation between the wells connected to low Vdd and high Vdd. Layout tools must support these spacing constraints. Low voltage swing signals must be restored to full voltage swing with a voltage level converter to avoid large leakage currents when a high Vdd gate is driven by a low Vdd input. Most level converter designs require access to both high Vdd and low Vdd, which complicates layout and may require that they straddle two standard cell rows, additionally the PMOS wells connected to different Vdd must be spatially isolated. Voltage level converters are not available in ASIC libraries. Synthesis and optimization tools must insert level converters where needed, and prevent low Vdd gates driving high Vdd gates in other cases.

If voltage level converters are combined with the flip-flops, the power and delay overheads for voltage level restoration are less. Due to the additional power and delay overheads for asynchronous level converters (those not combined with flip-flops), there have been reservations about whether they provide any practical benefits over only using level converter flip-flops [222]. There has also been concern about their noise immunity [110].

Multi-Vdd circuitry has more issues with capacitive cross-coupling noise due to high voltage swing aggressors on low voltage swing lines. Thus it may be best to isolate high Vdd and low Vdd circuitry into separate voltage islands, rather than using multi-Vdd at a gate level. Multi-Vdd at the gate-level can also require additional voltage rails. Gate level multi-Vdd requires tool support to cluster cells of the same Vdd to achieve reasonable layout density. An additional voltage regulator is needed to generate the lower Vdd.

Using multiple threshold voltages is expensive. Each additional PMOS and NMOS threshold voltage requires another mask to implant a different density of dopants, which substantially increases processing costs. A set of masks costs on the order of a million dollars today and an additional V_{th} level increases the fabrication cost by 3% [165]. Each additional mask also increases the difficulty of tightly controlling process yield, motivating some manufacturers to limit designs to a single NMOS and single PMOS threshold voltage.

To take full advantage of multiple threshold voltages within gates, standard cells with multi- V_{th} and skewed transistor widths must be provided. High V_{th} can be used to reduce leakage while low V_{th} can be used to reduce dynamic power. For example, using low V_{th} PMOS transistors and high V_{th} NMOS transistors in a complementary CMOS NOR gate, as leakage is less through the PMOS transistors that are in series. In gates that have an uneven probability of being high or low, there is more advantage to using high V_{th} to reduce leakage for the pullup or pulldown network that is more often off. Similarly, for wider transistors with high V_{th} may be preferable for gates

that have low switching activity, while narrower transistors with low V_{th} is better when there is higher switching activity.

2.5.9.2 What can we do about it?

There are tools to automate characterizing a library at different V_{dd} operating points. Characterization can take several days or more for a large library. Standard cell library vendors can help by providing more V_{dd} characterization points. Commercial tools do not adequately support multi-V_{dd} assignment or layout, but separate voltage islands are possible.

There are voltage level converter designs that only need to connect to high V_{dd} [196]. Some asynchronous level converters designs have been shown to be robust and have good noise immunity in comparison to typical logic gates at low V_{dd} [123]. It would help if voltage level converters were added to standard cell libraries.

Foundries often support high and low V_{th} cells being used on the same chip. Power minimization tools can reduce power by using low V_{th} cells on the critical path, with high V_{th} cells elsewhere to reduce leakage. Combining dual V_{th} with sizing reduced leakage by 3 to 6× for a 5% increase in delay on average versus using only low V_{th} [183]. From a design standpoint, an advantage of multiple threshold voltages is that changing the threshold voltage allows the delay and power of a logic gate to be changed without changing the cell footprint, and thus not perturbing the layout. As discussed in Chapter 7, optimization with multiple threshold voltages is straightforward, providing those cells are provided in the library. Optimization runtime increases at worst linearly with the number of cells in the library.

Geometric programming optimization results on small benchmark circuits in Chapter 6 suggest that multi-V_{dd} and multi-V_{th} may only offer 20% power savings versus optimal choice of single V_{dd}, single NMOS V_{th}, and single PMOS V_{th}. As ASIC designers are limited to V_{th} values specified by the foundry, there may be more scope for power savings in ASICs when V_{th} is

suboptimal. After scaling Vdd from 1.2V to 0.8V by using a low Vth of 0.08V, we found power savings of up to 26% by using a second higher Vth to reduce leakage with our linear programming optimization approach in Chapter 7, and average power savings were 16%. We found that power savings with gate-level multi-Vdd were generally less than 10%. Using multi-Vdd is more appropriate at a module level, making a good choice of a single supply voltage for the module based on the delay of critical paths.

With 9% timing slack versus the maximum clock frequency, Stok et al. [196] reduced power consumption by 31% by scaling from Vdd of 1.2V to Vdd of 1.0V. Usami et al. [223] implemented automated tools to assign dual Vdd and place dual Vdd cells, with substrate biasing for the transistors to operate at low Vth in active mode to increase performance and high Vth in standby mode to reduce leakage. They achieved total power reduction of 58% with only a 5% increase in area. The ARM1176JZ-S [77] synthesizable core supports dynamic voltage scaling, but this requires additional software and hardware support. This demonstrates that ASICs can use such methods with appropriately designed RTL, software, and EDA tool support, reducing the power gap due to voltage scaling alone to 1.0 \times .

2.5.10 Floorplanning, cell placement and wire routing

The quality of floorplanning of logic blocks and global routing for wires, followed by cell placement and detailed wire routing, have a significant impact on wire lengths. A significant portion of the capacitance switched in a circuit is wiring capacitance. The power consumption due to interconnect is increasing from about 20% in 0.25um to 40% in 0.09um [200]. Wire lengths depend on cell placement and congestion. Larger cells and additional buffers are needed to drive long wires. We estimate that poor floorplanning, cell sizing and cell placement with inaccurate wire load models can result in 1.5 \times worse power consumption in ASICs compared to custom.

2.5.10.1 What's the problem?

Custom chips are partitioned into small, tightly placed blocks of logic. Custom datapaths are manually floorplanned and then bit slices of layout may be composed. Automatic place and route tools are not good at recognizing layout regularity in datapaths.

We used BACPAC [200] to examine the impact of partitioning. We compared partitioning designs into blocks of 50,000 or 200,000 gates in 0.13um, 0.18um, and 0.25um. Across these technologies, using 200,000 gate blocks increased average wire length by about 42%. This corresponds to a 9% to 17% increase in total power. The delay is also about 20% worse with larger partitions [38]. The net increase in energy per operation is 1.3 to 1.4 \times .

When sizing gates and inserting buffers, the first pass of synthesis uses wire load models to estimate wire loads. Wire load models have become increasingly inaccurate, with wires contributing a larger portion of load capacitance in the deep submicron. A conservative wire load model is required to meet delay constraints, but this results in most gates being over-sized [38], making the power higher.

Physical synthesis iteratively performs placement and cell sizing, to refine the wire length estimates. Cell positions are optimized then wire lengths are estimated with Steiner trees. Steiner tree wire length models used by physical synthesis are inaccurate if a wire route is indirect. There can be too many critical paths to give them all a direct route. Power minimization increases path delay, so more paths are critical, increasing congestion. This may degrade performance. For example for the base configuration of Tensilica's Xtensa processor for a tight performance target of 400MHz clock frequency in 0.13um, we found that the clock frequency was 20% worse after place and route when power minimization was used. The tool flow used Synopsys Design Compiler for synthesis, Synopsys Physical Compiler for cell placement and power minimization, and Cadence Silicon Ensemble for routing.

2.5.10.2 What can we do about it?

Physical synthesis, with iteratively refined wire length estimates and cell placement, produces substantially better results than a tool flow using only wire load models. In our experience, physical synthesis can increase speed by 15% to 25%. The cell density (area utilization) increases, reducing wire lengths, and then cells may be downsized, which reduces power by 10% to 20%.

Earlier power minimization tools often ended up increasing the worst critical path delay after layout if the delay constraint was tight. This is less of a problem in today's tools, where power minimization is integrated with physical synthesis. Tool flow integration has also improved, particularly as some of the major CAD software vendors now have complete design flows with tools that perform well throughout the design flow – rather than using for example Synopsys tools for synthesis and Cadence tools for place and route.

An ASIC designer can generate bit slices from carefully coded RTL with tight aspect ratio placement constraints. Bit slices of layout may then be composed. With bit slices, Chang showed a 70% wire length reduction versus automated place-and-route [33], which would give a 1.2 to 1.4 \times increase in energy efficiency. Stok et al. [196] found that bit slicing and some logic optimization, such as constant propagation, improved clock frequency by 22% and reduced power consumption by 20% for seven DSP functional macros implemented in 0.13um, improving the energy efficiency by a factor of 1.5 \times . Compared to bit slicing using a library of datapath cells, manual placement and routing can still achieve smaller wire lengths [33], leaving a gap of about 1.1 \times .

2.5.11 Process technology

After the layout of a chip is verified, it is then fabricated in the chosen process technology by a foundry. Within the same nominal process technology generation, the active power, leakage power, and speed of a chip differ substantially depending on the process technology used to

fabricate the circuit. Older technologies are slower and are cheaper per mask set. However, newer technologies have more dies per wafer and thus may be cheaper per die for larger production runs. Newly introduced technologies may have lower yield, though these problems are typically ironed out as the technology matures [143].

High performance chips on newer technologies have substantially higher subthreshold leakage power as threshold voltage is scaled down with supply voltage to reduce dynamic power. Gate tunneling leakage is also higher as transistor gate oxide thickness is reduced for the lower input voltage to the transistor gate to retain control of the transistor. However, the power consumption and power per unit area can be lower in deeper submicron technologies if performance is not increased [128].

For example in 65nm, Intel's low power P1265 process reduces leakage by a factor of 300, but has 55% lower saturation drain current and hence is roughly 2.2× slower [112], compared to their higher performance P1264 technology [220]. To reduce leakage they increased oxide thickness from 1.2nm to 1.7nm, increased gate length from 35nm to 55nm, and increased threshold voltage from about 0.4V to 0.5V (at drain-source voltage of 0.05V) [112]. Note that the higher threshold voltage results in a greater delay increase if supply voltage is reduced.

While Intel started selling processors produced in 65nm bulk CMOS technology at the start of 2006, AMD is still producing chips in 90nm silicon-on-insulator (SOI) technology [19][85]. AMD is on track to offer 65nm SOI chips in the last quarter of 2006 [160]. Intel is a technology generation ahead, and has the cost advantage of using cheaper bulk CMOS and more dies per wafer with its smaller technology. However, SOI has better performance per watt than bulk CMOS, so Intel has only a slight advantage in terms of performance and energy efficiency.

In the same nominal technology generation, there are substantial differences between available technologies. Different technology implementations differ by up to 25% in speed [38], 60% in

dynamic power, and an order of magnitude in leakage. We compared several gates in Virtual Silicon's IBM 8SF and UMC L130HS 0.13um libraries. 8SF has about 5% less delay and only 5% of the leakage compared to L130HS, but it has 1.6 \times higher dynamic power [228]. Our study of two TSMC 0.13um libraries with the base configuration of Tensilica's Xtensa processor showed that TSMC's high V_{th} , low-k library was 20% lower power/MHz, with 66% less leakage power and 14% less dynamic power, than the low V_{th} , low-k library (see Table 2.5).

The power consumption, RC delays¹, and IR drop² in the wires can be reduced by use of copper wires and low-k interlayer dielectric insulator. Copper interconnect has 40% lower resistivity than aluminum. Low-k dielectrics of 2.7 to 3.6 electrical permittivity (k) are used in different processes, compared to SiO_2 's dielectric constant of 3.9. Using low-k interlayer dielectric insulator reduces interconnect capacitance by up to 25%, reducing dynamic power consumption by up to 12%. High-k transistor gate dielectrics increase the transistor drive strength and thus speed, and can also reduce the gate tunneling leakage by an order of magnitude [133].

Narendra et al. showed that silicon-on-insulator (SOI) was 14% to 28% faster than bulk CMOS for some 0.18um gates. The total power was 30% lower at the same delay, but the leakage power was $\times 1.2$ to $\times 20$ larger [152]. A 0.5um DSP study showed that SOI was 35% lower power at the same delay as bulk CMOS [181]. Double-gated fully depleted SOI is less leaky than bulk CMOS.

Table 2.5 Dynamic and leakage power consumption for two different 0.13um TSMC libraries for Tensilica's Xtensa processor for the base configuration with a clock frequency of 100MHz.

Library	low Vdd, low k-dielectric	low Vdd, low k-dielectric, high V_{th}
Dynamic power (uW)	6.48	5.66
Leakage power (mW)	0.67	0.25
Total power (mW)	7.15	5.90

¹ The wire RC delay for a signal to propagate on a wire is given by the resistance (R) of the wire multiplied by the capacitance of the wire and the capacitance of the load (C). For more detail, see Appendix E.5.

² Voltage drop in a wire is give by current (I) \times resistance (R). Voltage drop in the power grid reduces the supply voltage available to logic gates.

In the StrongARM, caches occupied 90% of the chip area and were primarily responsible for leakage. A 12% increase in the NMOS channel length reduced worst case leakage by a factor of 20. Lengthening transistors in the cache and other devices reduced total leakage by $\times 5$ [144]. Transistor capacitance, and thus dynamic power, increases linearly with channel length. Channel length can be varied in ASICs to reduce leakage if such library cells are available.

As a process technology matures, incremental changes can be made to improve yield, improve performance and reduce power consumption. In Intel's 0.25um P856 process the dimensions were shrunk by 5% and, along with other modifications, this gave a speed improvement of 18% in the Pentium II [23]. The 0.18um process for the Intel XScale had a 5% shrink from P858, and other changes to target system-on-chip applications [45]. There was also a 5% linear shrink in Intel's 0.13um P860 process and the effective gate length was reduced from 70nm to 60nm [212]. A 5% shrink reduces transistor capacitance and dynamic power by about 5%. These process improvements are typical of what is available to high volume custom designs.

We estimate that different choices within the same process technology generation may give up to 1.6 \times difference in power.

2.5.11.1 What's the problem?

Standard cells are characterized in a specific process. The cells must be modified and libraries updated for ASIC customers to take advantage of process improvements. Without such updates, 20% speed increase and greater reductions in power may be unavailable to ASIC customers. Finding the lowest power for an ASIC requires synthesis with several different libraries to compare power at performance targets of interest. The lowest power library and process may be too expensive.

2.5.11.2 What can we do about it?

Generally, it requires little extra work to re-target an ASIC EDA flow to a different library. ASICs can be migrated quickly to different technology generations, and updated for process improvements. In contrast, the design time to migrate custom chips is large. Intel started selling 90nm Pentium 4 chips in February 2004 [85], but a 90nm version of the XScale was only reported in June 2005 [169] and is not currently in production to our knowledge. Meanwhile, ARM has synthesized the more recent Cortex-A8 core for 65nm [10]. ASICs should be able to take full advantage of process improvements, closing the gap for process technology to 1.0x.

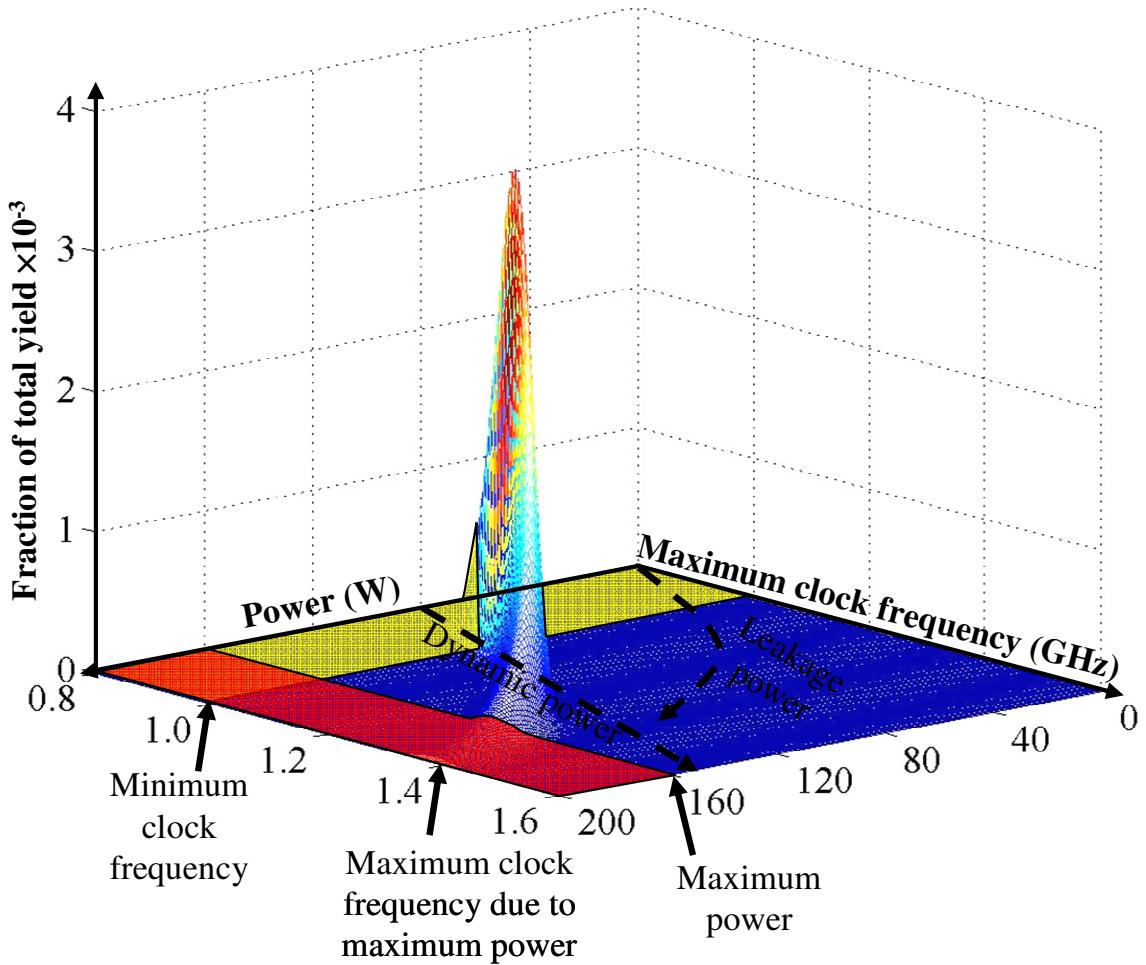


Figure 2.9 This graph illustrates yield versus the maximum clock frequency and total power at that clock frequency for fabricated chips¹. The minimum clock frequency is 1.0GHz and the maximum power consumption is 160W. The maximum clock frequency of about 1.4GHz is determined from the power constraint. 2.3% of the chips are slower than 1.0GHz and 2.2% are faster than 1.4GHz. 10.7% of the chips have power consumption of more than 160W and 2.3% have power consumption of more than 200W.

2.5.12 Process variation

Chips fabricated in the same process technology vary in power and speed due to process variation, as illustrated in Figure 2.9. Some of the chips fabricated may be too slow, while some are significantly faster. In previous technology generations, the faster chips could be sold at a premium. However, faster chips have more leakage power and greater variation in leakage power

¹ Data was generated with a normal frequency distribution of $f = N(1.2, 0.1)$ and distribution for total power of $P = 100f + e^{-10+10f+N(0,0.4)}$, where the dynamic power at clock frequency f is $100f$. The leakage power distribution with clock frequency is similar to that for 0.18um technology in [21].

[21]. Thus the faster chips may consume too much power, particularly if run at a higher clock frequency where dynamic power is also higher as it increases linearly with clock frequency.

There are a number of sources of process variation within a plant, such as optical proximity effects, and wafer defects. The channel length L , transistor width, wire width and wire height have about 25% to 35% variation from nominal at three standard deviations (3σ). Transistor threshold voltage V_{th} and oxide thickness have about 10% variation at 3σ [153]. Decreased transistor oxide thickness substantially increases gate tunneling leakage, and a decrease in V_{th} or L can cause a large increase in subthreshold leakage current, though these transistors are faster. Dynamic power scales linearly with transistor and wire dimensions, as capacitances increase.

To ensure high yield accounting for process variation, libraries are usually characterized at two points. To meet the target speed, the process' worst case speed corner is used – typically 125°C, 90% of nominal V_{dd} , with slow transistors. To prevent excessive power, the active power may be characterized at a worst case power corner, e.g. -40°C, 110% of nominal V_{dd} , and fast transistors. Leakage is worse at high temperature. Due to V_{dd} alone, the active power is 50% higher at the worst case power corner than at the worst case speed corner. These process corners are quite conservative and limit a design. The fastest chips fabricated in a typical process may be 60% faster than estimated from the worst case speed corner [38]. Similarly, examining the distribution of power of fabricated 0.3um MPEG4 codecs [205], the worst case power may be 50% to 75% higher than the lowest power chips produced.

Table 2.6 This table compares the rated power consumption of chips operating at the same clock frequency that are sold by Intel and AMD today [3][4][5][102][103][108][206]. Higher speed parts can operate at a lower supply voltage, reducing the power consumption. These lower power processors are sold at a premium. The 5148 Xeon has yet to be released and the supply voltage has not been stated, though it is likely to be similar to the 1.3V T7600 Core 2 Duo that is used in laptops.

Processor	Model	Codename	Technology (nm)	Frequency (GHz)	Voltage (V)	Power (W)	Power Increase
Athlon 64 X2 Dual-Core	4800+	Windsor	90	2.40	1.25	65	
Athlon 64 X2 Dual-Core	4800+	Windsor	90	2.40	1.35	89	×1.4
Athlon 64 X2 Dual-Core	3800+	Windsor	90	2.00	1.08	35	
Athlon 64 X2 Dual-Core	3800+	Windsor	90	2.00	1.25	65	×1.9
Athlon 64 X2 Dual-Core	3800+	Windsor	90	2.00	1.35	89	×2.5
Athlon 64	3500+	Orleans	90	2.20	1.25	35	
Athlon 64	3500+	Orleans	90	2.20	1.40	62	×1.8
Turion 64	MT-40	Lancaster	90	2.20	1.20	25	
Turion 64	ML-40	Lancaster	90	2.20	1.35	35	×1.4
Core 2 Duo	T7600	Merom	65	2.33	1.30	35	
Xeon 5100 series	5148	Woodcrest	65	2.33	unknown	40	×1.1
Xeon 5100 series	5140	Woodcrest	65	2.33	1.40	65	×1.9
Core 2 Duo	T7200	Merom	65	2.00	1.30	35	
Xeon 5100 series	5130	Woodcrest	65	2.00	1.40	65	×1.9
Core Duo	L2500	Yonah	65	1.83	1.21	15	
Core Duo	T2400	Yonah	65	1.83	1.33	31	×2.1
Core Duo	L2400	Yonah	65	1.66	1.21	15	
Core Duo	T2300	Yonah	65	1.66	1.33	31	×2.1

Exploiting the variation in power consumption, Intel and AMD have been selling lower power chips at a premium. The power consumption of the cheaper, higher power parts is typically up to about 2× that of the low power chips, as shown in Table 2.6. Note that Intel’s Merom (laptop), Conroe (desktop) and Woodcrest (server) chips are essentially the same [97], though voltages, caching strategies and so forth may be changed for lower power but lower performance for the laptop version.

Custom circuitry can be designed to ameliorate process variation in fabricated chips. In the Pentium 4, the clock is distributed across the chip to 47 domain buffers, which each have a 5 bit programmable register to remove skew from the clock signal in that domain to compensate for process variation [127]. A similar scheme was used to reduce clock skew in the 90nm XScale [44]. The body bias can be changed to adjust the transistor threshold voltage, and thus the delay and leakage power. Body bias can be applied at a circuit block level to reduce the standard

deviation in clock frequency between dies from 4.1% to 0.21%, improving speed by 15% versus the slower chips without body bias, while limiting the range in leakage power to $3\times$ for a 0.15um test chip [213]. To do this, representative critical path delays and the leakage current must be measured while the bias is varied. Additional power rails are needed to route the alternate NMOS and PMOS body bias voltages from the body bias generator circuitry, resulting in a 3% area overhead [213]. Forward body bias allowed Vdd to be reduced from 1.43V to 1.37V giving a 7% reduction in total power for a 0.13um 5GHz 32-bit integer execution core [226].

2.5.12.1 What's the problem?

For ASIC parts that are sold for only a few dollars per chip, additional testing for power or speed binning is too expensive. Such ASICs are characterized under worst case process conditions to guarantee good yield. Thus ASIC power and speed are limited by the worst case parts. Without binning, there may be a power gap of $\times 2$ versus custom chips that are binned. Custom chips that have the same market niche as ASICs have the same limitation on testing for binning, unless they are sold at a much higher price per chip.

The complicated circuitry and tight control of layout and routing required to compensate for process variation in a fabricated chip is not possible within an ASIC methodology.

2.5.12.2 What can we do about it?

To account for process variation, ASIC power may be characterized after fabrication. Parts may then be advertised with longer battery life. However, post-fabrication characterization of chip samples does not solve the problem if there is a maximum power constraint on a design. In this case, ASICs may be characterized at a less conservative power corner, which requires better characterization of yield for the standard cell library in that process. For typical applications, the power consumption is substantially less than peak power at the worst case power corner.

Additional steps may be taken to limit peak power, such as monitoring chip temperature and powering down if it is excessive.

We estimate a power gap of up to 1.3 \times due to process variation for ASICs in comparison to custom designs that compensate for process variation, from analysis of a 15% increase in custom speed with the pipeline model in Chapter 3.

2.6 Conclusions

We compared synthesizable and custom ARM processors from 0.6um to 0.13um. We also examined discrete cosine transform cores, as an example of dedicated low power functional units. In these cases, there was a power gap of 3 to 7 \times between custom and ASIC designs.

We have given a top-down view of the factors contributing to the power gap between ASIC and custom designs. From our analysis, the most significant opportunity for power reduction in ASICs is using microarchitectural techniques to maintain performance while reducing power by voltage scaling. Reducing the pipeline delay overhead and using pipelining to increase timing slack can enable substantial power savings by reducing the supply voltage and downsizing gates. Multiple threshold voltages may be used to limit leakage while enabling a lower Vdd to be used. Choosing a low power process technology and limiting the impact of process variation reduces power by a large factor.

In summary, at a tight performance constraint for a typical ASIC design, we believe that the power gap can be closed to within 2.6 \times by using these low power techniques with fine granularity standard cell libraries, careful RTL design and EDA tools targeting low power. The remaining gap is mostly from custom designs having lower pipelining overhead and using high speed logic on critical paths. Using a high speed logic style on critical paths can provide timing slack for significant power savings in custom designs. High speed logic styles are less robust and require careful layout, and thus are not amenable to use in an ASIC EDA methodology.

An example of combining low power and high performance design techniques on DSP functional macros was provided by Puri et al. [165] at the 2003 Design Automation Conference. Corrected results will appear in [196]. To improve performance and reduce power consumption, they used arithmetic optimizations, logic optimization, a finer grained library, voltage scaling from 1.2V to 1.0V, and bit-slicing. Performance improved from 94MHz to 177MHz and energy efficiency increased from 0.89MHz/mW to 2.78MHz/mW – a factor of 3.1×. This demonstrates the power savings that may be achieved by using low power techniques in ASICs.

The next chapter details our power and delay model that incorporates the major factors that contribute to the power gap between ASIC and custom. It includes pipelining, logic delay, voltage scaling and gate sizing. The logic delay is determined by factors such as the logic style, wire lengths after layout, process technology, and process variation which affects the worse case delay.

Chapter 3. Pipelining to Reduce Power

Algorithmic and architectural choices can reduce the power by an order of magnitude [182]. We assume that ASIC and custom designers make similar algorithmic and architectural choices to find a low power implementation that meets performance requirements for the target application.

Circuit designers usually explore trade-offs of different microarchitectural features to implement a given architecture for typical applications. The analysis may be detailed using cycle accurate instruction simulators, but low level circuit optimizations are not usually examined until a much later design phase. High level microarchitectural choices have a substantial impact on the performance and power consumption, affecting the design constraints for low level optimizations.

This chapter examines the power gap between ASIC and custom with pipelining and different architectural overheads. Other researchers have proposed high level pipelining models that consider power consumption, but they do not consider gate sizing and voltage scaling. We will augment a simple pipeline model with a model of power savings from voltage scaling and gate sizing versus timing slack. This enables simultaneous analysis of the power and performance trade-offs for both high-level and low-level circuit optimizations.

Pipelining does not reduce power by itself. Pipelining reduces the critical path delay by inserting registers between combinational logic. Glitches are prevented from propagating across register boundaries, but otherwise logic activity remains the same. However, the clock signal to registers has high activity which contributes to the dynamic power. Pipelining can also reduce the instructions per clock cycle (IPC), due to high branch misprediction penalties and other hazards, and thus can reduce the energy efficiency. The timing slack from pipelining can be used for voltage scaling and gate downsizing to achieve significant power savings (see Figure 3.1 and Figure 3.2).

From our analysis, pipelining contributes up to a factor of $5.1\times$ to the power gap between ASIC and custom at a tight performance constraint. There is no timing slack at a tight performance constraint for the ASIC where additional pipeline stages will reduce performance, but at this point there is still timing slack for voltage scaling and gate downsizing in a custom design. A custom design may also use additional pipeline stages to further improve performance, as pipeline stage delay overheads are less for custom. The power gap is less as the performance constraint is relaxed, reducing the gap to $4.0\times$ at only 7% lower performance. The gap can be reduced to $1.9\times$ if the pipeline stage delay overhead is reduced.

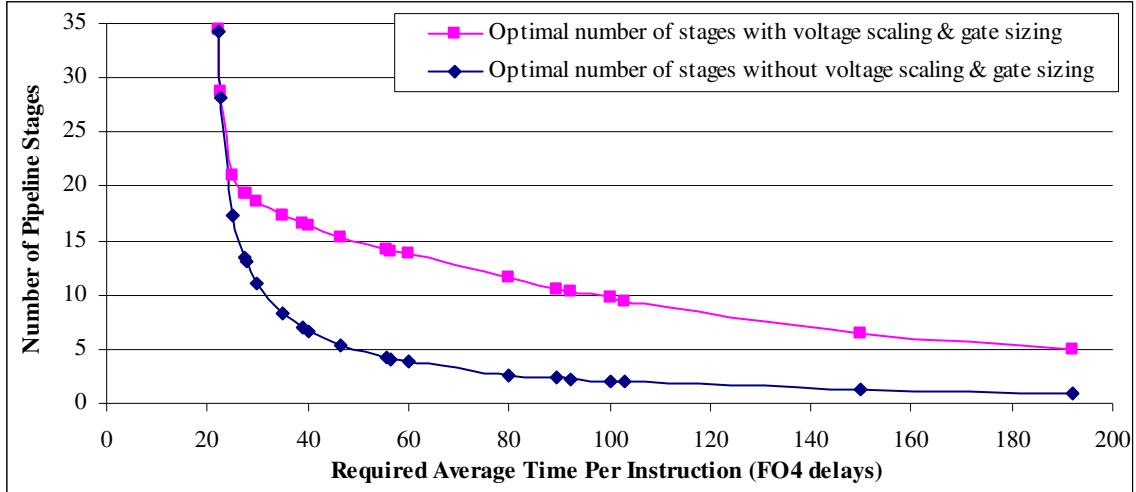


Figure 3.1 The optimal number of pipeline stages to minimize energy/instruction is shown versus the performance constraint. At a tight performance constraint additional stages penalize performance, little timing slack is available, and there is little opportunity for voltage scaling and gate sizing. At more relaxed performance constraints, additional stages provide timing slack for a substantial power reduction with voltage scaling and gate downsizing, as shown in Figure 3.2. These results are for the custom design parameters with our pipeline model.

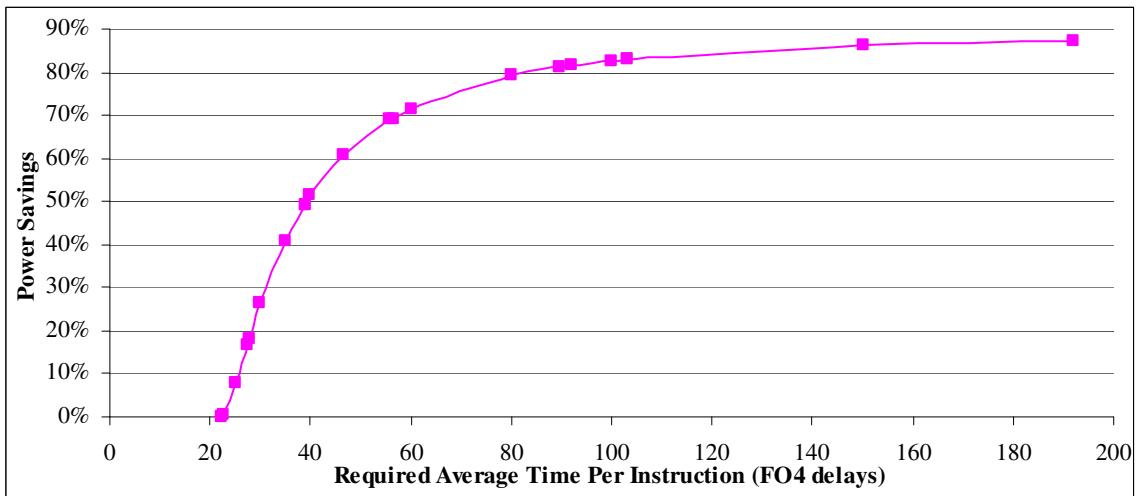


Figure 3.2 Power savings with additional pipeline stages to provide timing slack for voltage scaling and gate sizing versus power consumption without these methods and fewer pipeline stages. These results are for the custom design parameters with our pipeline model.

3.1 Introduction

Pipelining and parallelism allow the same performance to be achieved at lower clock frequencies. The timing slack can be used to reduce the power by using a lower supply voltage, a higher threshold voltage and reduced gate sizes. Parallel computation trades off area for increased throughput. Pipelining breaks up a datapath into multiple stages with registers between each stage

to store the intermediate results. The shorter critical path length from pipelining allows a higher clock frequency. Computation in each pipeline stage can proceed simultaneously if there is no data interdependency, and thus the throughput is higher.

Chandrakasan and Brodersen examined pipelining and parallelism on an 8-bit datapath composed of an adder and comparator in 2.0um technology [32]. If two such datapaths are run in parallel, the clock period can be doubled, and the timing slack can be used to reduce to decrease the supply from 5V to 2.9V. The circuit capacitance more than doubles to 2.15 \times due to wiring overheads for the parallel datapath and multiplexing of the results. Running the datapath in parallel and reducing the supply reduces the dynamic power by 64%. If instead the datapath is pipelined with registers between the adder and comparator, the supply can be reduced from 5V to 2.9V and the capacitance overhead for the latches is 15%, giving a net power reduction of 61%. The area is more than doubled to 3.4 \times for the two parallel datapaths, whereas the area for the pipelined datapath is only 1.3 \times with the additional registers.

For an inverse discrete cosine (IDCT) core in 0.7um process technology with 0.5um channel length, the pipelining power overhead was about 20% of the total power. Without pipelining, the critical path would have been 4 \times as long and Vdd would have to be increased from 1.32V to 2.2V to achieve the same performance, increasing the total power by 2.2 \times [240].

Not all microarchitectural techniques for higher performance enable increased energy efficiency. Multiple instructions are executed in parallel execution units in a superscalar architecture, but the additional hardware to determine which instructions can be executed in parallel and reorder the instructions can reduce the energy efficiency [96]. Speculative execution before the outcome of a branch instruction is known wastes energy if the branch is mispredicted. Implementing speculative execution requires branch prediction logic and may require logic to rewind incorrect results. Software hints for branch prediction can reduce the hardware overhead [96].

Very deep pipelines are less energy efficient, as the pipelining overheads are too large and there is an increased penalty for pipeline hazards. Consequently, Intel is moving from the Pentium 4 NetBurst architecture with 31 stages to the Intel Core architecture with two processor cores that run in parallel, each having a 14 stage pipeline [97]. The Cedar Mill Pentium 4 has about 4.4× the energy/operation of the Yonah Core Duo, despite Yonah having only 2% lower performance on the SPEC CINT2000 benchmark and both being 65nm designs [79].

3.1.1 Power and performance metrics

A typical metric for performance is millions (MIPS) or billions of instructions per second (BIPS) on a benchmark. Commonly used performance benchmarks are the Dhrystone integer benchmark [234], and the integer (SPECint) and floating point (SPECfp) benchmarks from the Standard Performance Evaluation Corporation [189]. Power is measured in watts (W). To account for both power and performance, metrics such as BIPS^3/W , BIPS^2/W , and BIPS/W are used [24][186]. The inverse of these metrics are also often used. For example, energy per instruction (EPI) corresponds to W/BIPS , and energy-delay product¹ corresponds to W/BIPS^2 if we assume that the CPI is fixed.

Minimizing energy or power consumption leads to very large clock periods and low performance being optimal, as dynamic and leakage power can be greatly reduced by using the timing slack to reduce gate sizes, to reduce the supply voltage, and to increase the transistor threshold voltages (see Figure 3.3). Thus metrics placing more emphasis on performance are often used, for example BIPS^3/W and BIPS^2/W . More pipeline stages are optimal for metrics with higher weights on

¹ The energy-delay product was historically a voltage independent metric. In older process technologies without velocity saturation, the drain current of a transistor in the saturated region was proportional to the supply voltage squared. Thus to (dis)charge a capacitor with capacitance C holding charge CV_{dd} , the delay was proportional to $1/V_{dd}$. The dynamic power is proportional to CV_{dd}^2 , so the energy delay product (PT^2) was independent of V_{dd} . However, in today's technologies the current is limited by velocity saturation, leakage power must be also considered, and threshold voltage is usually not scaled down as fast as supply voltage. All of these factors mean that BIPS^2/W is no longer a voltage independent metric. See Chapter 5 for more detail on the factors affecting critical path delay and power consumption.

performance, as illustrated in Figure 3.4. Alternatively, the power consumption may be minimized for a specified performance or delay constraint. Changing the microarchitecture may change the delay constraint on the clock period to meet the given performance constraint, for example computing inverse discrete cosine transform serially or in parallel.

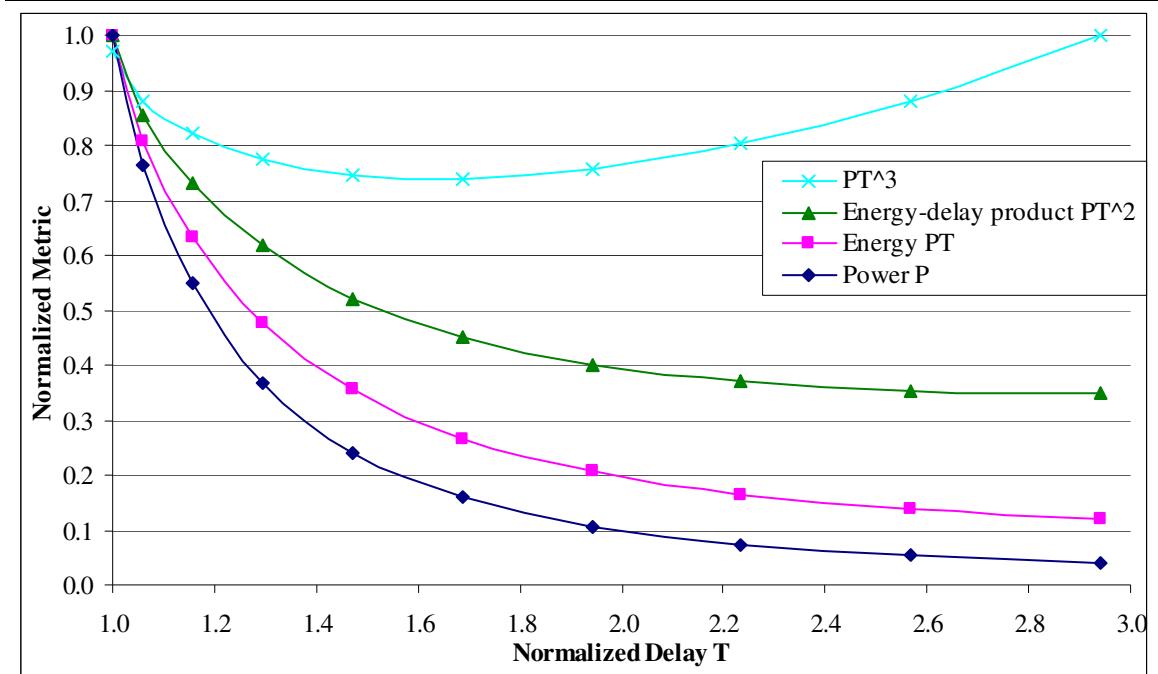


Figure 3.3 This graph shows the minimum power P for ISCAS'85 benchmark c880 versus the normalized delay constraint T . The corresponding energy (PT), energy-delay product (PT^2), and energy \times delay 2 (PT^3) are also shown. The power-performance metrics were normalized versus their maximum value. Delay was normalized to the minimum delay that could be achieved for the benchmark. These results are from geometric programming optimization of gate sizes, NMOS threshold voltage, PMOS threshold voltage and supply voltage (see Chapter 6 for details).

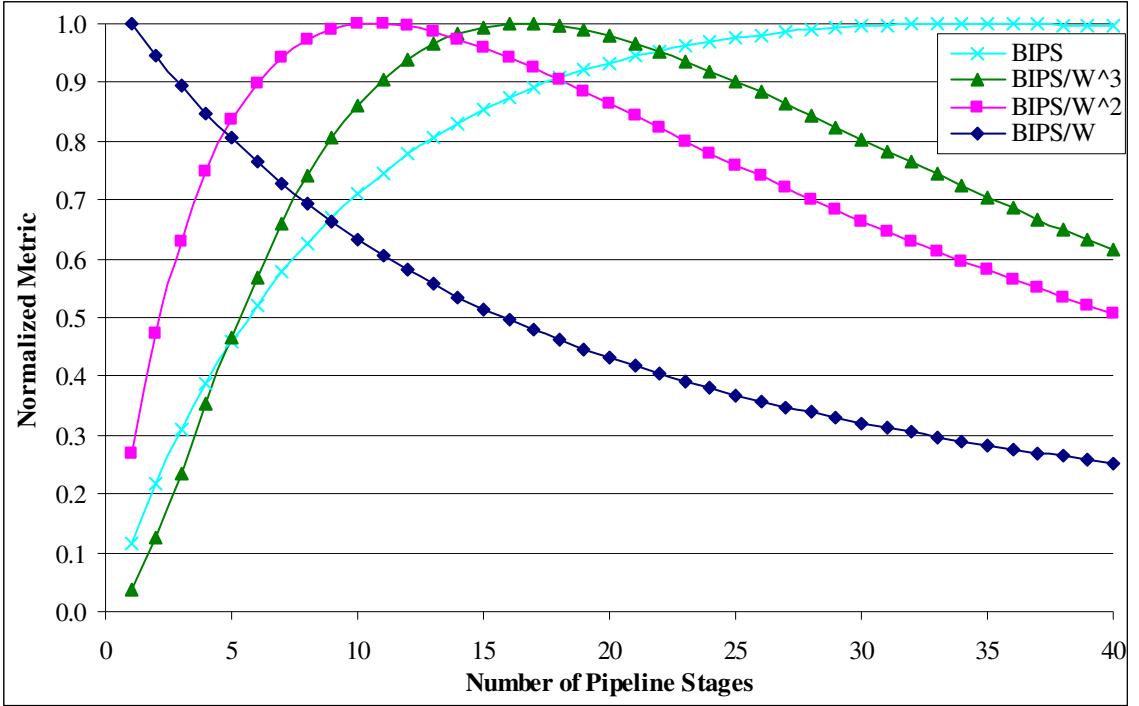


Figure 3.4 This graph shows the optimal value for various power-performance metrics versus the number of pipeline stages for the custom design parameters with our pipeline model. The metrics have been normalized versus their maximum value.

3.1.2 Parallel datapath model

Bhavnagarwala et al. developed a model for using parallel datapaths to scale down the supply voltage V_{dd} [17]. To meet the same performance with n datapaths, the clock frequency can be reduced by a factor of $1/n$, and the net switching activity and the dynamic power for the datapaths remain the same if the supply voltage is fixed. If voltages are fixed, the leakage power increases because there are n datapaths leaking rather than one. There is additional routing and multiplexing circuitry for the parallel datapaths, which adds to the dynamic and static power consumption. The expression for the total power that they derive is [17]

$$P_{total} = \frac{1}{2} \alpha C_{datapath} V_{dd}^2 F \left(1 + \frac{C_{overhead}}{C_{datapath}} \right) + P_{static\ for\ datapath} \left(n + \frac{C_{overhead}}{C_{datapath}} \right) \quad (3.1)$$

where $C_{datapath}$ is the total datapath capacitance that switches with activity α , F is the number of operations per second, and $C_{overhead}$ is the capacitance of the additional routing and multiplexing

circuitry. They estimate the overhead capacitance to depend quadratically on the number of parallel datapaths [17]:

$$C_{overhead} = (mn^2 + \Gamma)C_{datapath} \quad (3.2)$$

where m and Γ are fitting constants. From this they estimate that the power savings with parallel datapaths range from 80% power savings with four parallel datapaths for technology with channel length of 0.25um to 15% power savings with two parallel datapaths for technology with channel length of 0.05um. The optimal number of parallel datapaths decreases with technology generation as supply and threshold voltage V_{th} are scaled down, and the ratio of V_{dd}/V_{th} decreases, increasing the performance penalty for lower V_{dd} [17].

The overhead for parallel datapaths is very application dependent, with m in Equation (3.2) having a value from 0.1 to 0.7 depending on the application [17]. Thus the usefulness of parallelism depends greatly on the application. Generally, ASIC and custom designs can make similar use of parallel datapaths, but ASICs have larger wiring overheads with automatic place and route. ASICs suffer higher delay overheads than custom for pipelining – this has a much greater impact on the energy efficiency than ASIC overheads for parallel datapaths, so the remainder of this chapter focuses on the power gap between ASIC and custom with pipelining.

3.1.3 Pipeline model

Pipeline delay models suggest that deeply pipelined designs with logic depth of as low as 8 FO4 delays per stage are optimal for performance [98]. For integer and floating point SPEC 2000 benchmarks, Srinivasan et al. found that the optimal pipeline stage delay was 10 FO4 delays to maximize BIPS, 18 FO4 delays for the BIPS³/W metric, and 23 FO4 delays for BIPS²/W [186]. They assumed an unpipelined combinational logic delay of 110 FO4 delays and 3 FO4 timing delay overhead.

Harstein and Puzak did similar analysis following the work of Srinivasan et al. They assumed an unpipelined combinational logic delay of 140 FO4 delays and 2.5 FO4 timing delay overhead [89]. The optimal pipeline stage delay was 22.5 FO4 delays for the BIPS³/W metric, which is close to the result from Srinivasan et al. given the difference in unpipelined delays. In their models, the pipeline stage delay T is given by [89]

$$T = \frac{t_{\text{comb total}}}{n} + t_{\text{timing overhead}} \quad (3.3)$$

where $t_{\text{comb total}}$ is the unpipelined delay, n is the number of pipeline stages, and $t_{\text{timing overhead}}$ is the timing overhead for the registers and clocking. Their performance metric, average time per instruction, can be written for a scalar architecture as [89]

$$T/\text{instruction} = T(1 + \gamma n) \quad (3.4)$$

where γ is the increase in cycles per instruction (CPI) per pipeline stage due to pipeline hazards, and it is assumed that on average an instruction would complete execution every cycle in the absence of hazards. To determine the power for the registers, they use the expression from Srinivasan et al. [89][186],

$$P_{\text{timing}} = \left(\frac{1}{T} \alpha_{\text{clock gating}} E_{\text{dynamic}} + P_{\text{leakage}} \right) N_L n^\eta \quad (3.5)$$

where $\alpha_{\text{clock gating}}$ is the fraction of time the pipeline is not clock gated; E_{dynamic} and P_{leakage} are respectively the dynamic switching energy and the leakage power for a latch; N_L is the number of latches if there is only a single pipeline stage; n is the number of pipeline stages; and η is the latch growth factor with the number of pipeline stages.

We augment Harstein and Puzak's model by allowing timing slack to be used for voltage scaling and gate sizing to reduce the dynamic power and leakage power for the combinational logic and

the registers. In addition, we assume different ASIC and custom values for $t_{timing\ overhead}$ and include pipeline imbalance in the pipeline stage delay for ASICs.

Harstein and Puzak assume that $\alpha_{clock\ gating}$ is $1/(1+\gamma n)$ [89], with dynamic power consumption for pipeline hazards avoided by shutting off the clock to stalled pipeline stages. This is a reasonable assumption if there is no speculative execution. We will make the same assumption for the value of $\alpha_{clock\ gating}$. We do not consider power gating or reverse body biasing to reduce the leakage during a pipeline stall. For these leakage reduction techniques, the delay and power overhead to raise and lower the voltage are only justified when the circuitry will be unused for at least tens of clock cycles [44].

With an unpipelined combinational logic delay of 180 FO4 delays and 3 FO4 timing delay overhead for custom, we find that a clock period of 8 FO4 delays is optimal to maximize performance, 21 FO4 delays to maximize BIPS³/W, and 59 FO4 delays to maximize BIPS²/W. When power is included in the metric, the optimal clock period is significantly larger than that determined by Srinivasan et al., because we allow timing slack to be used to reduce power by voltage scaling and gate sizing. The optimal clock period for a typical ASIC is 2× to 4× larger than custom due to the 20 FO4 delay pipeline stage overhead.

The largest power gap between ASIC and custom is when it is difficult for the ASIC to meet the performance constraint. At the maximum performance for a typical ASIC of 56 FO4 delays on average per instruction, the ASIC power is 5.1× that of custom. As the performance constraint is relaxed, the power gap decreases to 4.0× at only 7% lower performance. For very low performance requirements, the energy efficiency of ASIC and custom microarchitectures is essentially the same.

The delay overhead is the most important factor for the power gap between ASIC and custom with pipelining. If the pipeline stage delay overhead can be reduced from 20 FO4 delays to 5 FO4 delays, the power gap between ASIC and custom is only up to 1.9×.

In Section 3.2, we discuss the overheads for pipelining in ASIC and custom designs. Using our geometric programming optimization results for dynamic power and leakage power with voltage scaling and gate sizing in Chapter 6, we augment Harstein and Puzak’s model with power reduction versus timing slack in Section 3.3. Then in Section 3.4, this augmented model of pipeline power and delay is used to estimate the power gap between ASIC and custom due to pipelining focusing on the impact of the required performance and the pipeline stage delay overhead. The effect of other factors in the pipeline model on the power gap is considered in Section 3.5. Glitching and additional power overheads affect the minimum energy per operation as discussed in Section 3.6. The results are summarized in Section 3.7.

3.2 Pipelining overheads

There are several pipelining overheads that we need to consider when comparing pipelining for ASIC and custom designs. There is timing overhead for the registers that store the combinational logic outputs of each stage, and the power consumption for the registers and clock signal. The delay of combinational logic in different pipeline stages may be imbalanced. The penalty for pipeline hazards that delay the next instruction being executed increases with the number of pipeline stages. These overheads are typically less for carefully designed custom circuits compared to ASICs.

Pipeline hazards include data dependency, branch misprediction, cache misses, and so forth. For example, the Willamette Pentium 4 with 20 pipeline stages has 10% to 20% less instructions per cycle than the Pentium III which has only 10 pipeline stages [107].

Adding the timing overhead and pipeline imbalance, the *pipelining delay overhead* is typically about 30% of the clock period for ASICs and 20% of the clock period for custom designs [38]; however, custom designs usually have a much smaller clock period than ASICs. The pipelining delay overhead may be 30% for a custom design with many pipeline stages such as the Pentium 4. When we compared the microarchitectural impact on ASIC and custom speeds, we estimated the pipelining delay overhead in FO4 delays¹ for a variety of custom and ASIC processors as shown in Table 3.1 and Table 3.2 [38]. The pipelining delay overhead ranges from as low as about 2 FO4 delays in some custom designs to 20 FO4 delays in the ASIC processors. See Appendix A.1 for more detailed tables with more processors.

The following subsections estimate timing overhead; pipeline imbalance; instructions per cycle with number of pipeline stages for ASIC and custom designs; and discuss the power overhead for pipelining. We will look at a pipeline model incorporating these factors in Section 3.3.

¹ The total delay for the logic without pipelining was calculated from the number of pipeline stages, estimated pipelining overhead, and the FO4 delay which was determined from the effective gate length as detailed in Section 2.2. The number of pipeline stages listed are for the integer pipeline; the Athlon floating point pipeline is 15 stages [2], and the Pentium III and Pentium 4 also have about 50% longer floating point pipelines. The FO4 delays for custom processes were calculated using Equation (2.2). For example, Intel's 0.18um process has an effective gate length of 0.10um, and the FO4 delay is about 50ps [95]. The estimated FO4 delay for these custom processes may be more than the real FO4 delay in fabricated silicon for these chips, because of speed-binning, unreported process improvements, and better than worse case operating conditions. As a result, the custom FO4 delays/stage may be underestimated.

Table 3.1 Characteristics of ASICs and super-pipelined Pentium 4 processors assuming 30% pipelining delay overhead [7][8][9][10][71][137][149][170][192][209][212][215][216][217][241]. The FO4 delay for typical (“typ”) process conditions was calculated with Equation (2.2), and Equation (2.3) was used for worst case (“wc”) process conditions.

Custom PCs	Frequency (MHz)	Technology (nm)	Effective Channel Length (nm)	Voltage (V)	Integer Pipeline Stages	FO4 delays/stage	20% Pipelining Overhead (FO4 delays)	Unpipelined Clock Period (FO4 delays)	Pipelining Overhead % of Unpipelined Clock Period	Clock Frequency Increase by Pipelining	Process Conditions
Pentium 4 (Willamette)	2000	180	100	1.75	20	10.0	3.0	143	2.1%	$\times 14.3$	typ
Pentium 4 (Gallatin)	3466	130	60	1.60	31	9.6	2.9	212	1.4%	$\times 22.0$	typ
ASICs											
Xtensa T1020 (Base)	250	180	130	1.80	5	61.5	18.5	234	7.9%	$\times 3.8$	typ
Lexra LX4380	266	180	130	1.80	7	57.8	17.4	301	5.8%	$\times 5.2$	typ
iCORE	520	180	150	1.80	8	25.6	7.7	151	5.1%	$\times 5.9$	typ
ARM 926EJ-S	200	180	130	1.80	5	64.1	19.2	244	7.9%	$\times 5.9$	wc
ARM 1026EJ-S	540	90	50	1.00	6	61.7	18.5	278	6.7%	$\times 4.5$	wc
ARM 1136J-S	400	130	80	1.20	8	52.1	15.6	307	5.1%	$\times 5.9$	wc
ARM Cortex-A8	800	65	40	1.20	13	62.5	18.8	588	3.2%	$\times 9.4$	wc

Table 3.2 Custom design characteristics [55][71][73][78][80][84][85][94][105][115][148][149][163][180][206][220], assuming 20% timing overhead. The FO4 delay for typical (“typ”) process conditions was calculated with Equation (2.2).

Custom Processors	Frequency (MHz)	Technology (nm)	Effective Channel Length (nm)	Voltage (V)	Integer Pipeline Stages	FO4 delays/stage	20% Pipelining Overhead (FO4 delays)	Unpipelined Clock Period (FO4 delays)	Pipelining Overhead % of Unpipelined Clock Period	Clock Frequency Increase by Pipelining	Process Conditions
Alpha 21264	600	350	250	2.20	7	13.3	2.7	77	3.4%	$\times 5.8$	typ
IBM Power PC	1000	250	150	1.80	4	13.3	2.7	45	5.9%	$\times 3.4$	typ
Custom PCs											
Athlon XP (Palomino)	1733	180	100	1.75	10	11.5	2.3	95	2.4%	$\times 8.2$	typ
Athlon 64 (Clawhammer)	2600	130	80	1.50	12	9.6	1.9	94	2.0%	$\times 9.8$	typ
Pentium III (Coppermine)	1130	180	100	1.75	10	17.7	3.5	145	2.4%	$\times 8.2$	typ
Core 2 Extreme (Conroe)	2930	65	35	1.34	14	19.5	3.9	222	1.8%	$\times 11.4$	typ
Custom ARMs											
StrongARM	215	350	250	2.00	5	37.2	7.4	156	4.8%	$\times 4.2$	typ
XScale	800	180	135	1.80	7	18.5	3.7	107	3.4%	$\times 5.8$	typ
Halla (ARM 1020E)	1200	130	80	1.10	6	20.8	4.2	104	4.0%	$\times 5.0$	typ

Table 3.3 Comparison of ASIC and custom timing overheads, assuming balanced pipeline stages [38]. Alpha 21164 [16], Alpha 21264 [80] and Pentium 4 [127] setup times were estimated from known setup times for latches and pulse-triggered flip-flops. The pulse-triggered latches in the Pentium 4 are effectively used as flip-flops rather than as transparent latches. Timing overhead for flip-flops was calculated from $t_{CQ} + t_{su} + t_{sk} + t_j$. As there are two latches, positive and negative-edge triggered, per clock cycle, the timing overhead for latches was calculated from $2t_{DQ} + t_j \text{ multicycle}$, where multi-cycle jitter $t_j \text{ multicycle}$ of 1.0 FO4 delays was assumed for ASICs.

Contributions to ASIC and custom timing overhead in FO4 delays	ASICs		Custom		
	D-type flip-flops	Good Latches	Pass transistor latches in Alpha 21164	Edge-triggered flip-flops in Alpha 21264	Clock-pulsed latches in Pentium 4
Clock-to-Q delay t_{CQ}	4.0			2.0	2.0
2× D-to-Q latch propagation delay $2 \times t_{DQ}$		4.0	2.6		
Flip-flop setup time t_{su}	2.0			0.0	0.0
Edge jitter t_j				0.1	0.7
Clock skew t_{sk}				0.7	0.3
Budget for clock skew and edge jitter $t_{sk} + t_j$	4.0	1.0			
Timing overhead per clock cycle	10.0	5.0	2.6	2.8	3.0

3.2.1 Timing overhead per pipeline stage for ASIC and custom designs

The timing overhead specifies the delay for the registers and synchronization of the clock signal to the registers. It includes the setup time during which the input to the register must be stable before the clock signal arrives; the delay for a signal to propagate from a register's input to output; clock skew accounting for the clock signal arriving at different registers at different times; and clock jitter in the arrival time of the periodic clock signal.

The timing overhead for an ASIC may be as much as 10 FO4 delays, but can be reduced to 5 FO4 delays if latches are used instead of D-type flip-flops. We have used Design Compiler scripts to automate replacement of flip-flops by latches, achieving 5% to 20% speed increase in the Xtensa processor [37]. In comparison, the custom timing overhead can be as low as 2.6 FO4 delays as detailed in Table 3.3.

Table 3.4 Summary of pipeline imbalance for ASIC and custom designs

	Pipeline Imbalance (FO4 delays)
Typical ASIC with flip-flops	10.0
Carefully balanced ASIC with flip-flops	2.6
Optimal design with flip-flops, no slack passing	1.0
Slack passing via latches or cycle stealing	0.0

3.2.2 Pipeline imbalance in ASIC and custom designs

The pipeline imbalance for an ASIC with flip-flops can range from 10 FO4 delays down to 2.6 FO4 delays in a carefully balanced design. Unbalanced critical path delays in different pipeline stages can be addressed in several ways. ASICs may use automatic retiming of the register positions to balance critical path delays in different stages. Slack passing by using transparent latches or by useful clock skew is commonly used in custom designs. Useful clock skew tailors the arrival time of the clock signal to different registers by adjusting buffers in the clock tree, and can be used in ASIC designs [52]. Pipeline imbalance with different design techniques is summarized in Table 3.4.

From our experiments replacing flip-flops by latches in the 5-stage Tensilica Xtensa processor [37], we estimate that a typical ASIC may have imbalance of 15% of the clock period. The base configuration of the Xtensa has a maximum stage delay (clock period) of about 67 FO4 delays, of which 15% is 10 FO4 delays.

The imbalance between pipeline stages for a well balanced ASIC can be as low as 10% of the clock period. For example, the 8-stage STMicroelectronics iCORE has about 10% imbalance in the critical sequential loop through IF1, IF2, ID1, ID2, and OF1 back to IF1 through the branch target repair loop [170]. The iCORE has about 26 FO4 delays per pipeline stage, thus the imbalance is about 2.6 FO4 delays.

Automated flip-flop retiming won't typically achieve a pipeline imbalance of a single gate delay. Retiming is based on assumptions such as fixed register delay, fixed register setup time, and fixed gate delays. In reality, this depends on the drive strength and type of flip-flop chosen, and the gate

loads which are changed by retiming. Additionally, the combinational gates and registers will usually be resized after retiming. Reducing the clock period by retiming may be limited by input or output delay constraints, such as reading from and writing to the cache [91]. These issues limit the optimality of automated retiming.

Slack passing is not limited by the delay of a particular stage. Custom designers also have tighter control of gate delays, register positions, and better knowledge of wire loads that depend on layout – which is not known for retiming in the synthesis stage of an ASIC EDA methodology. Thus custom designs may be able to balance stages, whereas ASICs typically suffer some pipeline imbalance.

With useful clock skew or transparent latches, slack passing between stages can eliminate pipeline imbalance. From the iCORE and Xtensa examples, a 10% to 15% reduction in clock period can be achieved by slack passing for ASICs with imbalanced pipeline stages.

Table 3.5 Cycles per instruction (CPI) for different processors [48][53][93][129] [170][176][244].

Processor	# of Pipeline Stages	IPC	CPI	Increase in CPI/stage, γ
ARM7TDMI	3	0.53	1.90	30.0%
ARM9TDMI	5	0.67	1.50	10.0%
ARM810	5	0.71	1.40	8.0%
DEC StrongARM	5	0.61	1.63	12.7%
Intel XScale	7	0.56	1.78	11.2%
STMicroelectronics iCORE	8	0.70	1.43	5.4%
Pentium 4 (Willamette)	20	not known		3.0%
Pentium 4 (Cedar Mill)	31	0.54	1.84	2.7%

3.2.3 Instructions per cycle versus number of pipeline stages

Instructions per cycle (IPC) and its reciprocal *cycles per instruction* (CPI) are measures of how quickly instructions are executed after accounting for pipeline stalls due to hazards. Reductions in IPC can be caused by cache misses, waiting for data from another instruction that is executing, branch misprediction, and so forth. The CPI for a number of processors is summarized in Table 3.5.

The CPI is very application dependent, as some applications have more branches and other hazards. For the Cedar Mill Pentium 4 with 31 pipeline stages, the CPI ranges from 0.64 to 7.87 for different benchmarks in the SPEC CINT2000 benchmark set. The geometric mean for Cedar Mill for the SPEC CINT2000 benchmark set was 1.84 [53]. The 1.5GHz Willamette Pentium 4 with 20 pipeline stages is 15% to 20% faster for integer applications than a 1.0GHz Pentium III with 10 pipeline stages [93], which corresponds to a 20% to 23% worse IPC.

For a variety of benchmarks, the IPC for the five stage DEC StrongARM ranges from 0.30 to 0.83 [244], and the IPC ranges from 0.38 to 0.82 for the seven stage Intel XScale [48]. The geometric means of the IPC values were 0.61 and 0.56 respectively.

The IPC for the three stage ARM7TDMI was 0.5, whereas the ARM9TDMI with five pipeline stages had an IPC of 0.7 [176]. The higher IPC for the five stage ARM810 and ARM9 pipelines was achieved by adding static branch prediction, single cycle load, single cycle store, and doubling the memory bandwidth [129]. The eight stage STMicroelectronics iCORE also achieved an IPC of 0.7 by microarchitectural optimizations in an ASIC EDA methodology [170]. The ARM810 used standard cells for the control logic, but was otherwise full custom [129]. The ARM7TDMI and ARM9TDMI were full custom, but synthesizable versions of these processors were also created [65].

Without additional microarchitectural features such as data forwarding and improved branch prediction to maintain high IPC, the CPI increases approximately linearly with the number of pipeline stages as more pipeline stages are stalled when a hazard is encountered [89]. Assuming that an unpipelined design has an IPC of close to 1.0, which is somewhat optimistic as there may be cache misses and off-chip memory will take more than a cycle to read, the CPI increase per pipeline stage ranges from 30.0% to 2.7%.

Table 3.6 Register and clock tree power consumption in various processors. The StrongARM [144] and XScale [45] are custom embedded processors. The Alpha 21264 [76] and Itanium [6] are custom desk top processors. The MCORE [75] is a semi-custom processor, and the 16-bit CompactRISC ASIC [139] was synthesized.

Processor	# of Pipeline Stages	Registers and Clock Tree
		Power as % of Total Power
16-bit CompactRISC	3	34%
MCORE	4	36%
StrongARM	5	25%
XScale (DSP FIR filter)	7	18%
XScale (Dhrystone MIPS)	7	23%
Alpha 21264	7	32%
Itanium	8	33%

3.2.4 Power overheads for pipelining

The majority of the power overhead for pipelining is power consumption in the clock tree and registers. The registers and clock tree can consume from 18% to 36% of the total power, as shown in Table 3.6. Clock gating was used in these processors to reduce the power consumed by the registers and clock tree.

Branch prediction, data forwarding and other microarchitectural techniques to maintain a high IPC with deeper pipelines also take some power. We assume that these additional power overheads are small relative to the clock tree and register power, and do not explicitly include them in the pipeline power model in the same manner as [89] and [186].

The percentage of power consumed by registers and the clock tree depends on the application. For example, the clock tree accounts for 18% of the XScale for the DSP FIR benchmark, but it is 23% for the Dhrystone MIPS 2.1 benchmark [45].

In Motorola's 0.36um 1.8V MCORE embedded processor with a four stage pipeline, the datapath and clock tree each contribute 36% of the total power, and control logic contributes the other 28% of the total power. If the custom datapath was instead synthesized, the datapath's power would have been 40% higher [75].

Few breakdowns of power data for synthesized processors are available. We expect register and clock tree power to consume a similar portion of the total power in ASICs. In a synthesized 0.18um 1.8V National Semiconductor 16-bit CompactRISC processor with a three stage pipeline, the register file consumed 34% of the processor core's total dynamic power [139].

We will now examine a pipelining power model that incorporates these pipelining overheads.

3.3 Pipelining power and delay model

To build the pipeline power and delay model, we first calculate the minimum pipeline stage delay T_{\min} . Given some upper limit on the clock period, the actual clock period T can be anywhere between the upper limit and the minimum. We will discuss a simple experimental fit to determine the reduced power from voltage scaling and gate downsizing with timing slack ($T - T_{\min}$). We can then find the optimal number of pipeline stages and optimal amount of timing slack to use for power reduction in order to minimize the power.

3.3.1 Pipeline stage delay

The number of pipeline stages n in a processor varies widely. For example the ARM7 architecture has a three stage pipeline comprising instruction fetch, instruction decode, and execute [68]; whereas the Cedar Mill Pentium 4 has 31 stages [97]. The total unpipelined delay $t_{comb\ total}$ can range from about 50 to 300 FO4 delays, as was estimated earlier in Table 3.1 and Table 3.2.

If a pipeline with n stages is ideally balanced, the combinational delay per pipeline stage is

$$t_{comb} = \frac{t_{comb\ total}}{n} \quad (3.6)$$

The maximum delay of a pipeline stage, which limits the minimum clock period T_{\min} , is

$$T_{\min} = \frac{t_{comb\ total}}{n} + t_{imbalance} + t_{timing\ overhead} \quad (3.7)$$

where $t_{imbalance}$ accounts for the pipeline stages being unbalanced, and $t_{timing\ overhead}$ is the timing overhead.

The clock period T that is used must be at least T_{min} , but may be larger to provide slack for power reduction by voltage scaling and gate downsizing.

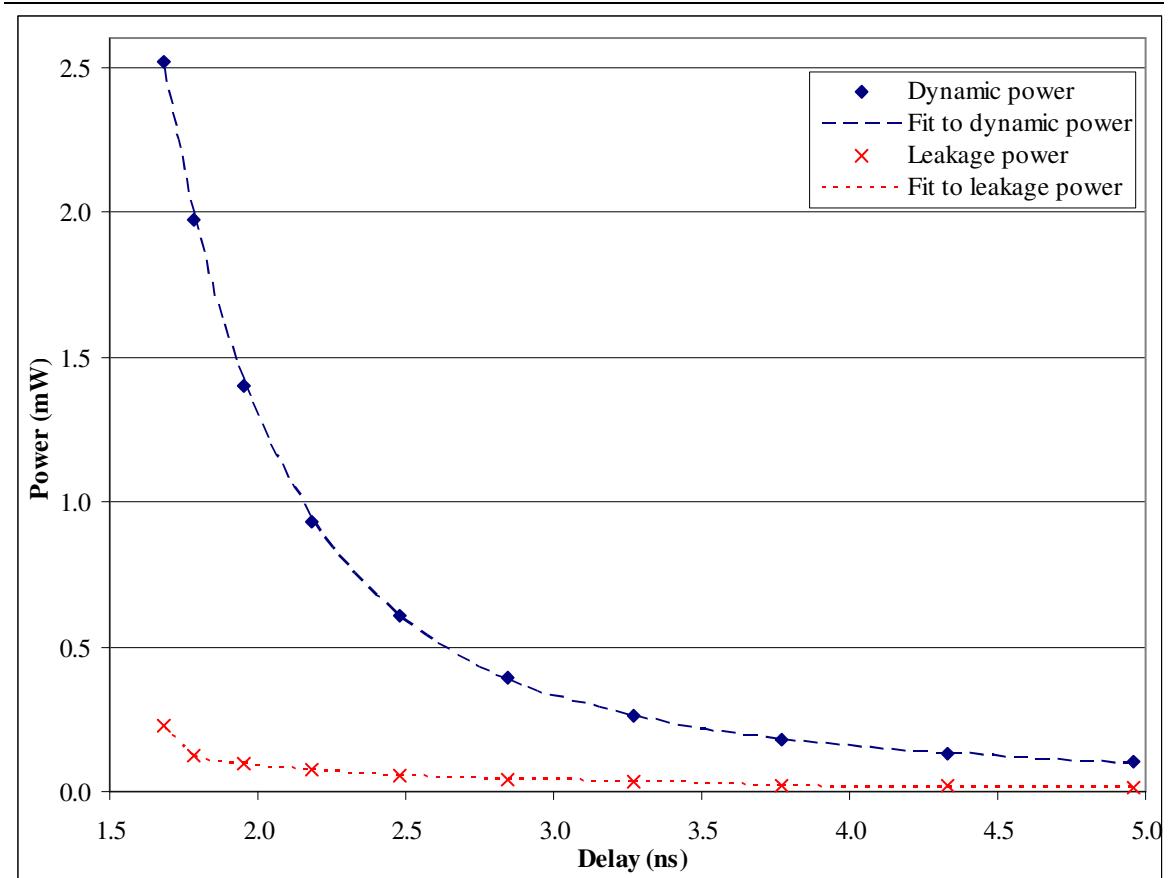


Figure 3.5 Curve fits for the dynamic power and the leakage power for ISCAS'85 benchmark c880. The total power was minimized by choosing optimal gate sizes, single supply voltage, single NMOS threshold voltage and single PMOS threshold voltage by geometric program optimization as detailed in Chapter 6.

3.3.2 Utilizing slack for voltage scaling and downsizing to reduce power

We can reduce the dynamic energy per clock cycle and reduce the leakage power by reducing the supply voltage V_{dd} , increasing the threshold voltage V_{th} , and reducing the gate size. The impact of gate size, supply voltage and threshold voltage on the leakage power and dynamic power is detailed in Chapter 5. In this section, we are just interested in how the leakage power and dynamic power decrease as timing slack is used to downsize gates and scale the voltages.

Increasing a gate's size reduces the delay, but increases the load on fanins, requiring them to also be upsized. Consequently, at a tight delay constraint there is substantially higher power consumption with many gates having been upsized. To meet a tight delay constraint, the supply voltage will also be higher and the threshold voltage may be lower to reduce the critical path delay. The rapidly increasing power consumption as T approaches the minimum delay T_{\min} results in the classic “banana” curve shape shown for dynamic power and leakage power in Figure 3.5. From a tight delay constraint, a small amount of timing slack can be used to significantly reduce the energy consumed per clock cycle.

The power versus delay curves with gate sizing and voltage scaling for the dynamic power and leakage power versus clock period are fit well by hyperbolic functions of the form

$$a + \frac{b}{\left(\frac{T}{T_{\min}} + c\right)^d} \quad (3.8)$$

where a , b , c and d are experimentally fitted constants¹. We require that $a \geq 0$, so that the power does not become negative as T becomes large. The values for these fitting parameters depend on a number of things including: the circuit topology and logic gates that compose the circuit; the allowed ranges of gate sizes, supply voltage, and threshold voltages; and the process technology.

As the leakage depends exponentially on the threshold voltage, and the threshold voltage can be increased with increasing clock period, the accuracy of the fit to leakage power can be improved by including an exponential term:

¹ Note that fitting these constants is a nonlinear non-convex optimization problem. A good fit may be found with some user guidance. For example, we used Microsoft Excel's Solver add-in to optimize the fitting constants using the Newton-Raphson method with the constraint that $c > -1$ to avoid divide by zero at $T = T_{\min}$ and starting values of $a = 0$, $c = -0.99$, $d = -1.00$ with b chosen so as to fit the value at $T = T_{\min}$.

$$a + \frac{be^{\lambda T/T_{\min}}}{\left(\frac{T}{T_{\min}} + c\right)^d} \quad (3.9)$$

where exponent λ is also an experimentally fitted constant.

Accurate fits for the dynamic power and the leakage power for ISCAS'85 benchmark c880 are shown in Figure 3.5. This 0.13um data for c880 was determined by geometric program optimization of the gate sizes, the supply voltage, the NMOS threshold voltage and the PMOS threshold voltage as detailed in Chapter 6. The allowed range for the supply voltage was 1.3V to 0.6V. The allowed ranges for the threshold voltages were $\pm 0.13V$ from the nominal threshold voltage. The relative root mean square error is 0.5% for the dynamic power fit and 3.1% for the leakage power fit, where the relative root mean square (RMS) error is given by

$$\text{Relative Root Mean Square Error} = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{y_{\text{data},i} - y_{\text{fit},i}}{y_{\text{data},i}} \right)^2} \quad (3.10)$$

The leakage power contributes from 5.9% to 11.8% of the total power. The maximum error in the total power is 0.7%. The fits are

$$P_{\text{dynamic}}(T) = 0.0422 + \frac{0.758}{\left(\frac{T}{T_{\min}} - 0.352\right)^{2.704}} \quad (3.11)$$

$$P_{\text{leakage}}(T) = \frac{0.212e^{-0.971T/T_{\min}}}{\left(\frac{T}{T_{\min}} - 0.997\right)^{0.182}} \quad (3.12)$$

where fitting coefficients have been shown to three significant figures, and T_{\min} for c880 was 1.69ns. Without the exponential term, the best fit for the leakage has relative RMS error of 12.1%.

With pipelining allowing higher clock frequency, the switching activity and hence dynamic power increases proportionally to the clock frequency, that is as $1/T$. The dynamic power fit implicitly includes the dependence of switching activity on T .

Glitching caused by spurious transitions from signals propagating through the logic at different speeds also affects the switching activity. Glitching depends approximately linearly on the logic depth [186], so pipelining reduces glitching by reducing the logic depth. Glitching only has a small impact on the power gap between ASIC and custom, so we do not consider it at this stage – see Section 3.6.1 for a discussion of the impact of glitching.

We must also account for the power consumption of the registers and clock tree, considering that the number of registers per pipeline stage may vary with pipeline depth.

3.3.3 Power consumption of the registers and clock tree

Deeper pipelines typically require more registers per pipeline stage, because balancing the stage delays may require the additional registers to be placed at cut points where there are more edges. To take into account the register and the clock tree power, Harstein and Puzak [89] use a power model from Srinivasan et al. [186], which has the form

$$P_{\text{timing}} = \left(\frac{1}{T} \alpha_{\text{clock gating}} E_{\text{dynamic}} + P_{\text{leakage}} \right) \beta n^{\eta} \quad (3.13)$$

where $\alpha_{\text{clock gating}}$ is the fraction of time the pipeline is not clock gated; βn^{η} is the additional fraction of power due to the registers and the clock tree; and η is a scaling factor – the “latch growth factor”. η takes into account how many additional registers are required as the pipeline depth increases. If $\eta = 1$, the number of registers for per pipeline stage does not vary with the depth.

Srinivasan et al. show that the latch growth factor η has a value of about 1.7 for a floating point unit, and 1.9 for a Booth recoder and Wallace tree multiplier. Throughout most of their analysis they assume a value for η of 1.1 [186].

Harstein and Puzak's power model is based on the work by Srinivasan et al., and also assumes a value for η of 1.1 [89]. For a fixed circuit with neither gate sizing nor voltage scaling, as the dynamic power consumption is proportional to the switching activity and hence the clock frequency, the dynamic energy per clock cycle is independent of the clock frequency. Thus, they assume that the register and clock tree power is the only significant change in the total power with the number of pipeline stages n , as the dynamic energy for the combinational logic is fixed.

Harstein and Puzak's omission of combinational power in Equation (3.13) is an error in their model. If combinational power is included, the theoretical power model closely fits their simulation results, as detailed in Appendix A.3. Note that the pipeline power model used by Srinivasan et al. [186] did include the power consumption of combinational logic and latches with their "LatchRatio" factor.

Voltage scaling and gate sizing can be used to reduce the power consumption of the registers and clock tree, and we assume that their power scales in the same manner versus timing slack as the combinational logic. Including the combinational logic's power consumption in Equation (3.13) gives

$$P_{\text{timing}} = \left(\frac{1}{T} \alpha_{\text{clock gating}} E_{\text{dynamic}} + P_{\text{leakage}} \right) (1 + \beta n^\eta) \quad (3.14)$$

Allowing timing slack ($T - T_{\min}$) for gate sizing and voltage scaling to help reduce the power, using the dynamic and leakage power models from Section 3.3.2 normalized to their value at T_{\min} ,

$$P_{total}(T) = \frac{1}{T_{min}} \left(\alpha_{clock\ gating} \frac{P_{dynamic}(T)}{P_{dynamic}(T_{min})} (1 - k_{leakage}) + k_{leakage} \frac{P_{leakage}(T)}{P_{leakage}(T_{min})} \right) (1 + \beta n^\eta) \quad (3.15)$$

where $k_{leakage}$ is the fraction of total power due to leakage at T_{min} . Our dynamic and leakage power models account for the decrease in switching activity for $T > T_{min}$, but we still require the $1/T_{min}$ factor to account for switching activity varying with T_{min} . The $1/T_{min}$ factor also affects leakage, because the fraction of total power due to leakage at T_{min} is determined versus the dynamic power without clock gating ($\alpha_{clock\ gating} = 1$).

As in the earlier models, we will assume that a value for η of 1.1 is a reasonable estimate for the integer pipeline of a processor. Assuming a given value for η , we can calculate the value of β in Equation (3.15) from the number of pipeline stages and the percentage of register and clock tree power in a processor. From the register and clock tree power data for different processors in Table 3.6 and assuming η of 1.1, β ranges from 0.026 to 0.15. We use a value for β of 0.05, which is typical for most of the processors of five to eight pipeline stages. See Appendix A.2 for more details.

We use Harstein and Puzak's model for the clock gating factor [89],

$$\alpha_{clock\ gating}(n) = 1/(1 + \gamma n) \quad (3.16)$$

where clock gating enables avoiding dynamic power consumption due to pipeline hazards by shutting off the clock to stalled pipeline stages. This is a reasonable assumption if there is no speculative execution.

We now look at incorporating these models to choose the optimal number of pipeline stages and allocation of slack for voltage scaling and gate sizing.

3.3.4 The pipeline power and delay model for optimization

We now have the pipeline stage delay and can calculate the power reduction from the timing slack used for voltage scaling and gate sizing. The number of clock cycles per instruction must be accounted for. Assuming one instruction would be executed per cycle if there were no hazards, and assuming that penalty for pipeline hazards increases linearly with the number of stages as discussed in Section 3.2.3, the average time per instruction is [89]

$$T/\text{instruction} = T(1 + \gamma n) \quad (3.17)$$

where γ is the increase in CPI per pipeline stage due to hazards.

Typical metrics that we wish to optimize include maximizing the performance, minimizing the energy per operation, and minimizing the power consumption for a given performance constraint. The numerical solution of these optimization problems will usually give a non-integer value for the number of pipeline stages n , though in a real circuit n must be integral.

3.3.4.1 Maximum performance: minimum $T/\text{instruction}$

The maximum performance can be found by minimizing the average time per instruction,

$$\begin{aligned} & \text{minimize} && T(1 + \gamma n) \\ & \text{subject to} && T = \frac{t_{\text{comb total}}}{n} + t_{\text{imbalance}} + t_{\text{timing overhead}} \\ & && n \geq 1 \end{aligned} \quad (3.18)$$

By taking the derivative and setting it to zero to find the minimum, this has solution

$$\begin{aligned} n &= \sqrt{\frac{t_{\text{comb total}}}{\gamma(t_{\text{imbalance}} + t_{\text{timing overhead}})}} \\ T &= t_{\text{imbalance}} + t_{\text{timing overhead}} + \sqrt{\gamma t_{\text{comb total}} (t_{\text{imbalance}} + t_{\text{timing overhead}})} \\ \min T/\text{instruction} &= t_{\text{imbalance}} + t_{\text{timing overhead}} + \gamma t_{\text{comb total}} + 2\sqrt{\gamma t_{\text{comb total}} (t_{\text{imbalance}} + t_{\text{timing overhead}})} \end{aligned} \quad (3.19)$$

3.3.4.2 Maximum BIPS^m/W: minimum $P_{total}(T/\text{instruction})^m$

Minimizing the energy per operation is equivalent to maximizing BIPS/W (instructions per second per unit of power). The minimum energy per operation is found when there is substantial timing slack to reduce the dynamic and leakage power, and the pipelining power overheads are minimized. Consequently, a single pipeline stage is optimal to minimize the energy per operation, as this minimizes the power overhead for registers. More than one pipeline stage is optimal to minimize energy per operation when glitching and additional power overheads are accounted for – see Section 3.6 for further discussion.

The solution for minimum energy per operation is not particularly interesting from a circuit design viewpoint as most applications require higher performance than where the minimum energy per operation occurs. Thus the performance is usually more heavily weighted in the objective. The more general optimization problem is to maximize BIPS^m/W by finding the optimal number of pipeline stages n and optimal clock period T in

$$\begin{aligned}
 & \text{minimize} && P_{total}(T(1+\gamma n))^m \\
 & \text{subject to} && T_{\min} = \frac{t_{\text{comb total}}}{n} + t_{\text{imbalance}} + t_{\text{timing overhead}} \\
 & && T \geq T_{\min} \\
 & && n \geq 1 \\
 & && P_{\text{dynamic}}(T) = a_{\text{dynamic}} + \frac{b_{\text{dynamic}}}{\left(\frac{T}{T_{\min}} + c_{\text{dynamic}}\right)^{d_{\text{dynamic}}}} \\
 & && P_{\text{leakage}}(T) = a_{\text{leakage}} + \frac{b_{\text{leakage}} e^{\lambda_{\text{leakage}} T / T_{\min}}}{\left(\frac{T}{T_{\min}} + c_{\text{leakage}}\right)^{d_{\text{leakage}}}} \\
 & && P_{\text{total}} = \frac{1}{T_{\min}} \left(\frac{1}{(1+\gamma n)} \frac{P_{\text{dynamic}}(T)}{P_{\text{dynamic}}(T_{\min})} (1 - k_{\text{leakage}}) + k_{\text{leakage}} \frac{P_{\text{leakage}}(T)}{P_{\text{leakage}}(T_{\min})} \right) (1 + \beta n^\eta)
 \end{aligned} \tag{3.20}$$

where subscripts are included for the coefficients to fit the dynamic power and the leakage power models in Equation (3.8) and Equation (3.9) respectively. m is typically 0, 1, 2, or 3. m of 0

corresponds to minimizing power, regardless of instruction efficiency. $m=1$ corresponds to minimizing the energy per operation executed. The energy-delay product is given by $m=2$. Values for m of 2 or more emphasize minimizing delay over minimizing power.

The optimization problem in Equation (3.20) can be solved easily with the Newton-Raphson gradient descent algorithm. The optimization surface is illustrated in Figure 3.6 maximizing BIPS^2/W with the parameters for custom in Table 3.7. The maximum BIPS^2/W is 0.0521, corresponding to a minimum $P_{\text{total}}(T(1+\gamma n)^2$ of 19.2, which occurs at 10.7 pipeline stages and a clock period of 58.4 FO4 delays. The minimum clock period possible with 10.7 pipeline stages is 19.9 FO4 delays, so the ratio of T/T_{\min} of 2.94 provides a substantial amount of slack for voltage scaling and gate sizing. The average delay per instruction at this point is 89.6 FO4 delays. The scale is arbitrary as power has been normalized and we are only interested in the relative values of the metrics.

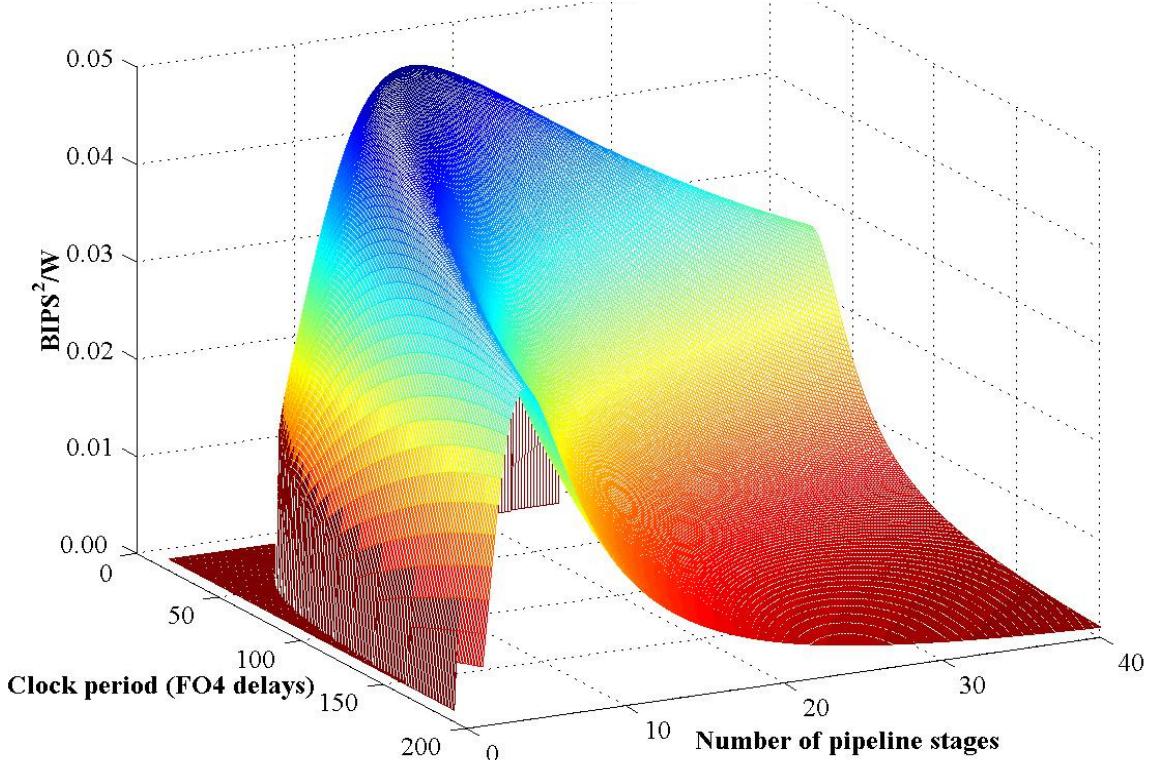


Figure 3.6 The surface for BIPS^2/W or $1/(P_{\text{total}}(T(1+\gamma n)^2)$ for the custom parameters in Table 3.7. The shape of the surface for ASIC parameters is similar.

3.3.4.3 Minimum power P_{total} for performance constraint $T_{\text{required per instruction}}$

Many applications require a specific performance. To find the minimum power for a given required performance, or average time per instruction $T_{\text{required per instruction}}$, we solve for n and T in

$$\begin{aligned}
 & \text{minimize} && P_{total} \\
 & \text{subject to} && T_{\min} = \frac{t_{\text{comb total}}}{n} + t_{\text{imbalance}} + t_{\text{timing overhead}} \\
 & && T \geq T_{\min} \\
 & && n \geq 1 \\
 & && T(1 + \gamma n) = T_{\text{required per instruction}} \\
 & && P_{\text{dynamic}}(T) = a_{\text{dynamic}} + \frac{b_{\text{dynamic}}}{\left(\frac{T}{T_{\min}} + c_{\text{dynamic}}\right)^{d_{\text{dynamic}}}} \\
 & && P_{\text{leakage}}(T) = a_{\text{leakage}} + \frac{b_{\text{leakage}} e^{\lambda_{\text{leakage}} T / T_{\min}}}{\left(\frac{T}{T_{\min}} + c_{\text{leakage}}\right)^{d_{\text{leakage}}}} \\
 & && P_{total} = \frac{1}{T_{\min}} \left(\frac{1}{(1 + \gamma n)} \frac{P_{\text{dynamic}}(T)}{P_{\text{dynamic}}(T_{\min})} (1 - k_{\text{leakage}}) + k_{\text{leakage}} \frac{P_{\text{leakage}}(T)}{P_{\text{leakage}}(T_{\min})} \right) (1 + \beta n^\eta)
 \end{aligned} \tag{3.21}$$

We will now use the solutions of Equation (3.21) to compare the minimum power for ASIC and custom designs for a given performance constraint.

Table 3.7 This table lists parameters and variables used for pipeline modeling. The default values that we assume for the parameters are listed. The excellent ASIC column corresponds to an ASIC using latches, or faster pulsed flip-flops where needed and useful clock skew, to reduce the register delay and impact of clock skew, and to address imbalance between pipeline stage delays.

Parameter for Design Style Overhead	Represents	Value for		
		Typical ASIC	Excellent ASIC	Custom
$t_{imbalance}$ (FO4 delays)	delay overhead for unbalanced pipeline stages	10	0	0
$t_{timing overhead}$ (FO4 delays)	timing overhead per pipeline stage	10	5	3

Parameter	Represents	Value
$k_{leakage}$	fraction of total power due to leakage at T_{min}	0.1
m	exponent for delay per instruction in the objective	varies
$T_{required per instruction}$ (FO4 delays)	required delay per instruction	varies
$t_{comb total}$ (FO4 delays)	total unpipelined combinational delay	180
β	coefficient for power due to registers and the clock tree	0.05
γ	increase in clock cycles per instruction (CPI) with pipeline stages due to hazards	0.05
η	latch growth factor for increase in number of registers with pipeline depth	1.1

Optimization Variable	Represents
n	number of pipeline stages
T (FO4 delays)	clock period

Dependent Variable	Represents
$P_{dynamic}$	dynamic power
$P_{leakage}$	leakage power
P_{timing}	power consumption of registers and clock
P_{total}	total power consumption
t_{comb} (FO4 delays)	combinational delay per pipeline stage
T_{min} (FO4 delays)	minimum clock period
$\alpha_{clock gating}$	fraction of time pipeline is not clock gated

3.4 ASIC versus custom pipelining

We will now estimate the gap between ASIC and custom designs due to microarchitecture using the model in Section 3.3 with default values for parameters listed in Table 3.7. We will assume parameter values that correspond to an integer pipeline in a high performance ASIC processor and the custom equivalent. For other applications, different parameter values should be considered – the impact of varying the parameters is discussed later in Section 3.5.

Pipeline stage delay overhead has the greatest impact on the results, so we initially focus on the differences due to this. The total pipelining delay overhead for a typical ASIC is 20 FO4 delays due to slow D-type flip-flops and imbalanced pipeline stages, compared to 3 FO4 delays for custom. If an ASIC design uses latches, or faster pulsed flip-flops where needed and useful clock skew, then the pipelining delay overhead may be as low as 5 FO4 delays.

We wish to determine the impact of microarchitecture in the absence of other factors such as slower logic style, so we assume an unpipelined combinational delay of 180 FO4 delays which is reasonable for the custom processors in Table 3.2, though less than we estimated for the ASICs in Table 3.1. To reduce the clock period to 40 FO4 delays, at least nine pipeline stages must be used for our typical ASIC that has a pipeline stage delay overhead of 20 FO4 delays; whereas a custom design with a pipeline stage delay overhead of 3 FO4 delays needs only five pipeline stages. As a large portion of the ASIC's clock period is devoted to the pipelining delay overhead, reducing the delay overhead is very important to improve ASIC performance.

As was assumed by Srinivasan et al. [186] and Harstein and Puzak [89], we assume a value of 1.1 for the latch growth factor η for an integer pipeline. From this and the clock tree and register power in Table 3.6, we estimate the coefficient β for the clock and register power to be 0.05, which is typical for most of the processors of five to eight pipeline stages (see Appendix A.2 for more details).

We assume that ASIC and custom designers can take the same advantage of data forwarding, branch prediction and other techniques to reduce the CPI penalty for deeper pipelines. A CPI penalty per stage of 0.05/stage for both ASIC and custom is assumed.

The power-delay curve fits from geometric programming optimization of ISCAS'85 benchmark c880 will be used to estimate the power savings that can be achieved by voltage scaling and gate sizing. The dynamic power and leakage power are normalized as in Equation (3.15), and the

minimum delay is set to the minimum stage delay from pipelining in Equation (3.7). When Vdd and Vth are chosen to minimize the total power consumption, leakage may be 8% to 21% of the total power consumption as discussed in Section 5.2.6. We assume that leakage is 10% of the total power at the minimum delay, as may be typical for high performance circuits in 0.13um.

We will first examine the maximum performance that can be achieved by ASICs and custom, and then look at metrics that include power consumption as well as performance. Then we will compare the power gap between ASIC and custom at maximum performance and relaxed performance constraints. The results for different metrics are summarized in Table 3.8.

Table 3.8 This table compares the minimum of various metrics for ASIC and custom. Below the normalized comparison versus custom are listed the optimal number of pipeline stages, optimal clock period, power, and energy per operation for optimizing each of the particular metrics as color coded. Note that the numerical solution below for the optimization problem has a non-integer value for the number of pipeline stages n , though in a real circuit n must be integral.

Normalized versus custom					
	T/instruction	P(T/instruction) ³	P(T/instruction) ²	Energy/operation	
Typical ASIC	2.5	3.7	1.7	1.0	1.0
Excellent ASIC	1.2	1.3	1.1	1.0	1.0

Minimum T/instruction (maximizing BIPS)

	# Pipeline Stages n	T_{\min}	Clock Period T (FO4 delays)	T/instruction (FO4 delays)	Power P	Energy / Operation
Typical ASIC	13.3	33.5	33.5	55.8	0.0356	1.987
Excellent ASIC	26.5	11.8	11.8	27.4	0.1174	3.220
Custom	34.2	8.3	8.3	22.4	0.1795	4.020

Minimum P(T/instruction)³ (maximizing BIPS³/W)

	# Pipeline Stages n	T_{\min}	Clock Period T (FO4 delays)	T/instruction (FO4 delays)	Power P	Energy / Operation	P(T/instruction) ³
Typical ASIC	8.5	41.1	64.6	92.2	0.0054	0.497	4225
Excellent ASIC	14.2	17.7	27.3	46.7	0.0143	0.668	1457
Custom	16.6	13.9	21.3	38.9	0.0192	0.747	1129

Minimum P(T/instruction)² (maximizing BIPS²/W)

	# Pipeline Stages n	T_{\min}	Clock Period T (FO4 delays)	T/instruction (FO4 delays)	Power P	Energy / Operation	P(T/instruction) ²
Typical ASIC	6.5	47.8	139.2	184.3	0.0010	0.175	32.3
Excellent ASIC	9.6	23.7	69.6	103.1	0.0020	0.204	21.0
Custom	10.6	20.0	58.6	89.6	0.0024	0.214	19.2

Minimum energy/operation P(T/instruction) (maximizing BIPS/W)

	# Pipeline Stages n	T_{\min}	Clock Period T (FO4 delays)	T/instruction (FO4 delays)	Power P	Energy / Operation
Typical ASIC	1.0	200.0	892.2	936.8	0.00011	0.105
Excellent ASIC	1.0	185.0	825.3	866.5	0.00012	0.105
Custom	1.0	183.0	816.4	857.2	0.00012	0.105

3.4.1 Maximum performance (minimum delay/instruction)

The maximum performance that can be achieved is given by Equation (3.19). The minimum delay per instruction is 22.4 FO4 delays for custom, 27.4 FO4 delays for an excellent ASIC, and 55.8 FO4 delays for a typical ASIC. The corresponding optimal clock period is 8.3 FO4 delays for custom, 11.8 FO4 delays for an excellent ASIC, and 33.5 FO4 delays for a typical ASIC. This compares to the custom 3.466GHz 0.13um Gallatin Pentium 4 which has a clock period of about 9.6 FO4 delays and the high performance, synthesized 520MHz 0.18um STMicroelectronics

iCORE which has a clock period of about 25.6 FO4 delays (FO4 delays were calculated in Table 3.1).

The result for custom of 8.3 FO4 delays per stage is similar to other estimates for the optimal pipeline stage delay to maximize performance. We assumed total unpipelined combinational delay of 180 FO4 delays and 3 FO4 pipeline stage delay overhead for custom. Hrishikesh et al. estimated 8 FO4 delays was optimal for integer benchmarks assuming unpipelined delay¹ of 120 FO4 delays and stage delay overhead of 1.8 FO4 delays [98]. The model used by Srinivasan et al. gave an optimal 9 FO4 delays per stage assuming unpipelined delay² of 110 FO4 delays and stage delay overhead of 3 FO4 delays [186]. From simulation, Harstein and Puzak found that an 8.9 FO4 stage delay was optimal to maximize performance assuming unpipelined delay of 140 FO4 delays and stage delay overhead of 2.5 FO4 delays [89][90].

The typical ASIC with 20 FO4 stage delay overhead is about 2.5× slower than custom with 3 FO4 stage delay overhead, whereas the excellent ASIC with only 5 FO4 stage delay overhead closes the performance gap to 1.2×. These results correspond fairly well with our earlier analysis which estimated a performance gap due to microarchitecture³ and timing overhead⁴ of 2.6× for a typical ASIC to 1.4× for an excellent ASIC [38].

Maximizing performance leads to deep pipelines being optimal, from 34.2 pipeline stages for custom to 13.3 stages for a typical ASIC. For a real world comparison, Intel's Prescott and Cedar Mill Pentium 4 custom processors have 31 integer pipeline stages [97] from Intel's pushing to the

¹ The model used by Hrishikesh et al. was based on the Alpha 21264 processor with seven pipeline stages and combinational delay of 17.4 FO4 delays per stage [98], or about 120 FO4 delays total for the unpipelined combinational delay.

² Srinivasan et al. also assumed a baseline design with 19 FO4 delays per stage, 2 FO4 latch delay and 1 FO4 delay for clock skew and clock jitter [186] with similar instruction execution latencies to Hrishikesh et al. Assuming the baseline has 7 stages, the unpipelined delay is about 110 FO4 delays.

³ Estimated performance gap due to microarchitecture alone was 1.8× for a typical ASIC and 1.3× for an excellent ASIC [38].

⁴ Estimated performance gap due to the timing overhead alone was 1.45× for a typical ASIC and 1.1× for an excellent ASIC [38].

extreme higher performance and higher clock frequency to compete with AMD; while ARM's Cortex-A8 synthesizable processor has thirteen pipeline stages [13].

3.4.2 Maximum BIPS^m/W with voltage scaling and gate sizing

The optimal clock period increases and the optimal number of pipeline stages decreases when power is included in the optimization objective. We can maximize metrics of the form BIPS^m/W with Equation (3.20), minimizing $P(T/\text{instruction})^m$. Results are listed in Table 3.8, except for minimizing power.

A single pipeline stage is optimal to minimize the power consumption ($m = 0$) or energy per operation ($m = 1$). This avoids the additional pipelining power overhead for more registers. The clock period T approaches infinity to minimize the power, and the minimum power approaches about 0.00008 as our dynamic power and leakage power fits tend to a fixed value for large T . With our dynamic and leakage power model parameters, the optimal ratio of T/T_{\min} is 4.46 to minimize the energy per operation, as explained in Appendix A.4. The minimum energy per operation for ASIC and custom designs is 0.105, independent of the pipeline stage delay overhead and total combinational delay. Harstein and Puzak also found that a single pipeline stage was optimal to minimize energy per operation [89]. More than one pipeline stage is optimal to minimize energy per operation when glitching and additional power overheads are accounted for – see Section 3.6 for further discussion.

The optimal clock period of 816 FO4 delays for custom to 892 FO4 delays for ASIC to minimize the energy/operation corresponds to a clock frequency of 31MHz and 28MHz respectively in 0.13um technology with 0.08um channel length. Some applications such as the discrete cosine transform (DCT) and its inverse (IDCT) [62][239][240] can be performed at such low clock frequencies via parallel datapaths, achieving low energy per operation. The DCT and IDCT cores

do have more than one pipeline stage – pipelining is used to implement the algorithm and allow clock gating of units that are not in use to reduce the dynamic power.

Many applications require higher performance, so metrics placing a greater weight on delay are commonly used. Comparing the inverse of BIPS^3/W , a typical ASIC has $3.5\times$ the $P(T/\text{instruction})^3$ of custom, while an excellent ASIC is only $1.3\times$ worse. The gap is larger for BIPS^3/W as the performance gap is multiplied. For both ASIC and custom, the ratio of T/T_{\min} is about 1.5 to maximize BIPS^3/W providing a significant amount of slack for voltage scaling and gate downsizing.

The optimal number of pipeline stages to maximize BIPS^3/W is roughly half the number of stages to maximize performance, as inclusion of power consumption in the objective substantially penalizes very deep pipelines for their additional registers. To maximize BIPS^3/W for custom, the optimal number of pipeline stages from the model is 16.6 and the optimal clock period is 21.3 FO4 delays, which is comparable to Intel’s Conroe Core 2 Extreme with 14 pipeline stages and a clock period of 19.5 FO4 delays. Conroe’s predecessor, the Yonah Core Duo, was specifically designed to be more energy efficient than the Pentium 4 models with 31 integer pipeline stages that maximized performance and have $4.4\times$ more energy per operation [79]. To maximize BIPS^3/W for a typical ASIC, the optimal clock period of 64.6 FO4 delays from the model is very similar to the clock period of a number of the ASICs in Table 3.1, though the number of pipeline stages for these ASICs ranges from 5 to 13.

As the BIPS^2/W metric weights performance less, even more timing slack is used for power reduction with T/T_{\min} ratios of 2.9 for both custom and ASIC. A typical ASIC has $1.7\times$ the $P(T/\text{instruction})^3$ of custom, while an excellent ASIC is only $1.1\times$ worse.

Without voltage scaling or gate sizing, the optimal number of pipeline stages to maximize both BIPS^3/W and BIPS^2/W is unchanged, but the clock period is T_{\min} as no timing slack is used for

power reduction. Thus including voltage scaling and gate sizing increases the optimal clock period $1.5\times$ for BIPS^3/W and $2.9\times$ for BIPS^2/W (the ratio of T/T_{\min}).

To maximize BIPS^3/W without voltage scaling and gate sizing, Harstein and Puzak found from simulation that 7 pipeline stages and a period of 22.5 FO4 delays was optimal [89], while Srinivasan et al. found that a period of 18 FO4 delays was optimal for the SPEC CPU2000 benchmark [186]. This compares to our result of an optimal clock period of 13.9 FO4 delays for custom without voltage scaling and gate sizing. Our optimal clock period to maximize BIPS^3/W is substantially lower as we assume a CPI penalty of only 0.05 per stage, whereas they both have a penalty of about 0.12 per stage. If we instead assume a CPI penalty γ of 0.12 per stage, the optimal clock period is 19.6 FO4 delays to maximize BIPS^3/W .

Graphs of BIPS^3/W and BIPS^2/W versus a given performance constraint are in Appendix A.5. We now look at the minimum power for a given performance constraint.

3.4.3 Minimum power for a given performance constraint

A fixed performance constraint can be used instead of weighting performance in the objective. For a fixed performance constraint, comparing metrics of the form BIPS^m/W comes down to comparing power as the instructions per unit time (BIPS) is fixed. Thus we look at the minimum power consumption to satisfy the performance constraint using Equation (3.21).

The power gap between a typical ASIC and custom ranges from $5.1\times$ at the maximum performance for the typical ASIC down to $1.0\times$ depending on the performance constraint, as shown in Figure 3.7. If the pipeline stage delay overhead is reduce from 20 FO4 delays to 5 FO4 delays, the power gap is at most $1.9\times$ at the maximum performance for the excellent ASIC. The corresponding optimal number of pipeline stages and optimal clock period are shown in Figure 3.8 and Figure 3.9. The optimal clock period T varies almost linearly with the performance

constraint. The clock frequency, $1/T$, is shown in Figure 3.10. There is little difference between the optimal clock period or clock frequency for ASIC and custom.

When the required average time per instruction is large, a typical ASIC has more pipeline stages than custom to get similar timing slack for power reduction, as shown in Figure 3.8. As the required performance increases with lower average time required per instruction, fewer stages is optimal for an ASIC due to the larger pipeline stage delay overhead. As the maximum performance is approached, the number of pipeline stages increases rapidly and the timing slack for power reduction approaches zero with the ratio of T/T_{\min} approaching one (see Figure 3.11).

At a more relaxed performance constraint of 80 FO4 delays per instruction, the power gap between a typical ASIC and custom is only 2.8 \times , and the excellent ASIC is only 1.1 \times worse. At this point, the optimal clock period for the typical ASIC of 55 FO4 delays corresponds to about 450MHz in 0.13um with channel length of 0.08um. Typical low power embedded processors in 0.13um are slower than this, indicating that the power gap between ASIC and custom is not that large in their lower performance market niche.

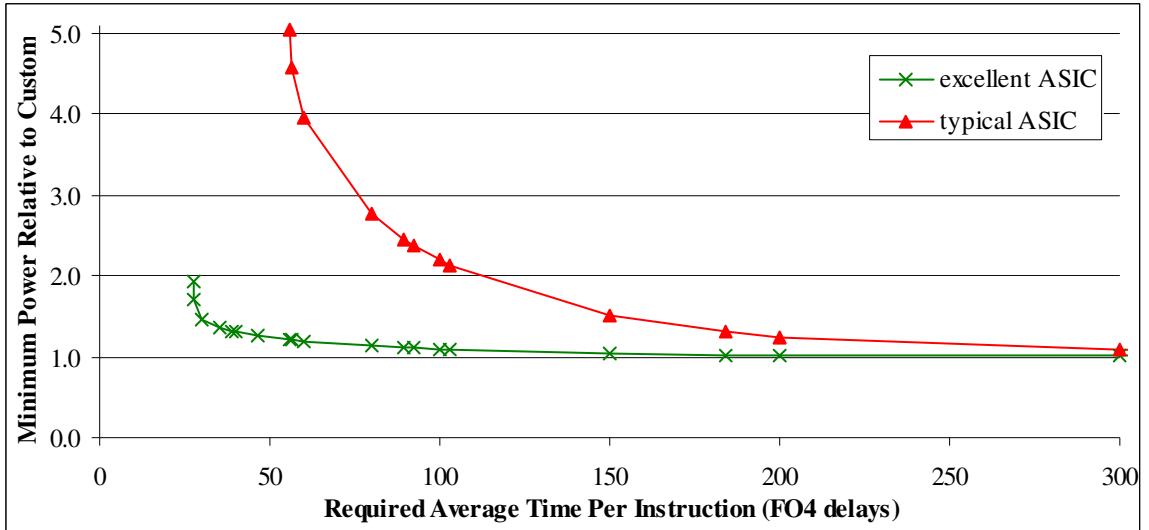


Figure 3.7 Minimum power relative to custom for the parameters listed in Table 3.7.

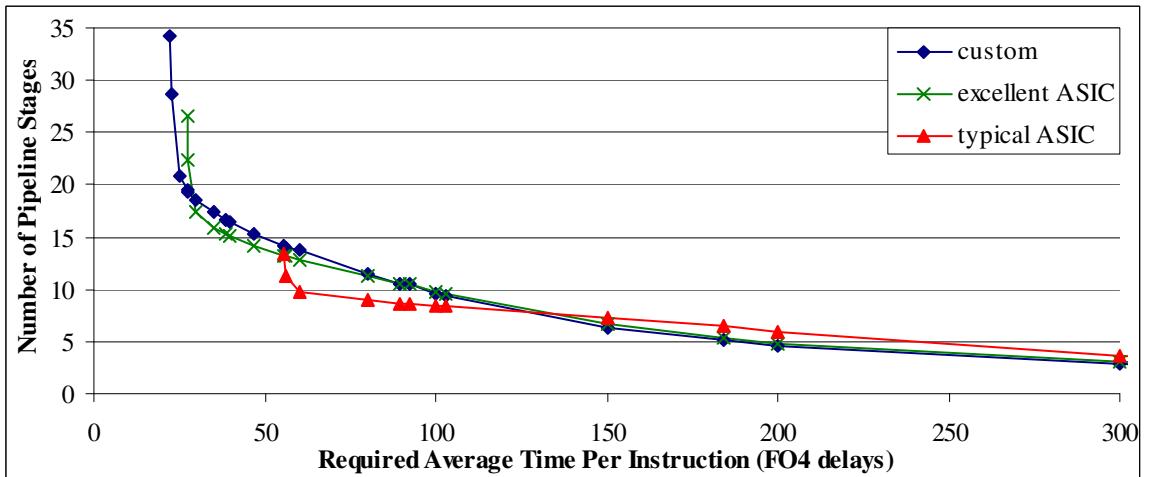


Figure 3.8 Optimal number of pipeline stages n to minimize power.

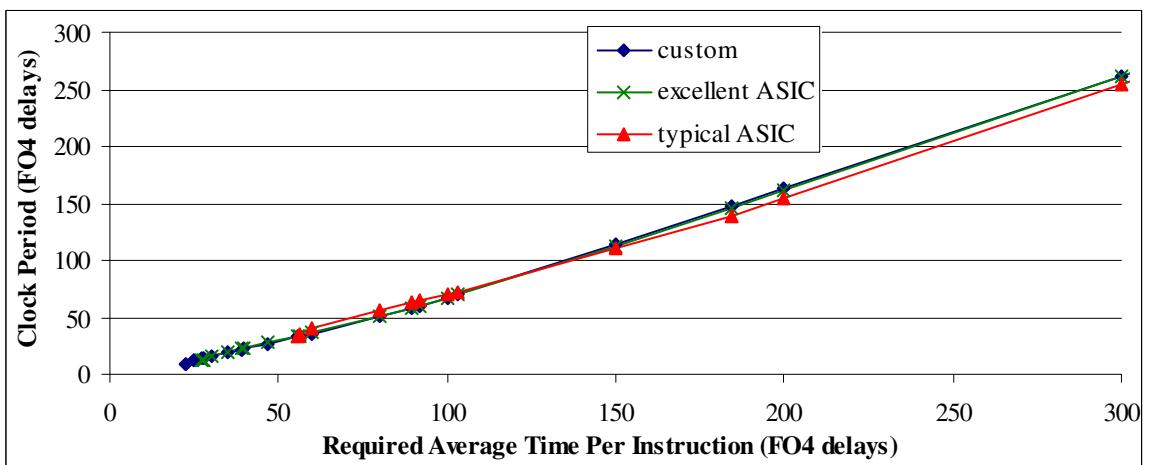


Figure 3.9 Optimal clock period T to minimize power.

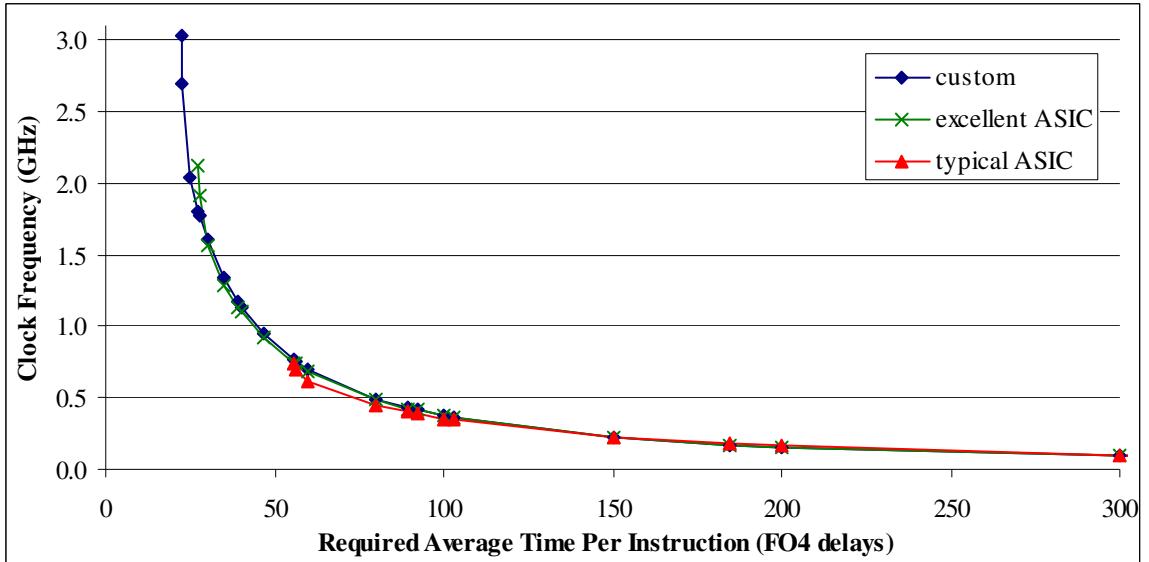


Figure 3.10 Optimal clock frequency to minimize power, corresponding to $1/T$. An FO4 delay of 40ps is used for 0.13um process technology with 0.08um channel length.

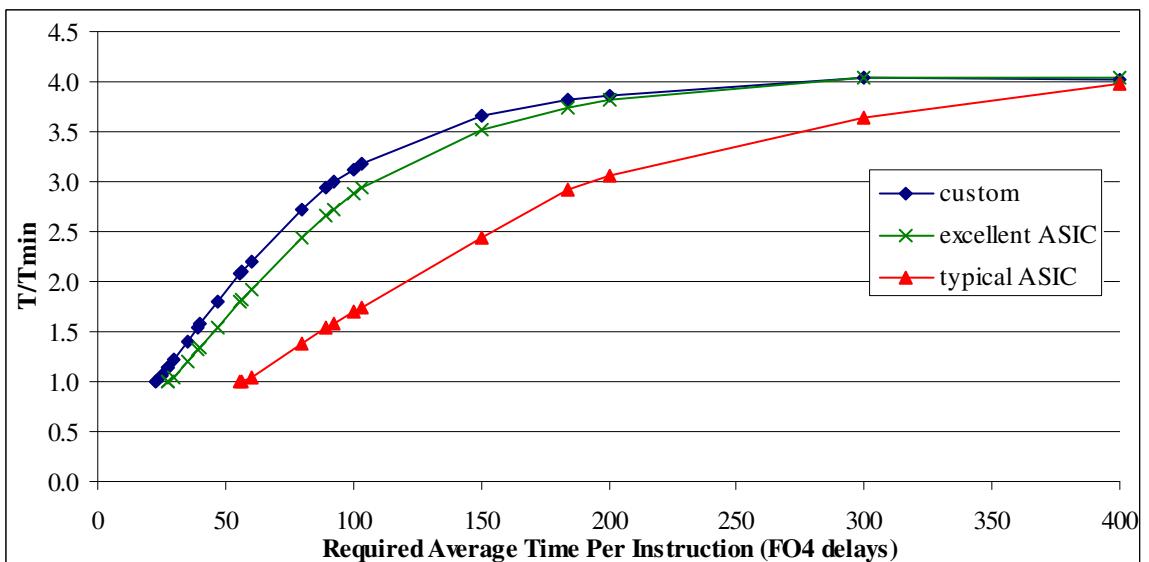


Figure 3.11 The ratio of the clock period to the minimum pipeline stage delay T/T_{min} approaches one as the performance constraint becomes tight. Consequently, there is substantially less timing slack ($T - T_{min}$) for power reduction by voltage scaling and gate sizing. The dynamic power and the leakage power increase significantly as T approaches T_{min} in equations (3.11) and (3.12).

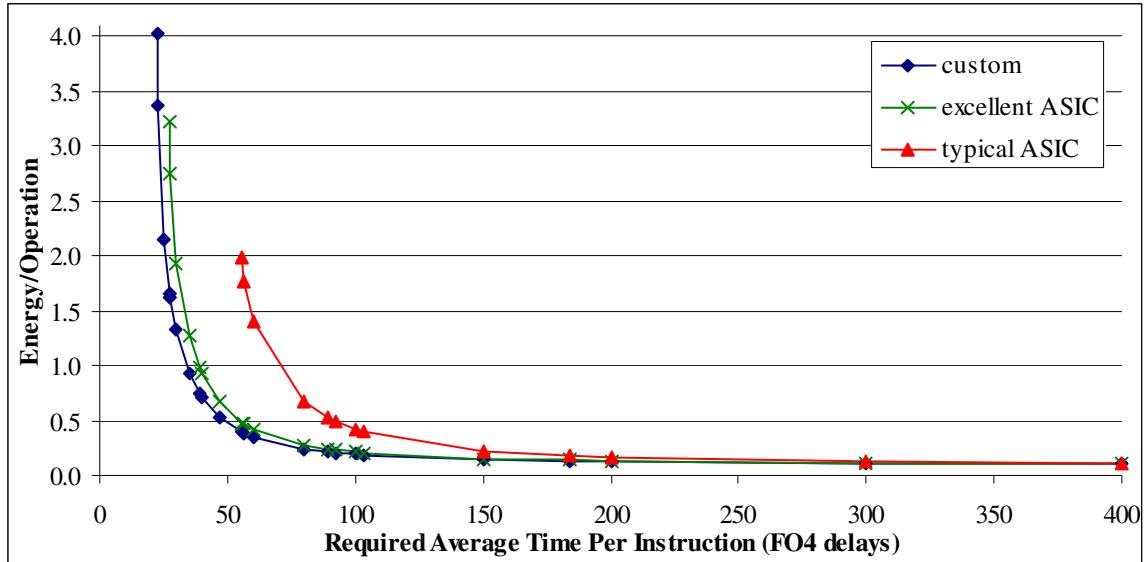


Figure 3.12 The minimum energy per operation at different performance constraints.

The minimum energy per operation for custom at maximum performance is 39× larger than the minimum energy per operation of 0.105 (see Figure 3.12). The energy/operation for the combinational logic is 12× larger, but the power for the registers and clock tree has grown from 5% of the total power with a single pipeline stage to 71% of the total power with 34.6 pipeline stages. The number of registers has increased by a factor of $34.6^{1.1} = 49.3$.

3.5 Other factors affecting the power gap between ASIC and custom

Thus far discussion has focused on the impact of the required performance and the pipeline stage delay overhead on the power gap. We have assumed values for various parameters in the model and we should check if any of those have any significant impact on the results.

3.5.1 Combinational logic delay

The delay of the combinational logic can be reduced by using a different logic style, For example, dynamic domino logic used for the 1.0GHz IBM PowerPC was 50% to 100% faster than static combinational logic with the same functionality [157]. For our model, this corresponds to reducing the unpipelined combinational logic delay from 180 FO4 delays to 120 FO4 delays.

The smaller unpipelined combinational delay has essentially the same effect as a more relaxed performance constraint. Fewer pipeline stages are required to meet the required clock period and the power gap is not as large at a relaxed performance constraint. With $t_{comb\ total}$ of 120 FO4 delays, the maximum performance is about 20% higher and the number of pipeline stages to achieve maximum performance is 18% less. The minimum delay per instruction for a typical ASIC is 47.9 FO4 delays at 11.0 pipeline stages, and for custom it is 17.5 FO4 delays at 28.3 pipeline stages. The performance gap between ASIC and custom of 2.7 \times is 10% worse than with $t_{comb\ total}$ of 180 FO4 delays. At maximum performance the power of the excellent ASIC is 2.0 \times that of custom, and the power of a typical ASIC is 5.8 \times that of custom at its maximum performance – 15% worse than with $t_{comb\ total}$ of 180 FO4 delays.

In practice, only custom designs can take advantage of domino logic and other high performance logic styles. The power gap is 7.9 \times between custom with $t_{comb\ total}$ of 120 FO4 delays and a typical ASIC with $t_{comb\ total}$ of 180 FO4 delays at maximum performance for the typical ASIC. The power gap is 3.9 \times between custom with $t_{comb\ total}$ of 120 FO4 delays and the excellent ASIC with $t_{comb\ total}$ of 180 FO4 delays at maximum performance for the excellent ASIC. Thus the impact of *logic style* on the power gap is 1.6 \times for the typical ASIC and 2.0 \times for the excellent ASIC, but we have not accounted for the additional power that may be required for high performance logic styles.

3.5.2 Clock gating

Without clock gating to avoid dynamic power consumption during pipeline stalls, the optimal number of pipeline stages is up to 20% less. The number of pipeline stages cannot be reduced in an ASIC at a tight performance constraint. At maximum performance for an ASIC, custom still has timing slack to reduce the number of pipeline stages. Consequently, the power gap between ASIC and custom is up to 16% larger when there is no clock gating. Clock gating during pipeline stalls reduces the dynamic power consumption significantly, by 40% for a typical ASIC at its

maximum performance and by about 60% for the excellent ASIC and custom at their maximum performance points.

3.5.3 CPI penalty

Floating point applications may have a far lower rate of branch mispredictions and other pipeline hazards than integer operations [89]. For applications with more pipeline hazards, the CPI penalty per additional pipeline stage is higher and a shallower pipeline is optimal.

If the CPI penalty per stage γ is 0.1 instead of 0.05, there is a larger performance penalty for more pipeline stages. The maximum performance is 26% less for the typical ASIC to 37% less for custom. The power gap between ASIC and custom is up to 20% worse and the optimal number of pipeline stages is up to 15% less for a given performance requirement. The relative power gap between the typical ASIC and custom is only 4.0 \times at the reduced maximum performance for the typical ASIC, and the relative power gap between the excellent ASIC and custom is only 1.7 \times at the reduced maximum performance for the excellent ASIC. The absolute power is up to 2.5 \times higher for deeper pipelines.

3.5.4 Leakage power

As supply voltage is scaled down to reduce power density for deeper submicron technologies, threshold voltage and gate oxide thickness must be reduced to maintain performance. Consequently, subthreshold leakage and gate leakage are more of a problem for deeper submicron technologies when higher performance is required [177]. We have assumed that leakage contributes about 10% of the total power, which is reasonable for 0.13um. To consider deeper submicron technologies or more aggressively scaled threshold voltage and oxide thickness, we consider leakage of 30% of the total power without clock gating.

If leakage is 30% of the total power, $k_{leakage}$ of 0.3, the effect is similar to reducing the clock gating. The power consumption of stalled pipeline stages is not reduced as much, and hence up to

15% fewer pipeline stages is optimal. Though at a tight performance constraint, slightly more pipeline stages can be optimal to provide slack to get the exponential reduction in leakage power. With 30% leakage, the relative power gap between ASIC and custom is from 15% higher to 18% less depending on the performance constraint, as custom gains less benefit from more pipeline stages. The absolute power is from 29% higher to 13% less with the exponential reduction in leakage when more timing slack is available.

Assuming that the threshold voltage is not changed, the leakage is less significant when the dynamic power is higher with higher clock frequency and more pipeline stages. Thus Harstein and Puzak [89] and Srinivasan et al. [186] found that more pipeline stages would be optimal to maximize BIPS^3/W if the leakage power is higher. However, we have normalized leakage to the total power, assuming that it scales up with the activity and dynamic power – using a lower V_{th} to give timing slack to help limit the increase in dynamic power. We actually find that the optimal number of pipeline stages to maximize BIPS^3/W decreases by 5% to 7%, because there is a higher power overhead for pipeline registers from the additional leakage which cannot be avoided during pipeline stalls by clock gating.

For lower performance targets, leakage can be reduced to a small portion of the total power consumption when the chip is active. For example in 65nm, Intel's low power P1265 process reduces leakage by a factor of 300, but has 55% lower saturation drain current and hence is roughly 2.2× slower [112], compared to their higher performance P1264 technology [220]. Thus increased leakage in deeper submicron technologies may not be an issue affecting the power gap between ASICs and custom if high performance is not required.

3.5.5 Register and clock tree power

The defaults of β of 0.05 and η of 0.05 correspond to the register and the clock tree power being 33% of total power when there are eight pipeline stages. If β is 0.1 the register and the clock tree

power is 50% of the total power with eight stages. With β of 0.1, the optimal number of pipeline stages is up to 15% less to reduce the power due to the registers and clock tree; however, at a tight performance constraint the number of stages cannot be reduced. β has little impact on the relative power gap between ASIC and custom, with values ranging from 6% worse to 6% better. β does have a significant impact on the absolute power, increasing it by up to 70% depending on the number of pipeline stages to meet required performance.

If η is 1.7, corresponding to a floating point pipeline, with defaults β of 0.05, the register and the clock tree power is 63% of the total power for an eight stage pipeline. η of 1.7 greatly increases the power penalty for more pipeline stages. Thus the optimal number of pipeline stages decreases by up to 40%, though at a tight performance constraint the number of stages cannot be reduced. At a tight performance constraint, η of 1.7 increases the relative power gap between a typical ASIC and custom by 20% to 6.1 \times , and by 56% for the excellent ASIC to 3.0 \times worse due to the excellent ASIC having substantially more pipeline stages at maximum performance. At maximum performance, η of 1.7 increases the absolute power by 2.7 \times for the typical ASIC to 6.2 \times for custom due to the additional pipeline stages.

3.5.6 Gate sizing and voltage scaling

The power savings with voltage scaling and gate sizing depend on how steep the power-delay banana curve is. The steepness depends on the range of allowable supply and threshold voltages in the process technology and the range of gate sizes in the library. For example at a tight delay constraint, allowing very large gate sizes will give a slight reduction in delay for a large increase in power, compared to using moderate gate sizes. To illustrate the importance of considering low-level circuit optimizations at the same time as higher level microarchitectural choices, we

removed gate sizing and voltage scaling from our model¹. This does not affect the performance, but greatly reduces the power gap.

With no voltage scaling nor gate sizing, the typical ASIC has 1.6× the custom power at the maximum performance for the typical ASIC, at which point the excellent ASIC is only 1% higher power than custom. The typical ASIC still requires 13.3 pipeline stages to achieve maximum performance, but the optimal number of stages for the excellent ASIC and custom drop from a similar value to only about 4 stages at this performance point, as there is no benefit for additional timing slack. With voltage scaling and gate sizing, the power gap between ASIC and custom is 5.1× or 300% of that without voltage scaling and gate sizing.

With no voltage scaling nor gate sizing, the excellent ASIC has 1.6× the custom power at the maximum performance for the excellent ASIC. The power gap with voltage scaling and gate sizing is only about 20% larger for the excellent ASIC versus that without voltage scaling and gate sizing.

We have used curve fits to the dynamic and leakage power from Chapter 6 geometric programming results for c880 with voltage scaling and gate sizing, where Vdd was allowed to range from 1.3V to 0.6V and Vth was allowed to range $\pm 0.13V$ from nominal. We will now compare this to voltage scaling without gate sizing from fits to the optimal Vdd and optimal Vth to minimize total power versus delay estimated from the 0.13um standard cell library data at different Vdd and different Vth in Section 5.2.6. The fits to normalized dynamic power and normalized leakage power were:

$$P_{dynamic}(T) = 0.00151 + \frac{0.130}{\left(\frac{T}{T_{min}} - 0.743\right)^{1.487}} \quad (3.22)$$

¹ Gate sizing and voltage scaling can be removed from the model by setting dynamic power and leakage power to fixed values. This was done by setting $a_{leakage} = a_{dynamic} = 1$, and setting other fitting coefficients for the dynamic and leakage power to zero.

$$P_{leakage}(T) = 0.0148 + \frac{0.224e^{-0.376T/T_{min}}}{\left(\frac{T}{T_{min}} - 0.944\right)^{0.644}} \quad (3.23)$$

The relative root mean square error is 0.5% for the dynamic power fit and 0.3% for the leakage power fit. The minimum delay¹ allowed T_{min} was 0.82, where the optimal Vth was 0.12V and the optimal Vdd was 1.32V, which is at the limit of what is reasonable for the process technology². Using the pipeline model with these fits for the dynamic and leakage power, the power gap is 5.2× between the typical ASIC and custom at maximum performance for the typical ASIC, and 2.2× between the excellent ASIC and custom at maximum performance for the excellent ASIC. This is very similar to what was determined using the fits to the geometric programming optimization results for c880, where the power gap between the typical ASIC and excellent ASIC versus custom were respectively 5.1× and 1.9×.

If we restrict T_{min} to 0.9, where optimal Vdd is 1.15V and optimal Vth is 0.14V, the power-delay curve is less steep. Consequently, the power savings from timing slack are less, giving a power gap of 3.8× between the typical ASIC and custom at maximum performance for the typical ASIC and 1.8× between the excellent ASIC and custom at maximum performance for the excellent ASIC. See Appendix A.6.1 for the revised fits with T_{min} of 0.9.

Gate sizing power-delay curves for the ISCAS'85 benchmarks with the TILOS-like gate sizer from the University of Michigan are shallower than our curves with voltage scaling. Using a power fit for the steepest gate sizing curve in the pipeline model, the power gap is 2.6× between the typical ASIC and custom at maximum performance for the typical ASIC and 2.0× between the excellent ASIC and custom at maximum performance for the excellent ASIC. In contrast using a power fit for the least steep gate sizing curve in the pipeline model, the power gap is 1.6×

¹ Delay was normalized versus the minimum delay with Vdd of 1.2V and Vth of 0.23V.

² From discussion with Bhushan Gupta, a reasonable upper limit for the supply voltage was 10% above the nominal voltage of 1.2V for STMicroelectronics' 0.13um technology.

for both the typical ASIC and excellent ASIC versus custom – no better than when voltage scaling and gate sizing are not allowed. This is not surprising, as the power savings with gate sizing, compared to power simply scaling as $1/T$, are at most 7% for the shallowest power-delay curve, whereas the power savings are 50% for the steepest sizing power-delay curve at $1.5 \times T_{\min}$. See Appendix A.6.2 for the fits with gate sizing.

In summary, the improvements with gate sizing and voltage scaling depend greatly on the steepness of the power-delay curves. Compared to the power gap of $1.6 \times$ with no voltage scaling nor gate sizing, voltage scaling can increase the power gap by $3.2 \times$ and gate sizing can increase the power gap by $1.6 \times$ with timing slack from pipelining.

3.6 Other factors affecting the minimum energy per operation

Glitching and other power overheads have minimal impact on the power gap between ASIC and custom, but they do have a significant impact on the minimum energy per operation. We concluded earlier in Section 3.4.2 that a single pipeline stage was optimal to minimize energy/operation, but this is no longer the case when glitching and other power overheads are included in the pipeline power model. We will now discuss this for completeness.

3.6.1 Glitching in complementary static CMOS logic

Different logic styles have different switching activity and some logic styles suffer from spurious signal transitions, glitches, propagating through the logic. Glitching increases the switching activity in complementary CMOS logic, but by construction glitches may not occur in dynamic logic. Glitches typically cause 15% to 20% of the switching activity in complementary CMOS logic [182].

Dynamic logic precharges before computation and can only be discharged once during computation in a clock cycle. Thus an incorrect signal transition, for example due to noise, cannot be corrected in dynamic logic and causes incorrect computational results. Complementary CMOS

logic is more robust to noise. As dynamic logic precharges every cycle, the switching activity and hence dynamic power is generally substantially higher than in complementary CMOS logic. Consequently, complementary CMOS logic is usually preferred for low power applications where high performance is not required, despite glitching.

As the combinational outputs must be stable by the time the level-sensitive latch opens or the appropriate clock edge arrives at the edge-triggered flip-flop, glitches do not propagate through the pipeline registers to the next pipeline stage¹. Thus pipeline registers reduce the switching activity due to glitches.

Based on experimental data from a dynamic circuit timing simulator, Srinivasan et al. modeled glitching's contribution to the dynamic power of the pipeline's combinational logic as depending linearly on the logic depth [186]. A generated glitch was assumed to have a high probability of propagating through the combinational circuitry of a pipeline stage to the pipeline registers for that stage. The range of combinational logic delays that they consider for a pipeline stage is only 4 to 34 FO4 delays; whereas, we have up to 180 FO4 delays for an unpipelined design. While the glitching power may be fit reasonably well by a linear model over Srinivasan et al.'s 8.5× range, glitching power data for pipelined 32-bit [171] and 64-bit [237] FPGA multipliers has sublinear growth with logic depth over a 32× range (see Appendix A.7.2 and Appendix A.7.3 for details).

The growth of glitching with logic depth depends on a number of factors. Glitches from a gate's output may propagate through the fanout logic gates. The glitch may not propagate if it is not the controlling input of a fanout gate – for example, if the other input of a two input OR gate is 1, or if the other input of a two input AND gate is 0. Inputs from the previous pipeline stage may be

¹ If the glitches did propagate through the registers, the computational result would be incorrect – a larger clock period would be required to ensure that the outputs of the combinational logic have stabilized. The clock period should be at least the longest (critical) path delay from the registers at the start of the pipeline stage, through the combinational logic, to the registers at the end of the pipeline stage. The register setup time during which the input must be stable must also be accounted for.

correlated and signals are correlated on reconvergent combinational paths, affecting the probability that a glitch propagates. If a gate has glitches occur on multiple inputs at the same time, fewer glitches or no glitches may propagate to the gate's output, depending on the arrival time of the glitches, their signal slews, and the gate's functionality. If the delay of paths through the logic are unbalanced, there is more glitching [135]. Some functional blocks have more glitching than others. For example, in an inverse discrete cosine transform (IDCT) core about 37% of the power consumption in the accumulators was due to glitches, whereas glitches accounted for only 14% of the power for the chip as a whole [240]. Some function implementations can avoid glitching. For example, gray code counters can be implemented without glitching as only a single bit changes at each count: 00, 01, 11, 10, 00 ... for a two-bit gray code counter.

3.6.1.1 Glitching power model

To account for glitching in complementary CMOS logic, as glitching only affects the dynamic power for the combinational logic and not the registers, and clock gating does prevent glitching, we use

$$P_{total}(T) = \frac{1}{T_{min}} \left(k_{leakage} \frac{P_{leakage}(T)}{P_{leakage}(T_{min})} \right) (1 + \beta n^\eta) + \frac{1}{T_{min}} \left(\alpha_{clock\ gating}(n) \frac{P_{dynamic}(T)}{P_{dynamic}(T_{min})} (1 - k_{leakage}) \right) (1 + \lambda \alpha_{glitching}(n) + \beta n^\eta) \quad (3.24)$$

where $\alpha_{glitching}(n)$ is the model for glitching as a fraction of the dynamic power due to non-spurious transitions; λ is a constant coefficient, default value of 1, so that we can explore a range of glitching overheads. Appendix A.7 details how we developed models for $\alpha_{glitching}(n)$.

In the vein of Srinivasan et al. [186], we consider a linear model for glitching with logic depth, which is inversely proportional to the number of pipeline stages,

$$\alpha_{glitching}(n) = \frac{2}{n} \quad (3.25)$$

Based on fits to glitching in 32-bit and 64-bit FPGA multipliers, we will also consider a model for glitching growing sublinearly with logic depth,

$$\alpha_{glitching}(n) = 1 - \frac{n}{4}(1 - e^{-4/n}) \quad (3.26)$$

The dynamic power overhead for the combinational logic estimated by these glitching models is shown in Figure 3.13 and Figure 3.14.

The sublinear model in Equation (3.26) provides similar results to the linear model in Equation (3.25) for deeper pipelines, with less than 10% lower glitching for $n = 14$ and more pipeline stages. For shallower pipelines, the glitching estimated by the sublinear model is substantially less than that from the linear model: for a single pipeline stage, the glitching overhead is 75% from the sublinear model, and 200% from the linear model. The sublinear model provides a reasonable lower bound on glitching versus logic depth, while the linear model provides us an upper bound.

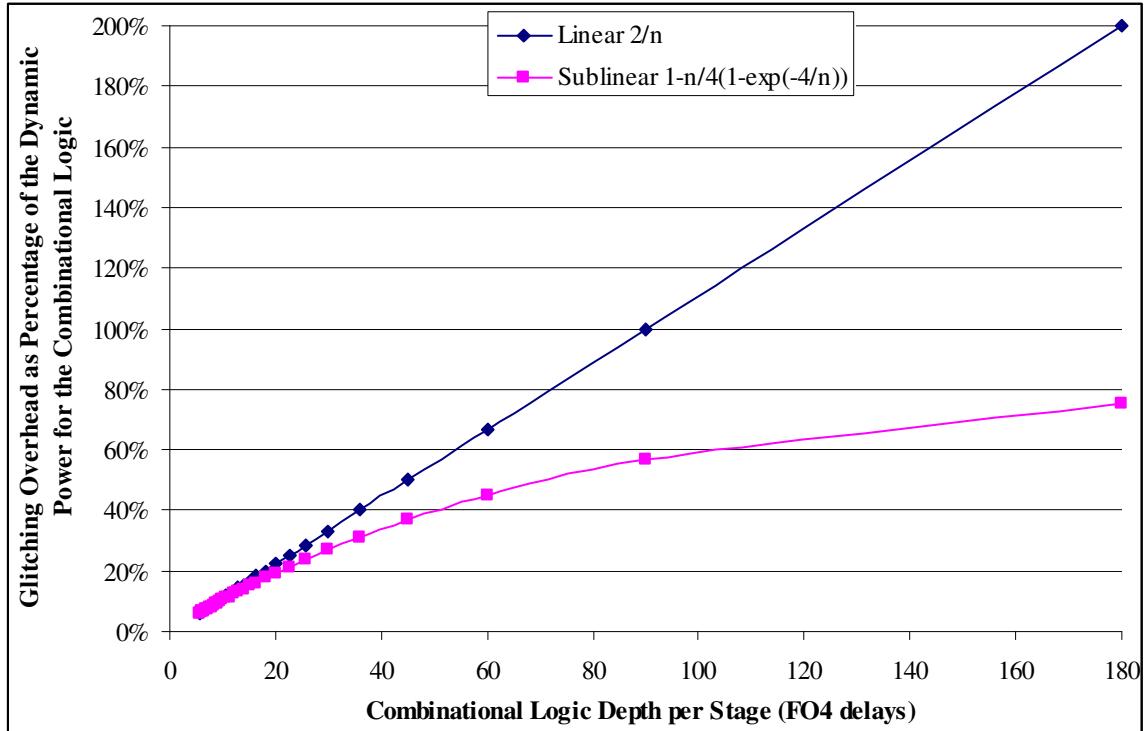


Figure 3.13 Glitching overhead estimated by the linear (Equation (3.25)) and sublinear (Equation (3.26)) glitching models with $\lambda = 1$ versus logic depth.

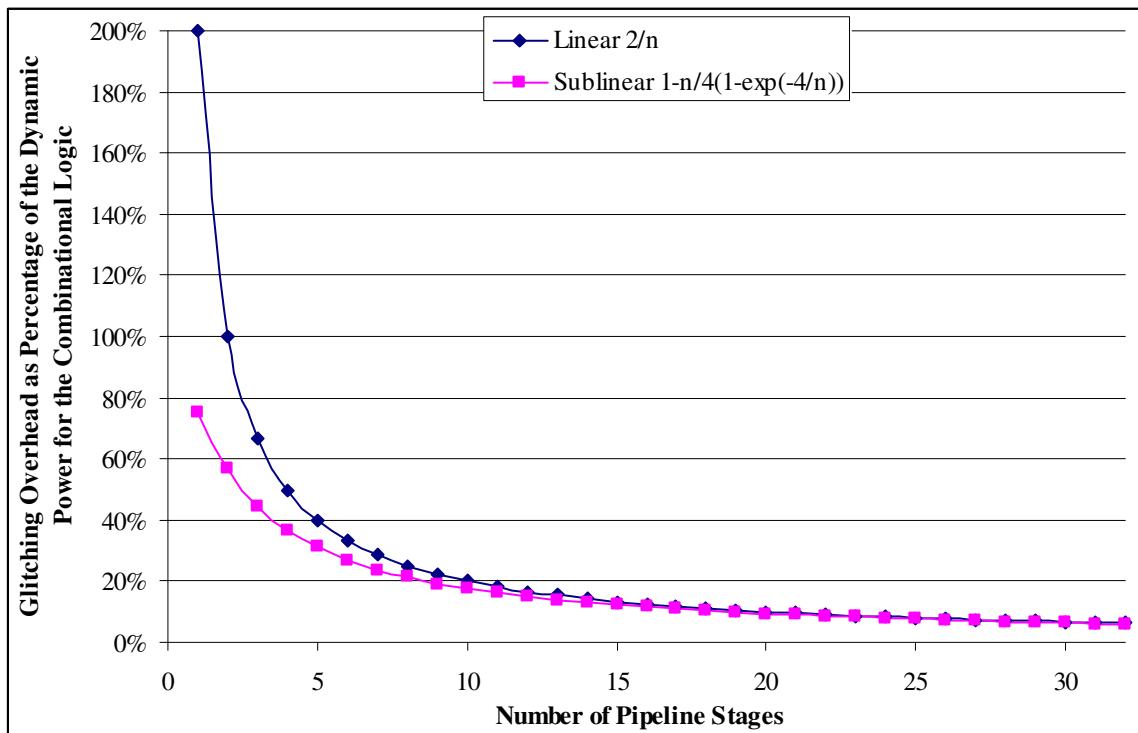


Figure 3.14 Glitching overhead estimated by the linear (Equation (3.25)) and sublinear (Equation (3.26)) glitching models with $\lambda = 1$ versus the number of pipeline stages.

3.6.1.2 Impact of glitching

The maximum performance for a typical ASIC was achieved with 13 pipeline stages, by which point glitching accounts for only 15% of the dynamic power for the combinational logic and only 6% of the total power. Moreover, the optimal number of pipeline stages for custom and ASIC designs are similar (this was shown in Figure 3.8), resulting in similar contributions from glitching. Consequently, glitching increases the power gap between ASIC and custom by at most 5%.

As glitching consumes a far larger percentage of the total power for a single stage pipeline, glitching does have a significant impact on the minimum energy per operation. More pipeline stages are optimal as the additional registers prevent propagation of glitches from the previous pipeline stage.

The minimum energy per operation that can be achieved and the optimal number of pipeline stages depend on how much glitching contributes to the total power. The contribution due to glitching was multiplied by constant coefficient λ in Equation (3.24) to explore the impact of a range of glitching overheads (see Figure 3.15 and Figure 3.16). As in Section 3.4.2, the optimal number of pipeline stages and the minimum energy per operation for ASIC and custom designs are independent of the pipeline stage delay overhead and total combinational delay; however, the optimal delay per instruction (inverse of performance) does vary between ASIC and custom as shown in Figure 3.18. As glitching decreases more slowly with the sublinear glitching model versus the linear model, more pipeline stages are optimal to minimize the energy per operation for the same glitching at a single pipeline stage.

For the linear glitching model in Equation (3.25) with default λ of 1, 5.3 pipeline stages minimizes the energy per operation. With the sublinear glitching model in Equation (3.26) and λ

of 1, 3.9 stages is optimal. The minimum energy/operation is 0.17 and 0.16 respectively, about 60% more than without glitching.

The ratio of the clock period to the minimum pipeline stage delay T/T_{\min} to minimize energy/operation is about 4.4 over a wide range for the linear and sublinear glitching models. The additional timing slack for voltage scaling and gate sizing helps reduce the power consumption due to glitching and the energy per operation. The optimal clock period is large, from 165 FO4 delays for custom with the linear glitching model in Equation (3.25) to 294 FO4 delays for ASIC with the sublinear glitching model in Equation (3.26). The corresponding delay per instruction is 208 FO4 delays for custom with the linear glitching model and 351 FO4 delays for ASIC with the sublinear glitching model. The optimal clock period and delay per instruction are shown respectively in Figure 3.17 and Figure 3.18.

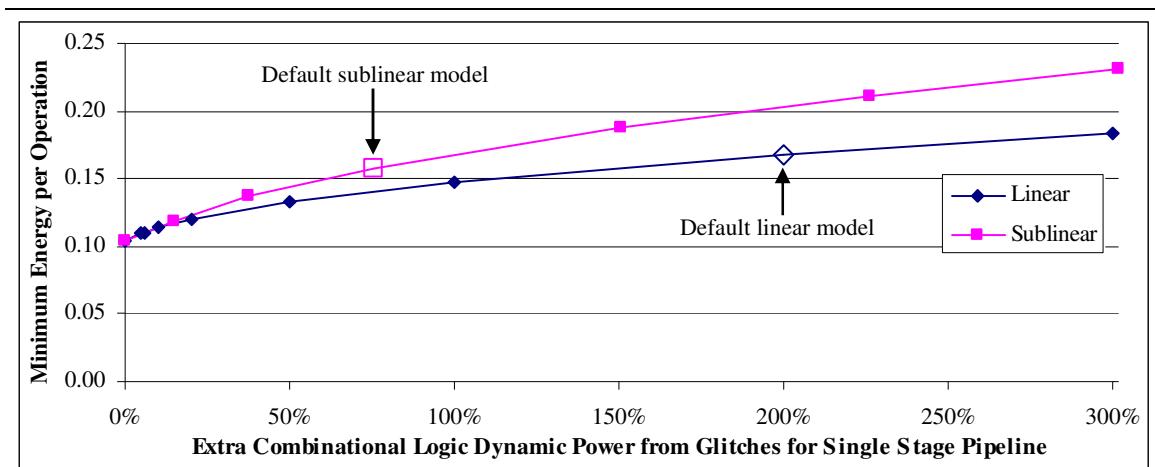


Figure 3.15 The minimum energy per operation for both ASIC and custom with different glitching overheads. Linear and sublinear growth of glitching with logic depth models were used, with default $\lambda = 1$.

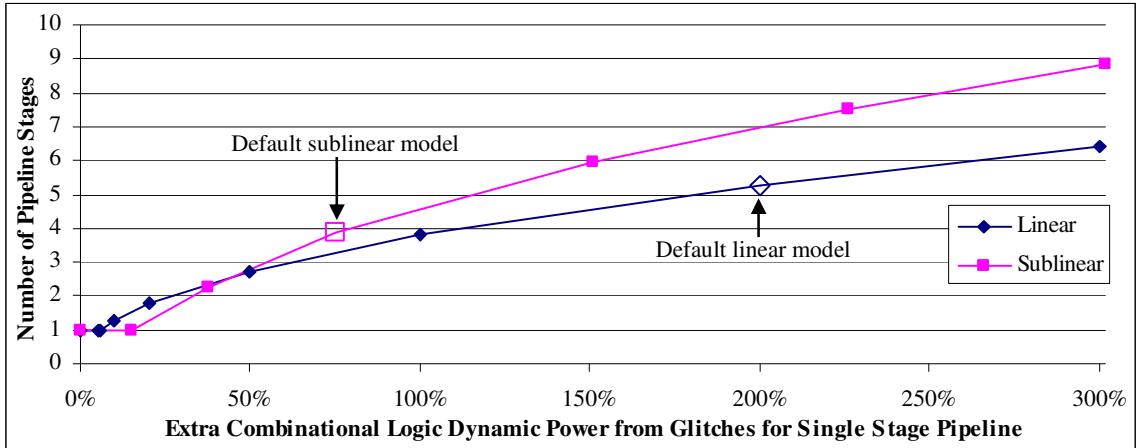


Figure 3.16 The optimal number of pipeline stages for both ASIC and custom to minimize energy/operation for different glitching overheads, with default $\lambda = 1$.

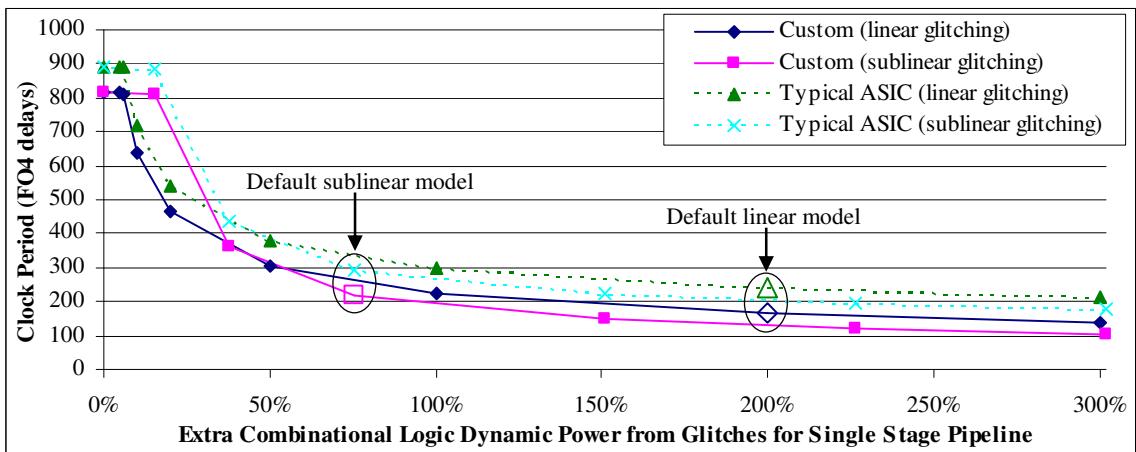


Figure 3.17 The clock period to minimize energy/operation for different glitching overheads, with default $\lambda = 1$. The optimal clock period differs for ASIC and custom, as shown.

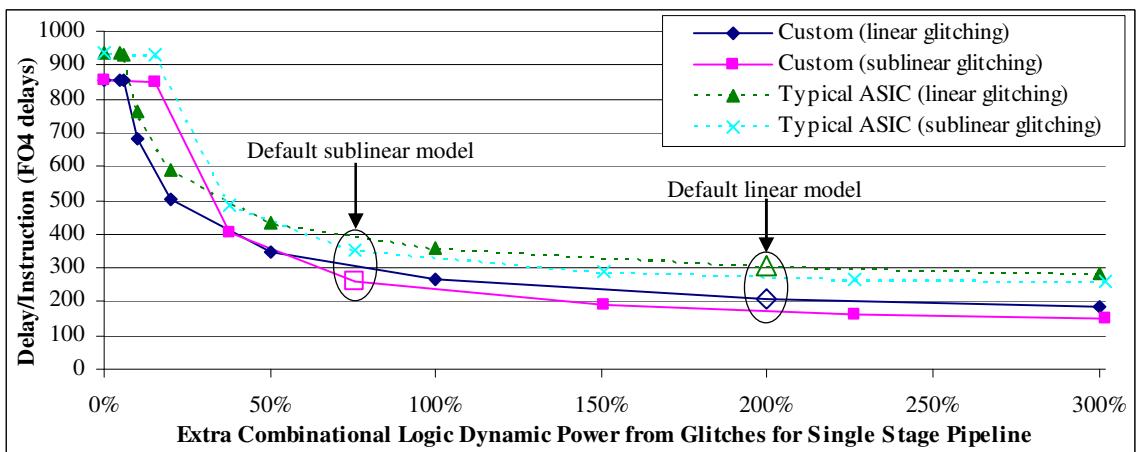


Figure 3.18 The average delay/instruction (inverse of performance) to minimize energy/operation for different glitching overheads, with default $\lambda = 1$. Optimal $T/\text{instruction}$ differs for ASIC and custom, as shown.

3.6.2 Additional power overheads

We have only accounted for power consumption in a processor. There may also be significant power consumption in the memory, off-chip communication, voltage supply, hard drive, video display and other peripheral devices. Memory contributes a significant component of total power consumption for a chip, for example in the StrongARM caches consume 43% of the total power [144]. The typical efficiency of the voltage supply may only be 70% after losses within the power supply itself and distribution to the chip. The voltage regulator for dynamic voltage scaling in Burd's ARM implementation had between 80% and 90% efficiency, depending on the desired voltage [27]. These other power overheads do not affect the minimum power of a processor for a given performance constraint, and thus do not affect the power gap between ASIC and custom designs. However, the additional power does affect the energy per operation.

We add a power overhead to Equation (3.15),

$$P_{total}(T) = \frac{1}{T_{min}} \left(\alpha_{clock_gating}(n) \frac{P_{dynamic}(T)}{P_{dynamic}(T_{min})} (1 - k_{leakage}) + k_{leakage} \frac{P_{leakage}(T)}{P_{leakage}(T_{min})} \right) (1 + \beta n^\eta) + P_{overhead} \quad (3.27)$$

The extra power consumption is assumed to be independent of the processor's clock frequency.

We consider a range of additional power overheads. Units for power consumption have been arbitrary as power was normalized in Equation (3.15). The power consumption has ranged from 0.18 for custom at maximum performance down to 0.00011 for a typical ASIC at minimum energy/operation. These values may correspond to 180W in a high performance PC processor to 0.11W in a low power embedded processor (multiplying by 1000). The power overheads may have a similar range, from say 100W or 0.1 in our normalized units for power, to 0.1W or 0.0001 in our normalized units. We extend this range for the power overhead down to 0.000001 to illustrate the point at which more than one pipeline stage is optimal for energy efficiency.

Very small power overheads have little impact on the minimum energy per operation. The minimum energy per operation steadily increases with the power overhead with larger power overheads, as shown in Figure 3.19. A power overhead of 0.1W or 0.0001 contributes about 20% of the total power in the minimum energy per operation processor, as shown in Figure 3.20. A power overhead of 100W or 0.1 contributes about 70% to 80% of the total power, dominating the power contribution from the processor and pushing the optimal point for the processor close to the maximum performance that can be achieved.

Two pipeline stages is optimal to minimize energy/operation at a power overhead of about 0.000035 or 0.035W, which is quite a low power overhead (see Figure 3.21). For larger power overheads, the optimal number of pipeline stages to minimize energy/operation is less for a typical ASIC than for custom due to the higher pipeline stage delay overhead in the typical ASIC. If the extra power consumption is 0.1 or 100W, 10.3 pipeline stages is optimal for the typical ASIC, 16.1 stage for the excellent ASIC, and 18.5 for custom. Correspondingly, the optimal clock period is smaller and the ratio of the clock period to the minimum pipeline stage delay (T/T_{\min}) approaches one as a high performance design becomes optimal to minimize energy/operation with larger power overheads. See Appendix A.8 for graphs of the optimal clock period, delay per instruction, and ratio of T/T_{\min} versus the power overhead.

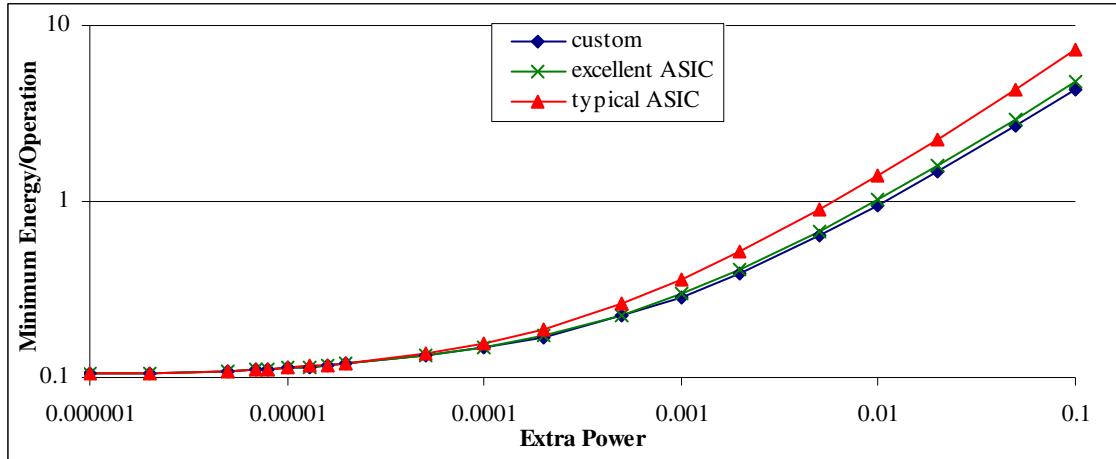


Figure 3.19 The minimum energy/operation is shown for different power overheads.

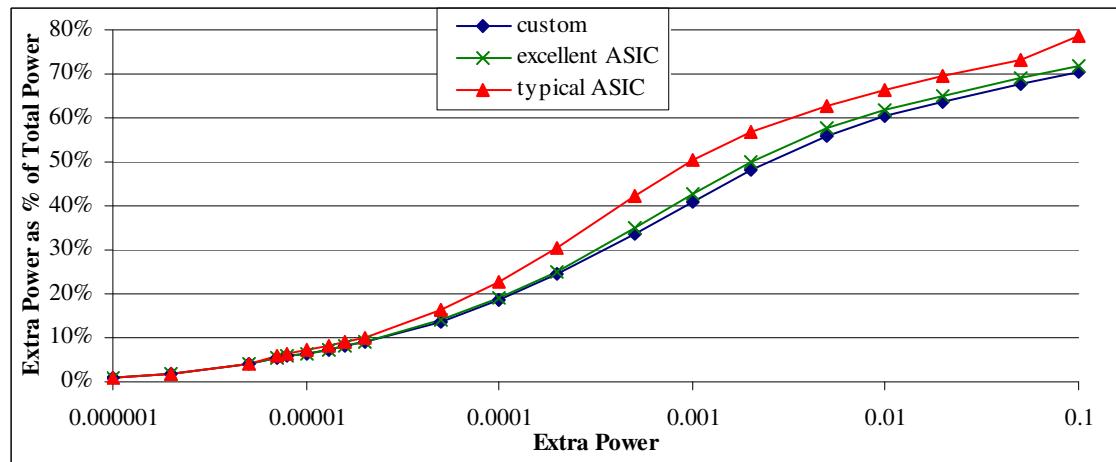


Figure 3.20 This graph shows the percentage of total power consumption due to the additional power overheads for the minimum energy/operation values.

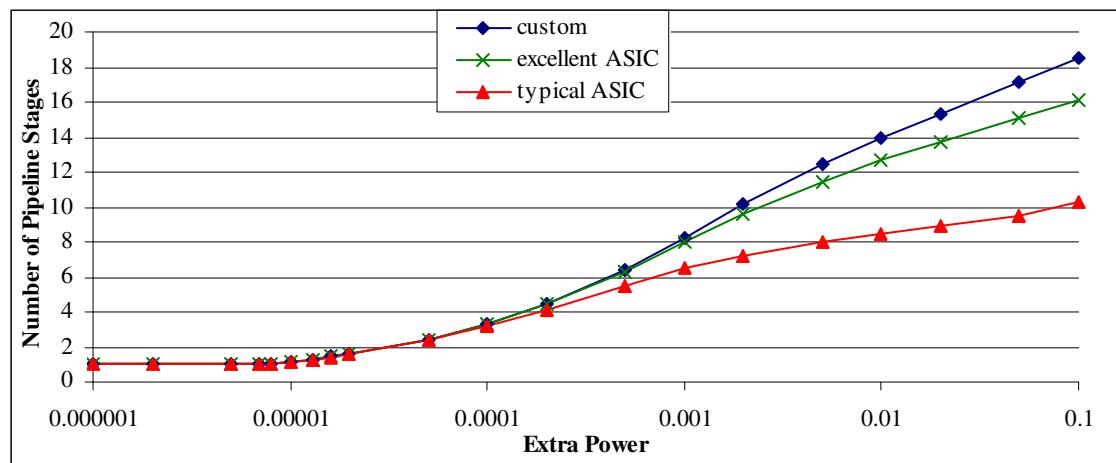


Figure 3.21 The number of pipeline stages to minimize energy/operation is shown for different power overheads.

3.6.2.1 The optimal number of pipeline stages for next generation many core chips

The power overhead depends on the application that a circuit is designed for. For example, to examine multi-core computing Intel has fabricated a 45nm chip with 80 processor cores that can be run in parallel [175]. This prototype is seen as being two generations ahead of product development. They examined power consumption from 20W to 240W, corresponding perhaps to laptop and server chips respectively.

For a server, 200W overhead or 2.5W per processor is easy to envisage. If we assume an overhead of 2W or 0.002, 10.2 pipeline stages is optimal to minimize energy per operation for custom. This suggests that high performance future generation chips with many cores will only have slightly less pipeline stages than seen in today's processors: 14 stages for Intel's Core 2 Duo and 12 stages in AMD's Athlon 64 chips.

In contrast, the power overhead in a laptop may be only 10W or 0.12W per processor. If we assume an overhead of 0.1W or 0.0001, 3.3 pipeline stages is optimal to minimize energy per operation for custom. Thus for energy efficiency, the processor cores on a many core chip for a laptop and other lower power applications will have a closer resemblance to today's low power embedded processors with fewer pipeline stages.

3.6.2.2 Combined impact of glitching and additional power overheads

More pipeline stages are optimal when glitching or the other power overheads are added to the power model. Somewhat deeper pipelines are optimal when these two factors are both considered. If the power overhead is high, a deeper pipeline is optimal which substantially reduces the glitching and thus glitching has little additional impact in this case. We illustrate the combined effect on the optimal number of pipeline stages for the earlier example with the 80 core Terascale chip [175].

With 2W or 0.002 power overhead, the optimal number of pipeline stages for custom is 11.4 if glitching grows linearly with logic depth with Equation (3.25) or 11.2 if glitching grows sublinearly with logic depth with Equation (3.26). This is only about one stage more than when we only considered the power overhead as glitching is less than 5% of the total power, while the power overhead is about 46% of the total power.

With 0.1W or 0.0001 power overhead, the optimal number of pipeline stages for custom is 6.1 if glitching grows linearly with logic depth with Equation (3.25) or 5.2 if glitching grows sublinearly with logic depth with Equation (3.26). Glitching has a more significant impact here as glitching contributes about 16% of the total power, while the power overhead is less than 11% of the total power.

3.7 Summary

ASICs have a substantially higher pipelining delay overhead than custom circuits, which reduces the benefit of additional pipeline stages and substantially reduces the timing slack available for power reduction. With pipelining to provide timing slack for power reduction, a typical ASIC with a 20 FO4 pipeline stage delay overhead may have 5.1 \times the power of a custom processor with 3 FO4 pipeline stage delay overhead at a tight performance constraint. The power gap is less at more relaxed performance constraints, reducing to 4.0 \times at only 7% lower performance. The delay overhead in an ASIC can be reduced by using latches or faster pulsed flip-flops on critical paths with useful clock skew, instead of slower D-type flip-flops that don't allow slack passing between unbalanced pipeline stages. If the ASIC pipeline stage delay overhead can be reduced to 5 FO4 delays, the gap is only 1.9 \times .

The difference in pipeline delay overhead is the most significant cause of the power gap between ASIC and custom with pipelining. We analyzed the impact of other factors such as smaller combinational logic delay, glitching in complementary CMOS logic, increased CPI penalty per

pipeline stage, no clock gating to avoid dynamic power during pipeline stalls, higher leakage power, and increased clock tree and register power. These other factors each increase the power gap by at most 20%.

Only custom designs can make use of high performance logic styles such as dynamic domino logic. If the custom design reduces the combinational logic delay with a high performance logic style, the ASIC may have up to 2.0 \times larger power gap at maximum performance, ignoring additional power consumption of high performance logic styles. We attribute this factor to logic style rather than microarchitecture.

Inclusion of voltage scaling and gate sizing in the pipeline model has a substantial impact on the power gap between ASIC and custom. The power gap with no voltage scaling nor gate sizing is only 1.6 \times , versus 5.1 \times with voltage scaling and gate sizing which is three times larger. It is important to consider high level circuit techniques to provide timing slack along with these low level circuit techniques that can reduce power if there is timing slack. Compared to the power gap of 1.6 \times with no voltage scaling nor gate sizing, voltage scaling can increase the power gap by 3.2 \times and gate sizing can increase the power gap by 1.6 \times with timing slack from pipelining. The improvements with gate sizing and voltage scaling depend greatly on the steepness of the power-delay curves, which depend on the range of allowable supply and threshold voltages in the process technology and the range of gate sizes in the library.

Harstein and Puzak found that a single pipeline stage was optimal to minimize energy per operation [89], but they did not consider glitching or additional power overheads for memory, off-chip communication, voltage supply, peripheral devices, et al. Pipeline registers reduce the switching activity due to glitches by preventing glitches propagating to the next pipeline stage. We found that about four to five pipeline stages is optimal when glitching is accounted for. The impact of other power overheads depends on their net power consumption. We predict that for

energy efficiency, future processors for servers will have only slightly shallower pipelines than today’s high performance microprocessors, while processors for laptops and other low power applications will resemble today’s low power embedded processors. Accounting for glitching and the other power overheads, 11 integer pipeline stages may be optimal to minimize energy per operation for a server chip with many processor cores, whereas about 6 stages would be optimal for a laptop CPU with a similar number of processor cores.

3.7.1 Comparison to related research

There is one other recent paper to our knowledge that has considered pipelining to provide timing slack for voltage scaling, though gate sizing was not considered and a fixed clock frequency of 2GHz was assumed. Heo and Asanović [92] found a 5× power reduction from voltage scaling with timing slack from pipelining to reduce the combinational logic delay of 24 FO4 delays per stage to 6 to 8 FO4 delays¹. However, the number of registers per pipeline stage was assumed to be constant with pipeline depth, and there was assumed to be no CPI performance penalty for additional pipeline stages. For the equivalent performance requirement with our model², we find a maximum power reduction of only 1.8×. The power and delay overhead for more registers and the CPI penalty for more pipeline stages reduces the power savings.

¹ Using 70nm transistor models, they scaled from a baseline Vdd of 0.9V with Vth of 0.21V for NMOS and –0.24V for PMOS to Vdd of about 0.3V with Vth of 0.17V for NMOS and –0.20V for PMOS.

² We assume a 37.1 FO4 delays/instruction performance constraint from the timing overhead and the CPI penalty for the 7.5 pipeline stages required to reduce t_{comb} from 180 to 24 FO4 delays.

Chapter 4. Linear Programming for Gate Sizing

For many ASIC designs, gate sizing is the main low level design technique used to reduce power. Gate sizing is a classical circuit optimization problem for which the same basic method has been used for the past 20 years. The standard approach is to compute a sensitivity metric, for example for the power versus delay tradeoff for upsizing, and then greedily resize the gate with highest sensitivity, iterating this process until there is no further improvement. Such methods are relatively fast, with quadratic runtime growth versus circuit size, but they are known to be suboptimal. The challenge has been to find a better approach that still has fast runtimes.

Our linear programming approach achieves 12% lower power even on the smallest ISCAS'85 benchmark c17, as shown in Figure 4.1. The linear program provides a fast and simultaneous analysis of how each gate affects gates it has a path to. Versus gate sizing using the commercial tool Design Compiler with a 0.13um library, we achieve on average 12% lower power at a delay constraint of 1.1 times the minimum delay (T_{\min}), and on average 17% lower at $1.2T_{\min}$ – in one case 31% lower. The runtime for posing and solving the linear program scales between linearly and quadratically with circuit size.

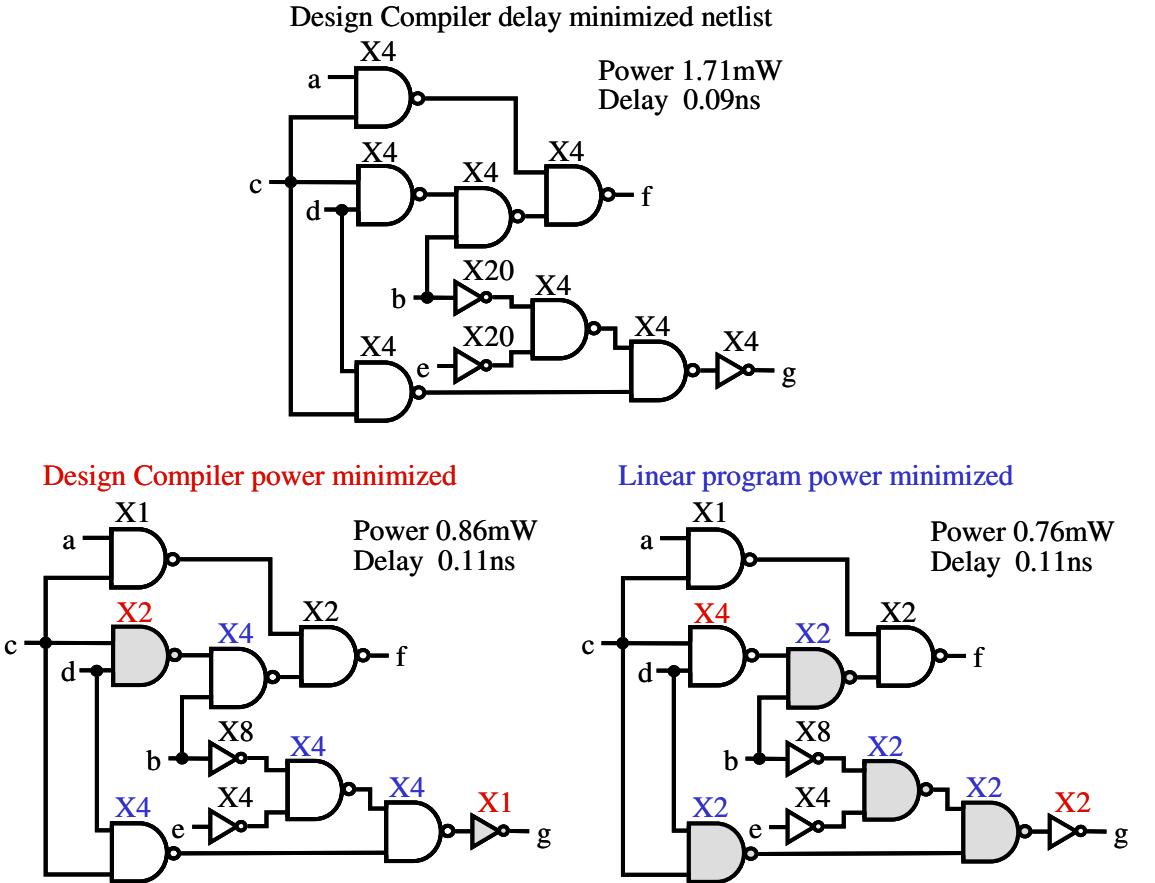


Figure 4.1 At a delay constraint of $1.2T_{\min}$ for ISCAS'85 benchmark c17, we achieve lower power than Design Compiler. The two shaded gates on the lower left circuit are suboptimally downsized by Design Compiler. In contrast, the linear programming approach downsizes four of the NAND2 gates and achieves 12% lower power.

4.1 Introduction

We wish to find the minimum power for a circuit to satisfy given delay constraints. To limit the solution space, we consider a gate-level combinational circuit with fixed circuit topology. The circuit may be represented as a directed acyclic graph (DAG), where each node is a logic gate and edges are connections between gates. The logic gate at each node is fixed; we do not allow nodes to be inserted or removed, or graph edges to change – for example pin swapping is not allowed.

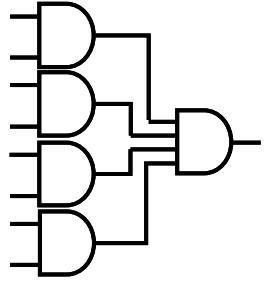
For each logic gate, there are a number of logic cell implementations that may be chosen from the available standard cell libraries. Each cell has different delay, dynamic power, and static power characteristics. These characteristics are determined by factors such as the gate oxide thickness, width, length and threshold voltage of transistors composing the logic cell; transistor topology –

for example stack forcing [152] and alternate XOR implementations; and the supply voltage. We shall limit discussion in this chapter to gate sizing. However, the same delay and power tradeoffs need to be considered for all these factors, and the optimization problem does not fundamentally differ except in the case of gate supply voltage, where there can be topological constraints on choosing the supply voltage of a gate. Our objective is to minimize the power subject to a delay constraint, but the approach herein is equally applicable to minimize the area subject to a delay constraint. Using libraries with multiple threshold voltages and multiple supply voltages is a relatively new low power technique and will be discussed in Chapter 7.

4.1.1 Gate sizing approaches

Gate sizing algorithms have changed little in the past 20 years. In 1985, Fishburn and Dunlop proposed a fast method (TILOS) to minimize area and meet delay constraints, greedily picking the transistor with maximum delay_reduction/transistor_width_increase at each step [64]. Variants of this are still standard in commercial sizing tools.

Approaches similar to TILOS can be used to minimize power when gate sizing. Srivastava et al. used max delay_reduction/power_increase to meet delay constraints, after reducing power by assigning gates to low supply voltage (Vdd) [188]. Downsizing a gate gives a linear reduction in the power to charge and discharge its internal capacitances and input pin capacitances, as the capacitance is proportional to the transistor widths in the gate. Leakage power is also reduced linearly as gate width is reduced. See Chapter 5 for more detail on how delay and power depend on gate size and other factors.



AND2 cell choices:
 AND2X1 – delay 2ns, power 1mW
 AND2X2 – delay 1ns, power 2mW } $-\frac{\Delta P}{\Delta d} = 1\text{mW}/1\text{ns}$

AND4 cell choices:
 AND4X1 – delay 2ns, power 2mW
 AND4X2 – delay 1ns, power 4mW } $-\frac{\Delta P}{\Delta d} = 2\text{mW}/1\text{ns}$

Figure 4.2 This simple example shows that greedily choosing the gate with the maximum sensitivity is suboptimal. If all the gates are initially size X2, the critical path is 2ns and power is 12mW. Consider a 3ns delay target. Picking the max power_reduction/delay_increase sensitivity results in sizing down the AND4 gate, giving total power of 10mW. If the four AND2 gates are sized down instead, the power is only 8mW.

Greedy heuristics that pick the gate with the maximum sensitivity fail to consider the whole circuit and are suboptimal – for example see Figure 4.2. The challenge is to find a better approach with a global view of the circuit that has fast runtimes.

Several groups have used convex optimization to find a globally optimal solution. Convex optimization requires convex delay and power models, such as linear or posynomial models. In our experience, linear models are inaccurate. Least squares fits of linear models versus gate size and load capacitance of 0.13um library data had delay inaccuracy of 19% to 30%, and least squares fits of piecewise linear models also has sizable error of 10% to 30% [174]. The analysis in [174] assumed a fixed input slew of 0.07ns, so these errors will be even larger once variable slew is taken into account. Linear program (LP) solvers with linear models can scale to problems with millions of variables. Higher order convex models, such as posynomials [113], are at best accurate to within 5% to 10% as shown in Section 6.2.2 and [113][174][207]. The accuracy is limited because real data for delay and power is not a convex function of gate size – standard cell layouts change significantly as the gate width changes, due to transistor folding for layout of larger cells and other cell layout concerns. Sacrificing delay accuracy is unacceptable, when a 10% delay increase can give 20% power savings (e.g. compare power at $1.1T_{min}$ and $1.2T_{min}$ in Table 4.2). Optimization with posynomial models requires using a geometric program solver with runtimes that scale cubically, as was detailed in Section 6.6. Thus geometric programming optimization of circuits of tens of thousands of gates or more is computationally infeasible. In

addition, convex models must assume at least a piecewise continuous range for optimization variables that are typically discrete, which introduces suboptimality when the resulting intermediate values must be rounded up or down in some manner to a discrete value – though this is less of an issue for a library with very fine-grained sizes.

It is possible to formulate a linear program to perform optimization at a global level with more accurate delay and power models. The basic approach is to use the linear program to specify the delay constraints, and the power and delay changes if the cell for a gate is changed. A heuristic is required to choose which cell change is the best to encode in the linear program, for example the cell that gives the best power_reduction/delay_increase. The solution to the linear program indicates which cells may be changed, or how much timing slack is available to change a cell to one that consumes less power. Cells with sufficient slack are then changed. This procedure of specifying the best alternate cells in the linear program, solving it, and assigning cell changes is iterated.

The linear program formulation requires some timing slack for the circuit to be downsized and upsized in an iterative manner to converge on a good solution. A 0.13um standard cell library was used. At a delay constraint of 1.1 times the minimum delay (T_{\min}), we achieve on average 12% lower power by sizing than Design Compiler at the same delay constraint, and on average 17% lower at $1.2T_{\min}$ – in one case 31% lower. Design Compiler is the commercial EDA synthesis tool which is most commonly used in industry, and it is generally considered to produce high quality results compared to other EDA tools [72]. The timing and power results for the optimized netlists have been verified in Design Compiler. Likewise, this sizing approach shows similar benefits over the TILOS sizing optimization approach used by Srivastava, Kulkarni, Sylvester and Blaauw in [124] and [188].

An overview of gate sizing approaches along the lines of TILOS is provided in Section 4.2. The initial work on the linear programming formulation, and the problems with delay and power accuracy that were encountered are described in Section 4.3. Then Section 4.4 discusses how the delay and power accuracy were addressed in the linear program. The optimization flow is detailed in Section 4.5. Section 4.6 compares our gate sizing results versus gate sizing in Design Compiler, and then Section 4.7 discusses computational runtime. Section 4.8 concludes with a short summary of this gate sizing work.

4.2 Overview of TILOS gate sizing

Starting with a circuit that violates delay constraints, TILOS aims to meet the delay constraints with the minimum increase in area. Transistors on critical paths, that is paths that don't satisfy the delay constraint, were analyzed with the following sensitivity metric [64]

$$Sensitivity_{reduce\ delay} = -\frac{\Delta d}{\Delta w} \quad (4.1)$$

where Δd is the change in delay on the path and $\Delta w > 0$ is the increase in transistor width. Δd was determined from convex delay models for the distributed RC network representing the circuit. The total circuit area was measured as the sum of the transistor widths, so the aim was to get the best delay reduction with the minimum transistor width increase. The transistor with the maximum sensitivity was upsized to reduce the path delay. This greedy approach proceeded iteratively upsizing transistors with maximum sensitivity until delay targets are met, or there are no upsizing moves to further reduce delay [64].

Dharchoudhury et al. used a similar approach for transistor-level sizing of domino circuits in 1997. By this time, distributed RC networks had fallen out of favor in industry due to inaccuracy and it was essential for timing accuracy to model individual timing arcs. The sensitivity to upsizing a transistor was computed by [56]

$$Sensitivity_{reduce\ delay} = -\frac{1}{\Delta w} \sum_{l \in timing_arcs} \frac{\Delta d_l}{Slack_l - Slack_{min} + k} \quad (4.2)$$

where $Slack_{min}$ is the worst slack of a timing arc seen in the circuit; $Slack_l$ is the slack on the timing arc; Δd_l is the change in delay on the timing arc if the transistor is upsized ($\Delta w > 0$); and k is a small positive number for numerical stability purposes. The weighting $1/(Slack_l - Slack_{min} + k)$ more heavily weights the timing arcs on critical paths.

Srivastava and Kulkarni in [124] and [188] used a delay reduction metric similar to Dharchoudhury et al., for gate-level sizing to minimize power and meet delay constraints in their TILOS-like optimizer. The sensitivity metric was [188]

$$Sensitivity_{reduce\ delay} = -\frac{1}{\Delta P} \sum_{l \in timing_arcs} \frac{\Delta d_l}{Slack_l - Slack_{min} + k} \quad (4.3)$$

where $\Delta P > 0$ is the change in total power for upsizing a gate. The same analysis was used in [183] for the delay versus leakage power trade-off. The timing slack on a timing arc through a gate from input i to output j is computed as [125]

$$Slack_{arc\ ij} = (t_{required\ at\ output\ j} - t_{arrival\ at\ input\ i}) - d_{arc\ ij} \quad (4.4)$$

where $t_{required\ at\ output\ j}$ is the time the transition must occur at gate output j for the delay constraint to be met on paths that include the timing arc; $t_{arrival\ at\ input\ i}$ is the arrival time at input i ; and $d_{arc\ ij}$ is the delay on the timing arc. The slack is the maximum increase in delay of the timing arc that will satisfy the delay constraints, and it will be negative if a delay constraint is not met on a path through the gates with that timing arc. Note here that the impact on delay of slew changing is not included in Equation (4.3), and that for accuracy the delay change on the timing arc Δd_l should include the delay change of the gate that drives gate input i due to the change in input capacitance of pin i as the gate is upsized. This TILOS gate sizing approach does not include gate downsizing,

but that is not important as the starting point for the TILOS gate sizer is with all gates at minimum size.

Similar metrics have also been used for greedy power minimization approaches when delay constraints have been met. For example, for leakage reduction [232]

$$Sensitivity_{reduce\ power} = -\frac{\Delta P_{leakage}}{\Delta d} \quad (4.5)$$

and for power reduction [188]

$$Sensitivity_{reduce\ power} = -\Delta P \sum_{l \in timing_arcs} \frac{Slack_l}{\Delta d_l} \quad (4.6)$$

The worst case theoretical complexity of these TILOS-like sizers is $O(|V||E|)$, as iteratively the gate with the maximum delay reduction is picked, then static timing analysis must be updated over the timing arc edges from gate to gate, and the number of size increases for a gate is limited. For example, a gate may be only upsized as many times as the number of cell sizes (minus one) in the library for that type of logic gate. Practically speaking, $|E|$ is $O(|V|)$ as gates with more than four inputs are very slow in today's technologies due to the longer series chain of transistors from the voltage rail to output. For our benchmarks, $|E|$ ranged from $1.59\times$ to $2.01\times|V|$. As such, we expect worst case runtime behavior to be $O(|V|^2)$. See discussion in Section 4.7.2 for power minimization runtimes with Design Compiler, which uses a TILOS-like optimizer.

The greedy approach of optimizing the gate with the greatest sensitivity is a *peephole* optimization approach. The optimizer considers only changing one gate at a time and only looks at the impact on the immediate neighborhood. For example, reducing a gate's delay will provide some timing slack there, but the primary output delay on that path may not be reduced as there may be other convergent paths that are timing critical. An approach with a *global* view is needed to consider multiple gates simultaneously and determine the overall impact of them changing.

4.3 Initial work on a linear programming formulation

The delay constraints and the impact of cells changing can be encoded in a linear program. This provides a global view of the circuit for optimization.

The idea for the linear programming approach came from the zero-slack algorithm, which determines a maximal safe assignment of additional delays to each node that avoids violating the delay constraints at the outputs, but if any further delay was added to a node then a delay constraint would be violated [151]. No timing slack remains in the resulting circuit, hence the name “zero-slack”. The essential idea is formulating a set of delay constraints that determine how delays along a path add up and what additional delays can be added at each node without violating output constraints. On each iteration, their algorithm distributes slack evenly to each node on the path with the gate with minimum slack. The additional slack assigned to each node determined the maximum additional wiring capacitance, in order to successfully place and route the circuit with simulated annealing [151]. In this constraint formulation, delay constraints are linear, and signal slew is ignored.

For our purposes, we want to minimize the power by changing gate cells, which can be formulated as the sum of assigned slack multiplied by the power_reduction/delay_increase,

$$\text{minimize} \quad \sum_{v \in V} \text{AssignedSlack}_v \frac{\Delta P_v}{\Delta d_v} \quad (4.7)$$

where V is the set of gates, ΔP_v is the change in power of gate v if its cell is changed, $\Delta d_v > 0$ is the corresponding change in delay, and the slack assigned to gate v is constrained in the range $0 \leq \text{AssignedSlack}_v \leq \Delta d_v$. In this formulation, the AssignedSlack_v variables are the “free” variables to be solved for by the optimizer. We weight the assigned slacks in the objective function and add the delay constraints. This differs from the zero-slack approach in [151], in that the weighted sum of slacks is minimized, and not all slack in the circuit is set to zero after the

additional delay assignments. In particular, there is no point having $AssignedSlack_v > \Delta d_v$. Assuming $\Delta P_v / \Delta d_v$ is a fixed, pre-determined value for each gate, both the objective and the constraints are linear, so this is a linear program.

The next subsection formally presents the linear programming formulation that we initially used to minimize power [154]. The inaccuracies in the delay and power models are detailed. The linear program gives a fast and simultaneous analysis of how changing each gate affects the gates it has a path to. From the LP solution, all the gates may be sized simultaneously, which avoids greedily sizing individual gates.

4.3.1 The initial version of the linear program

The combinational circuit is represented as a directed acyclic graph $G(V,E)$, where each gate is represented by a vertex in V , and edges in E between vertices represent wires. Assuming no gate drives more than one input on another gate, we can uniquely represent a directed edge from gates u to gate v , as uv ; u is a fanin of v , and v is a fanout of u .

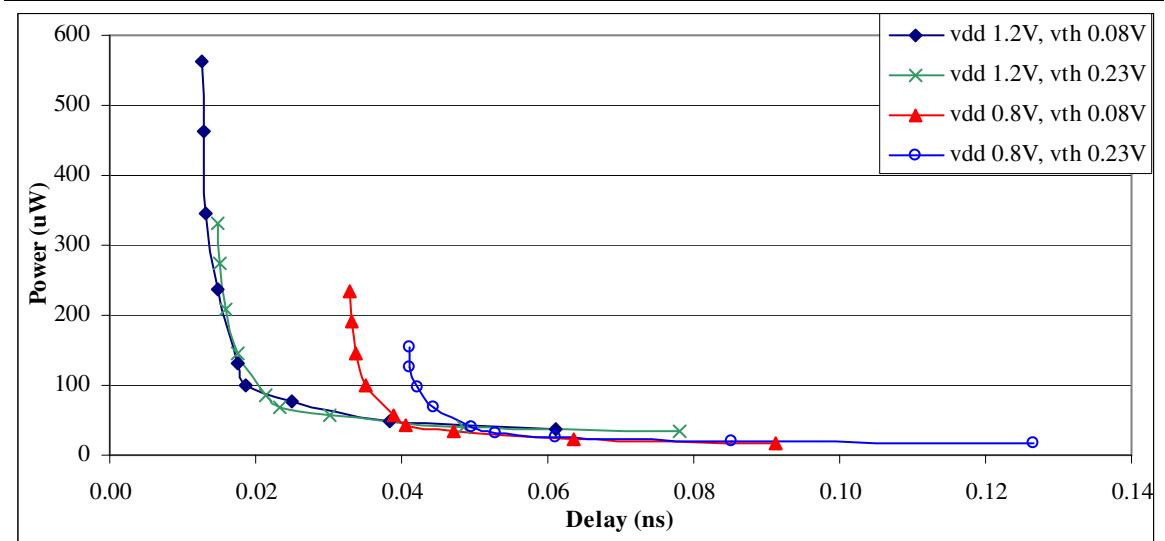


Figure 4.3 Power versus delay for inverter cells from the 0.13um PowerArc characterized libraries. The input signal was a ramp with 0.1ns slew. Each point on a curve is for a different gate size, and larger gate sizes have larger power consumption. The load capacitance was 8fF, and the switching frequency was 4GHz.

For each gate v , we determine the best alternate library cell that implements the same logic, by examining the following sensitivity metric:

$$\text{Sensitivity metric for changing cell} = \frac{\Delta P_v}{\Delta d_v}, \text{ where } \Delta P_v < 0, \Delta d_v > 0 \quad (4.8)$$

The cell alternative with the *minimum* value for this metric is the best alternative, giving the largest power_reduction/delay_increase. For example, Figure 4.3 shows alternative cell choices for an inverter with fixed values for load, input slew, and activity.

If the only cell alternates have $\Delta P_v > 0$ then the gate will not be changed. Alternatives with $\Delta d_v < 0$ are possible, but note that providing the gate delay is allowed to remain the same, these alternatives can be considered without assigning any timing slack to the gate. d_v and P_v may be determined from the input slew to a gate and from the load capacitance that the gate's output drives.

For each gate, the best cell alternative, if one exists that reduces power, is encoded in the linear program using a *cell choice* variable $\gamma_v \in [0,1]$, which determines if it is changed to a functionally equivalent cell. This terminology has been changed for clarity, but it is equivalent to the terminology in [154]. $\gamma = 1$ if the alternate cell is used, $\gamma = 0$ if not, but the LP solution may give a γ value in-between 0 and 1. The objective function for the linear program to minimize the total power is

$$\text{minimize} \quad \sum_{v \in V} \gamma_v \Delta P_v \quad (4.9)$$

and there are constraints on the cell choice variables,

$$0 \leq \gamma_v \leq 1, \text{ for all } v \in V \quad (4.10)$$

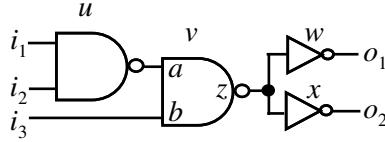


Figure 4.4 A combinational circuit for illustrating the delay constraints. Primary inputs are denoted i_1 to i_3 , and the primary outputs are o_1 and o_2 .

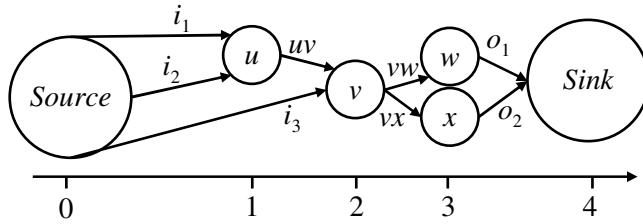


Figure 4.5 Directed acyclic graph (DAG) representation of the circuit in Figure 4.4. Primary inputs connect to the source, and primary outputs connect to the sink. The topological level is determined in a breadth first manner from the source which is level 0. Vertex names are noted inside the circle representing the node, and edge names are next to the edge.

These cell choice variables are the only “free variables” in this LP formulation; they determine the delay constraint variables, t_{vw} , on each vw between gates. Given γ from the LP solution, the cell that best minimizes gate v ’s power with delay less than $d_v + \gamma\Delta d_v$ is chosen and cells with $\Delta d_v < 0$ are considered here.

The delay constraints on edges between gates are

$$t_{vw} \geq t_{uv} + d_v + \gamma_v \Delta d_v, \text{ for all } v \in V, w \in \text{fanout}(v), u \in \text{fanin}(v) \quad (4.11)$$

Namely, the arrival time at the output of gate v (t_{vw} , on edge vw) is equal to the arrival time at the input of gate v on edge uv (t_{uv}), plus the delay of gate v , plus the change in delay Δd_v of gate v if its cell is changed. Example vertices u , v , and w are illustrated in Figure 4.4.

For simplicity, we assume that all circuit paths are subject to the same maximum delay constraint, T_{\max} , noting that it is straightforward to encode different delay constraints in the linear program if so desired. For example, where a combinational output goes to the cache and is required to arrive much earlier than an output going to a D-type flip-flop register. A circuit sink node is added to V' ,

such that all primary outputs of the combinational circuit connect to the sink and are subject to the constraint T_{\max} . Delay constraints to the circuit sink are

$$t_{wSink} \leq T_{\max}, \text{ for all } w \in \text{fanin}(Sink) \quad (4.12)$$

Similarly, we add a circuit source to V' , such that all primary inputs connect to the source, $V' = V \cup \{\text{Source}, \text{Sink}\}$. We assume that arrival times from the circuit source are at $t = 0$, though it is trivial to specify individual arrival times by input if so desired, giving

$$t_{Source u} = 0, \text{ for all } u \in \text{fanout}(\text{Source}) \quad (4.13)$$

where $t_{Source u}$ is the arrival time from the source to gate u . As there may be more than one connection from a gate to the circuit source or sink, to uniquely identify those edges we can use the primary input or primary output name. Figure 4.5 shows the directed graph $G(V', E')$ representation of the circuit in Figure 4.4.

The complete linear program to minimize power, subject to delay constraints, is

$$\begin{aligned} & \text{minimize} \quad \sum_{v \in V} \gamma_v \Delta P_v \\ & \text{subject to} \quad t_{vw} \geq t_{uv} + d_v + \gamma_v \Delta d_v, \text{ for all } v \in V, w \in \text{fanout}(v), u \in \text{fanin}(v) \\ & \quad t_{Source u} = 0, \text{ for all } u \in \text{fanout}(\text{Source}) \\ & \quad t_{wSink} \leq T_{\max}, \text{ for all } w \in \text{fanin}(Sink) \\ & \quad 0 \leq \gamma_v \leq 1, \text{ for all } v \in V \end{aligned} \quad (4.14)$$

The signal slew is not included in this formulation, nor are rise and fall delays considered separately. This leads to significant delay inaccuracy as described in the next subsection. The signal slew is the transition time of a signal from 0 to 1 or 1 to 0, and it is used as an index in the lookup tables for standard cells' delay and power.

4.3.2 Delay and power inaccuracy

The delay models in [154] were inaccurate due to ignoring slew and wire loads, and only considering the worst timing arc through a gate to determine d_v and Δd_v . As the fanin delay impact due to changed input capacitance C_{in} was not modeled, gates could not be upsized to avoid increasing fanin delay. There was no method for reducing delay, so [154] required starting at minimum delay. Also, the leakage in [154] is overstated for 0.18um, 59% of total power, which exaggerates the power reduction with multiple threshold voltages. It is straightforward to include wire loads when determining the load capacitance to determine d_v and P_v . With the help of Fujio Ishihara and Farhana Sheikh, we tested this approach on a 17,000 gate inverse discrete cosine transform block to implement dual supply voltages with a 0.13um library – the simplistic models resulted in a 24% increase in the clock period when measured in Synopsys Design Compiler, despite a tight delay constraint.

Mani, Devgan and Orshansky extended the approach in [154] to a second order conic program (SOCP) for power minimization in a statistical context [140]. The only improvement in the deterministic context to delay accuracy in their implementation was to use the average of worst case rise delay and worst case fall delay to determine d_v and Δd_v , though additional analysis was included to statistically account for process variation. Using $(\max\{d_{v,rise}\} + \max\{d_{v,fall}\})/2$ in place of $\max\{d_{v,rise}, d_{v,fall}\}$ will improve accuracy somewhat. However, this still results in poor static timing analysis accuracy if rise and fall delays are unbalanced, or if timing arcs through a gate from different inputs to outputs have significantly different delays. In particular, low power standard cell libraries typically may have a PMOS to NMOS width ratio as low as 1.5:1, instead of a ratio of about 2:1 to 1.8:1 which is required for roughly equal drive strengths in today's technologies, so the PMOS drive strength and rise times are slower. A lower width ratio reduces the circuit capacitance and switching power, and the lower capacitance may also increase the circuit speed. Secondly, they also assumed that “granularity of the Vth allocation is at the

NMOS/PMOS stack level” [140], which will result in unbalanced rise and fall delays for gates with one of the PMOS or NMOS stacks at low V_{th} and the other at high V_{th}. Thus, it remains important to model rise and fall timing arcs separately.

The approach in [154] does not handle multi-Vdd assignment in the linear program. Instead integer linear programming was used for multi-Vdd assignment, which scales poorly with circuit size. To minimize power, [154] prioritized reducing Vdd, over gate downsizing and increasing V_{th}. Similarly, the two approaches to multi-Vdd and multi-V_{th} assignment with gate sizing by Srivastava and Kulkarni prioritized Vdd reduction [124] [188].

Our experiments using the approach in [154] showed no benefit for prioritizing Vdd reduction versus a formulation with simultaneous choice between increasing V_{th}, reducing Vdd, or gate downsizing. These results are not detailed in this thesis due to the inaccuracy of the formulation in [154], but the newer work enabled 10% better power reduction on average when compared directly to Srivastava and Kulkarni’s results (see Section 7.5).

4.4 A power and delay accurate linear programming formulation

Static timing and power analysis are performed using the standard cell libraries to determine gate delay, slew and power values (e.g. d_{uv} , Δd_{uv} , ΔP_v) used in the linear program. There are a number of commercial software tools that perform static timing and power analysis, for example Synopsys Design Compiler [201] and software from Library Technologies [138]. We initially considered interfacing with a commercial tool for timing and power analysis. However, while these commercial tools internally use incremental analysis to minimize computational overheads by only computing the impact of netlist changes in regions necessary to return accurate information requested by the user, these internals are not exposed to the user. Considering alternate gate changes is a core part of the inner optimization loop for setting up the linear program, so it is essential that it be fast. In particular, when analyzing changing the gate for a cell,

only a very limited range of interactions must be considered to minimize computational overheads and we cannot specify to a commercial tool the appropriate local neighborhood to limit analysis to. It is also important to be able to roll back this change and consider alternatives for another gate, without any additional overheads to recompute the original timing and power values.

Thus using a commercial tool to perform analysis would have substantially increased runtimes. Instead, software modules were written to perform incremental timing and power analysis, with the ability to store temporary alternative values when considering cell changes and to store the best alternatives found. Design Compiler was used as a reference to debug the software for timing and power analysis, and later validate the results for optimized netlists. Appendix E provides details of how static timing and power analysis are performed to ensure delay and power accuracy versus Design Compiler.

The delay and power numbers from incremental analysis are used to specify the delay and power changes that are encoded in the linear program. However, the linear program remains somewhat limited in accuracy, because only first order changes, one gate's cell changing at a time, can be encoded and thus it is a linear approximation. Static timing and power analysis are performed after the linear program has been solved and cells have been changed, to more accurately determine the power of the new circuit and whether delay constraints have been met.

This section discusses how the linear program is accurately formulated using the data from incremental static timing and power analysis. The linear program is posed in a similar manner to [154], but each timing arc is modeled. The impact of slew on delay of fanouts and the impact of C_{in} changing on fanin delays are modeled. Wire loads are included when calculating load capacitance to determine d_v and P_v . This increases the accuracy of delay constraints. The LP formulation allows $\Delta d_v < 0$ and upsizing gates, which enables using this approach to reduce delay – for example to allow some gates to be upsized to give slack to downsize other higher power gates.

4.4.1 What variables must be modeled in the linear program for accuracy?

As outlined earlier, a central optimization issue is the accuracy of the delay and power models.

The linear program constraints must model the impact on delay and power due to changing gate size, Vdd, or Vth.

Consider changing a single gate's cell. The input capacitance C_{in} of the gate's input pins loads the fanin gates, affecting their delay and switching power. The gate's drive strength affects its delay and output slew, which may increase the delay of paths the gate is on, and the internal power of fanouts may be affected by the change in output slew. If the gate's voltage changes, that affects the switching power for the load it drives. The gate's subthreshold leakage increases exponentially with decreasing Vth. The gate size primarily determines C_{in} , which affects the load on fanins. Size (transistor width), Vdd, and Vth all affect the gate drive strength.

The impact of changing a gate on its power and delay and on that of neighboring gates was examined. Incremental timing analysis allowed exploring what neighborhood of affected gates needs to be considered for accurate analysis. The fanin level of logic must be considered, as there can be substantial delay and slew changes due to changing the load capacitance by changing the input capacitance C_{in} of a gate. Analysis was also performed with one or two fanout logic levels, from both the gate that is changed and its fanins.

More than 95% of the change in power occurs at the gate whose cell changes: switching power due to C_{in} ; switching power of the load with Vdd; leakage power; and internal power. Slew changes affect the short circuit power of neighboring gates, but short circuit power is only a small part of the total power – typically less than 10% [32]. Usually about 99% or more of the total power impact is accounted for at the gate, though in some cases it may be as low as 95%. Thus to determine with reasonable accuracy the change in power by changing a gate, we only need to consider the gate itself and can avoid computing the impact on other gates. We implicitly

consider the influence of input slew versus a gate's output slew on short circuit power in the linear programming approach, as internal power is included in the analysis.

In contrast, changing C_{in} and output slew significantly impacts the delay of neighboring gates. The impact of C_{in} is limited to the immediate fanins – the fanins of fanin gates do not see the change in load capacitance, as complementary static CMOS logic decouples this. However, the delay and slew changes of fanins and of the gate itself propagate forwards topologically. It is computationally expensive to calculate this impact over more than the fanin level of logic. Instead, we conservatively determine the worst case impact on the transitive fanout from the gate being changed and its fanins. This was sufficient to produce good power minimization results with fast computation time – though the impact of slew propagation to the fanouts must be considered as described in Section 4.4.2. Delay propagation is handled in the usual way for static timing analysis, with output arrival time constraints in terms of the arrival time at the inputs and delay of the gate.

To determine the impact of a gate x 's input capacitance changing on a fanin v , we assume the cell of the fanin gate has not changed and calculate the change in delay due to x changing, adding this to the delay constraint. For example, for the constraint on the arrival time of the rising output of gate v on edge vw is

$$t_{vw,rise} \geq t_{uv,fall} + d_{uv,rise} + \gamma_v \Delta d_{uv,rise,v} + \sum_{x \in fanout(v), x \neq w} \gamma_x \Delta d_{uv,rise,x} \quad (4.15)$$

where $t_{uv,fall}$ is the falling arrival time at v from gate u ; $d_{uv,rise}$ is the delay from the signal on uv to the output of v rising; and $\Delta d_{uv,rise,x}$ is the change in delay out of this timing arc if the cell of x changes. Here we have assumed gate v is of negative polarity, namely that a falling input may cause a rising output transition. Wire RC delays, for example on the edge from gate v to gate w ,

are also included in the delay calculations. The next subsection addresses the delay impact of signal slew.

4.4.2 Modeling the impact of signal slew propagation

For cells in our 0.13um libraries, $\Delta s_{out}/\Delta s_{in}$ ranges from -0.23 to 0.67, where Δs_{out} is the change in output slew due to change in input slew Δs_{in} . $\Delta d/\Delta s_{in}$ sensitivity ranges from -0.32 to 0.54, where Δd is the delay change due to Δs_{in} . Thus a slew change can significantly impact delay. Larger magnitude $\Delta d/\Delta s_{in}$ values occur with larger NOR and NAND drive strengths, and when the input voltage swing exceeds the gate supply voltage, for example -0.29 in the case of Vin=1.2V and Vdd=0.6V.

A simple approach was proposed in [162] to analyze the transitive fanout delay impact of slew on a given path. Their delay models were linear versus slew, and they did not consider rise and fall delay separately. So it is straightforward just to sum the delay impact along the path, giving, in our terminology, the transitive fanout delay/slew sensitivity β as

$$\beta_{uv} = \frac{\Delta d_z}{\Delta s_a} + \beta_{vw} \frac{\Delta s_z}{\Delta s_a} \quad (4.16)$$

where a and z are respectively the input and output pins of gate v shown in Figure 4.4, which has output delay d_z and output slew s_z due to a signal on a with input slew s_a ; $\Delta d_z/\Delta s_a$ and $\Delta s_z/\Delta s_a$ are the gradients for gate delay and output slew from the linear models; and the path goes through the connected gates u , v and w in the order $u \rightarrow v \rightarrow w$. A slew change Δs at the output of gate u increases the path delay by $\beta_{uv}\Delta s$. Our approach to calculate the transitive fanout delay impact of slew in Equation (4.17) below considers multiple paths simultaneously and is more accurate, providing upper and lower bounds on the slew impact.

To consider simultaneous changes, we account for gates in the transitive fanout changing by determining the worst case slew impact over all the alternate cells available for a gate. In reverse

topological order, we then calculate the maximum and minimum transitive fanout delay/slew sensitivity β :

$$\begin{aligned}\beta_{uv,\max} &= \max_{s_a, C_{load}} \left\{ \frac{\Delta d_z}{\Delta s_a} \right\} + \max_{w \in \text{fanout}(v)} \left\{ \beta_{vw,\max} \max_{s_a, C_{load}} \left\{ \frac{\Delta s_z}{\Delta s_a} \right\} \right\} \\ \beta_{uv,\min} &= \min_{s_a, C_{load}} \left\{ \frac{\Delta d_z}{\Delta s_a} \right\} + \min_{w \in \text{fanout}(v)} \left\{ \beta_{vw,\min} \min_{s_a, C_{load}} \left\{ \frac{\Delta s_z}{\Delta s_a} \right\} \right\}\end{aligned}\quad (4.17)$$

To conservatively bound the slew impact on delay, we multiply by $\beta_{uv,\min}$ if the output slew of u decreases, and by $\beta_{uv,\max}$ if it increases. We do model multiple outputs and separate rise/fall delays, but omit these in Equation (4.17) for clarity.

A change in slew propagating may reduce the delay if the lower bound $\beta_{uv,\min} > 0$ and the change in input slew is $\Delta s < 0$, as $\beta_{uv,\min} \Delta s < 0$. However, as the output slew is the maximum over the timing arcs, this decrease in delay may not propagate if the slew on this arc is not the maximum slew, even if this arc is on the critical path in terms of delay. Consequently, $\beta_{uv,\min} > 0$ may be optimistic. In practice, this does not appear to be a significant issue in most cases, but it might explain why delay reduction can perform poorly with more aggressive slew analysis, where $\beta_{uv,\min}$ is typically around 0.1.

Several approaches may be used when calculating β . Firstly, to be conservative and reduce computation time, β may be calculated over all the alternate cells for a gate, not just the current cell. Alternately, β may be calculated only for the current cell, which avoids re-computation after the best cell is chosen for each gate. This is specified by the `doAlternates` optimization parameter described in Appendix G.1. β could also be calculated over the current cell and the best alternate cell. This last option was not tried, because it requires additional computation in the inner optimization loop that sets up the linear program.

Secondly, we can conservatively calculate β over all alternate possible input slew and load conditions, or about the current input slew and load conditions only – optimistically assuming that they will not change substantially. This is specified by the `doCurrentConditionsOnly` parameter described in Appendix G.1.

In practice, it is not clear what the best approach to calculate β is. The results achieved with these different options typically vary within about 2% of the best solution, and no single approach is always best. Starting with a more conservative approach, calculating β over all alternate cells for a gate and over all possible load and input slew conditions, and then trying more aggressive settings, calculating β over only the cell for a gate and only the current load and input slew conditions, produces better results on average than starting with more aggressive settings. Such is the nature of heuristics – no approach can guarantee finding the best solutions, and different heuristics may perform better on different problems. With the conservative settings, typical values are 0.0 for β_{\min} and 0.3 for β_{\max} . With the aggressive settings, typical values are 0.1 for β_{\min} and 0.2 for β_{\max} . Results for these different slew analysis approaches are discussed in more detail in Appendix G.5.

Adding the additional slew terms to Equation (4.15), we have

$$t_{vw,rise} \geq t_{uv,fall} + d_{uv,rise} + \gamma_v (\Delta d_{uv,rise,v} + \beta_{vw,rise} \Delta s_{uv,rise,v}) + \sum_{x \in fanout(v), x \neq w} \gamma_x (\Delta d_{uv,rise,x} + \beta_{vw,rise} \Delta s_{uv,rise,x}) \quad (4.18)$$

where $\Delta s_{uv,rise,x}$ is the change in slew out of this timing arc if the cell of x changes. Multiplying by β_{vw} gives the worst case transitive slew impact on delay, where we use $\beta_{vw,\min}$ if $\Delta s_{uv} < 0$, or $\beta_{vw,\max}$ if $\Delta s_{uv} > 0$. The change in delay and slew of v due to the cell of w changing is included in the delay and slew changes for w , which is why we don't have a $\gamma_w (\Delta d_{uv,rise,w} + \beta_{vw} \Delta s_{uv,rise,w})$ term for $t_{vw,rise}$.

Comparing Equation (4.18) to Equation (4.11), there are additional terms here that consider the impact of slew, the impact of the cell of output gates x changing, and rise and fall timing arcs are considered separately.

We examined the importance of the β terms, which encapsulate the transitive fanout delay impact of slew, by setting $\beta = 0$ and performing optimization. With V_{th} of 0.23V, for most of the benchmarks the inaccuracy due to ignoring slew results in a violation of the delay constraints, and the delay reduction phase of the optimization flow fails to satisfy the delay constraints due to ignoring slew. The addition of a lower threshold voltage adds timing slack, which makes it possible to satisfy delay constraints, but the results were 6.2% higher power on average due to ignoring slew. This indicates how important it is to account for slew both in static timing analysis and in the actual optimization formulation. See Appendix G.2 for details.

4.4.3 Formulating cell changes

Now that we have identified what is necessary for delay and power accuracy, we can again address identifying the best alternative cells for a gate. As in Section 4.3, we use a *cell choice* variable $\gamma \in [0,1]$ to indicate whether the alternative is used ($\gamma = 1$ if it is, $\gamma = 0$ if not). However, instead of computing our sensitivity metric via Equation (4.8), $\Delta P_v/\Delta d_v$, we must consider multiple timing arcs. To allow the linear programming approach to be used for delay reduction, we must allow $\Delta d_v < 0$ to be encoded in the LP. As discussed in Section 4.4.1, ΔP_v is determined by summing over the change in leakage power, change in internal power, change in switching power of the gate's inputs, and change in switching power of the gate's outputs if Vdd changes – essentially the same as ΔP_v in the initial approach described in Section 4.3.

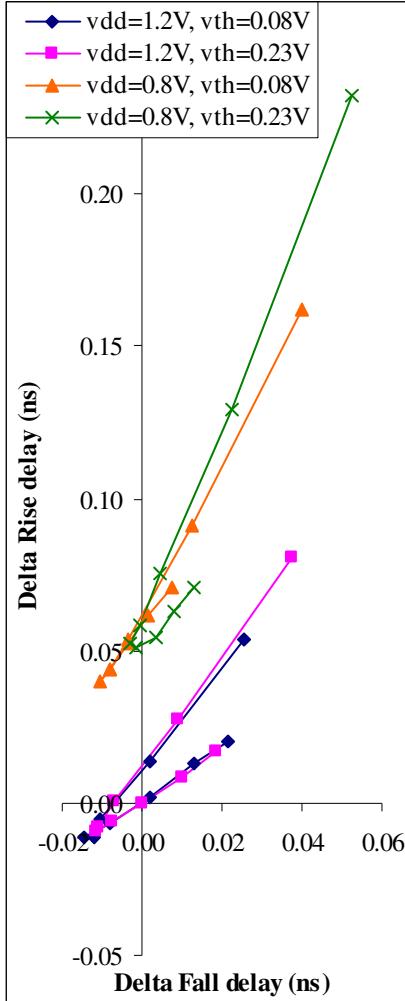


Figure 4.6 This graph shows that the change in rise delay and fall delay are unequal. It shows the change in rise delay versus the change in fall delay for an inverter gate in ISCAS'85 benchmark c5315 with different cell alternatives. Each point on a curve is a different cell drive strength, and the different curves correspond to different supply (v_{dd}) and threshold voltages (v_{th}). If the rise and fall delays were balanced, we would see a line of the form $\text{rise_delay} = \text{fall_delay}$, but in practice rise delay grows faster than fall delay as PMOS drive strength is less. The current cell for the gate is at the origin of the graph. There is a jump in rise delay going from $V_{dd}=1.2V$ to $V_{dd}=0.8V$ due to the additional delay from a voltage level converter.

When considering multiple timing arcs and Δd_v , there are two options that were considered. We could just use the worst change in delay and additionally the transitive delay impact of slew, or we could combine multiple timing arcs into the metric. The latter approach was performed by averaging the delay and transitive delay impact of slew over the timing arcs. Averaging the timing arcs is equivalent to the worst case if pull-up and pull-down drive strengths are balanced and individual timing arcs have similar delays. However, in practice this is not the case, as shown in Figure 4.6. Generally, the more conservative approach using the worst case delay change on a

timing arc to determine Δd_v produces slightly better results, because this is less likely to result in a delay change that violates the delay constraint. The two approaches produce results on average within 1%, though in some cases one may be 4% better than the other. A third possible approach, which has not been tried, would be to weight by slack on each timing arc. Note that when Δd_v is calculated, it includes the impact of the cell changing on the delay and the slew of the fanins of v on timing arcs that propagate to v .

The best alternative cell for a gate is chosen as follows. If a cell change reduces power and delay ($\Delta P < 0$, $\Delta d < 0$), pick the cell which best reduces the objective, delay or power. Otherwise: to reduce power pick the cell with maximum power_reduction/delay_increase (min $\Delta P/\Delta d$, $\Delta P < 0$, $\Delta d > 0$); to reduce delay pick the cell with the max delay_decrease/power_increase (max $\Delta P/\Delta d$, $\Delta P > 0$, $\Delta d < 0$). Here, Δd is the maximum delay change over its timing arcs, including the slew impact, or the average change over the timing arcs as discussed above. Note that for different gates we may encode cells that reduce delay or reduce power in the same linear program. For example when minimizing power, if a gate is already minimum size, then there may be no cell change that reduces power further, but if there size increase that reduces delay at the expense of power, we encode that in the LP to allow for the situation where other gates can better use the slack that would be created by upsizing the gate – the LP determines whether this is a worthwhile trade-off or not. These steps are shown later in Figure 4.8.

The best delay and power cell alternatives for a gate are cached by its input slews, input arrivals and load capacitance values, and that of its fanins along with their supply voltages. For caching, the delay and slew accuracy is 0.0001ns, voltage accuracy is 0.001V, and capacitance accuracy is to 1fF – less accurate caching can be used to reduce computation time as matches will be found more often. Caching provides substantial speed ups for setting up the LP on later iterations.

The initial approach in Section 4.3 used the $\gamma \in [0,1]$ values from the LP solution, to change a gate's cell to the cell with least power such that the gate's new delay is less than $d_v + \gamma\Delta d_v$. However, we now have to consider multiple timing arcs. Do we require that the new cell have delay less than $d_{v,arc} + \gamma_{v,arc}\Delta d_{v,arc}$ on each timing arc? What if it increases the delay on one arc, and reduces the delay on another arc? It was not clear how to resolve this issue, or indeed whether it was worth the additional computation time to compute the changes for all the different cell alternatives again. The LP solver runtime is small on the smaller benchmarks, so computing cell alternatives again would double the runtime on smaller benchmarks. Instead, a threshold approach was tried: if a cell reduced delay and $\gamma > 0.01$, the alternate cell was used; if a cell reduced power and $\gamma > 0.99$, the alternate cell was used. A number of different thresholds were tried, these produced the best results. The philosophy behind using a threshold of 0.01 for delay is that if that gate's delay needs to be reduced to meet delay constraints, then the alternate cell must be used. Conversely, a high threshold was set for power reduction, because if γ wasn't close to 1, we couldn't guarantee that delay constraints would be met if the cell was changed. In practice, this threshold approach produces very good power minimization results as described in Section 4.6.2.

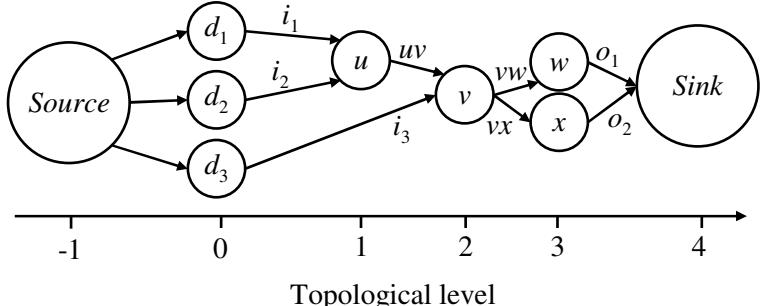


Figure 4.7 This diagram shows the inclusion of input drivers (d_1, d_2, d_3) in the graph representation (Figure 4.5), to more accurately model the impact of gates loading the circuit elements that drive the primary inputs. Input drivers may be set to any appropriate library cell – a drive strength X1 inverter is typical.

4.4.4 Input drivers

An additional accuracy improvement is modeling the impact of gates loading the primary inputs, by including input drivers in the circuit representation (Figure 4.7). The cell for the input drivers is user specified, and might be set to a drive strength X1 inverter for example. The switching power of the driver input pins is not included, nor is the internal power of the input drivers. The set of drivers is denoted D . The drivers are added to the extended graph $G(V'', E')$, where $V'' = V \cup D$. As the cell for a driver v cannot change, $\gamma_v = 0$, the delay constraint on an input driver is

$$\begin{aligned} t_{vw,rise} &\geq d_{Source\ v,rise} + \sum_{x \in fanout(v), x \neq w} \gamma_x (\Delta d_{uv,rise,x} + \beta_{vw} \Delta s_{uv,rise,x}), w \in fanout(v) \\ t_{vw,fall} &\geq d_{Source\ v,fall} + \sum_{x \in fanout(v), x \neq w} \gamma_x (\Delta d_{uv,fall,x} + \beta_{vw} \Delta s_{uv,fall,x}), w \in fanout(v) \end{aligned} \quad (4.19)$$

where $d_{Source\ v,rise}$ and $d_{Source\ v,fall}$ are respectively the rising and falling delay of the input driver.

4.4.5 The linear program

To minimize the total power, the complete formulation for the linear program formulation is

$$\begin{aligned}
 & \text{minimize} && \sum_{v \in V} \gamma_v \Delta P_v \\
 & \text{subject to} && t_{\text{Source } u, \text{fall}} = 0, \text{ for all } u \in D \\
 & && t_{\text{Source } u, \text{rise}} = 0, \text{ for all } u \in D \\
 & && t_{w \text{Sink}, \text{rise}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
 & && t_{w \text{Sink}, \text{fall}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
 & && 0 \leq \gamma_v \leq 1, \text{ for all } v \in V \\
 & && \gamma_v = 0, \text{ for all } v \in D
 \end{aligned} \tag{4.20}$$

For all $v \in V \cup D$, $w \in \text{fanout}(v)$, $u \in \text{fanin}(v)$, timing arc constraints:

$$\begin{aligned}
 t_{vw, \text{rise}} &\geq t_{uv, \text{fall}} + d_{uv, \text{rise}} + \gamma_v (\Delta d_{uv, \text{rise}, v} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, v}) \\
 &\quad + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{rise}, x} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, x}) \\
 t_{vw, \text{fall}} &\geq t_{uv, \text{rise}} + d_{uv, \text{fall}} + \gamma_v (\Delta d_{uv, \text{fall}, v} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, v}) \\
 &\quad + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{fall}, x} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, x})
 \end{aligned}$$

where $t_{uv, \text{fall}}$ is the falling arrival time at v from gate u ; $d_{uv, \text{rise}}$ is the delay from the signal on uv to the output of v rising; and $\Delta d_{uv, \text{rise}, x}$ and $\Delta s_{uv, \text{rise}, x}$ are the changes in delay and slew out of this timing arc if the cell of x changes. Multiplying by β_{vw} gives the worst case transitive slew impact on delay, where we use $\beta_{vw, \min}$ if $\Delta s_{uv} < 0$, or $\beta_{vw, \max}$ if $\Delta s_{uv} > 0$. The change in delay and slew of v due to the cell of w changing is included in the delay and slew changes for w . This is why we don't have a $\gamma_w (\Delta d_{uv, \text{rise}, w} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, w})$ term for $t_{vw, \text{rise}}$ for example. The LP with linear approximations cannot model the higher order delay impact of multiple cells changing ($\gamma_v \gamma_x$ terms). Solving such higher order problems would be much slower.

The delay constraints specified in Equation (4.20) assume that the gates have negative polarity, that is a rising input causes a falling output if there is a logical transition, or a falling input causes a rising output. Constraints for positive polarity and nonunate transitions can be handled similarly. Positive polarity means that a rising (falling) output is caused by a rising (falling) input transition.

A nonunate transition can be caused by both a rising input or a falling input, depending on the value of other inputs – for example for an XOR gate. Our software handles all timing arcs, including positive polarity and nonunate gates, and gates with multiple outputs. Wire delays are included in the timing analysis. The wire load model is specified in the library, or extracted post-layout wiring parasitics could be specified for each wire.

We can also use the same LP formulation approach to reduce delay when $T > T_{\max}$. When reducing delay, the objective is

$$\text{minimize} \quad \max\{\tau T_{\max}, T\} + k \sum_{v \in V} \gamma_v \Delta P_v \quad (4.21)$$

where k is a weight to limit the power increase when reducing delay, and τ limits the delay reduction. If the ratio of total power to the critical path delay is large, then k should be small to allow delay reduction. For our benchmarks, the best values were k of 0.01 and τ of 0.99, so that after delay reduction there is timing slack for further power minimization. In several cases to meet T_{\max} , k of 0.001 and τ of 0.98 were used.

The complete formulation of the linear program for delay reduction with a weighting on power is

$$\begin{aligned}
\text{minimize} \quad & \max\{\tau T_{\max}, T\} + k \sum_{v \in V} \gamma_v \Delta P_v \\
\text{subject to} \quad & t_{\text{Source } u, \text{fall}} = 0, \text{ for all } u \in D \\
& t_{\text{Source } u, \text{rise}} = 0, \text{ for all } u \in D \\
& t_{w \text{Sink}, \text{rise}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
& t_{w \text{Sink}, \text{fall}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
& 0 \leq \gamma_v \leq 1, \text{ for all } v \in V \\
& \gamma_v = 0, \text{ for all } v \in D
\end{aligned} \tag{4.22}$$

For all $v \in V \cup D$, $w \in \text{fanout}(v)$, $u \in \text{fanin}(v)$, timing arc constraints:

$$\begin{aligned}
t_{vw, \text{rise}} &\geq t_{uv, \text{fall}} + d_{uv, \text{rise}} + \gamma_v (\Delta d_{uv, \text{rise}, v} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, v}) \\
&\quad + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{rise}, x} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, x}) \\
t_{vw, \text{fall}} &\geq t_{uv, \text{rise}} + d_{uv, \text{fall}} + \gamma_v (\Delta d_{uv, \text{fall}, v} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, v}) \\
&\quad + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{fall}, x} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, x})
\end{aligned}$$

In summary then, the major differences between formulating the linear programs in Equation (4.20) and Equation (4.22) versus the initial approach in Equation (4.14) are as follows. The delay model is more accurate:

- All the timing arcs are modeled, instead of only the worst case timing arc.
- The first order impact of slew on delay is included.
- The impact of a gate's cell changing on the delay and slew out of its fanins is added.
- $\Delta d_v < 0$ is allowed in the LP formulation.

The approach described in Section 4.3 does not model the delay impact on fanins. Thus to avoid violating delay constraints gates were not allowed to be upsized, as this would increase the fanin delay. Similarly, if a gate is downsized reducing the fanin load, the additional slack for power minimization was not considered on that linear programming iteration. Modeling fanin delay impact and allowing $\Delta d_v < 0$ enable delay minimization with the linear programming approach.

The next section describes how these delay reduction and power minimization linear programs are used iteratively to reduce the circuit power consumption while meeting delay constraints.

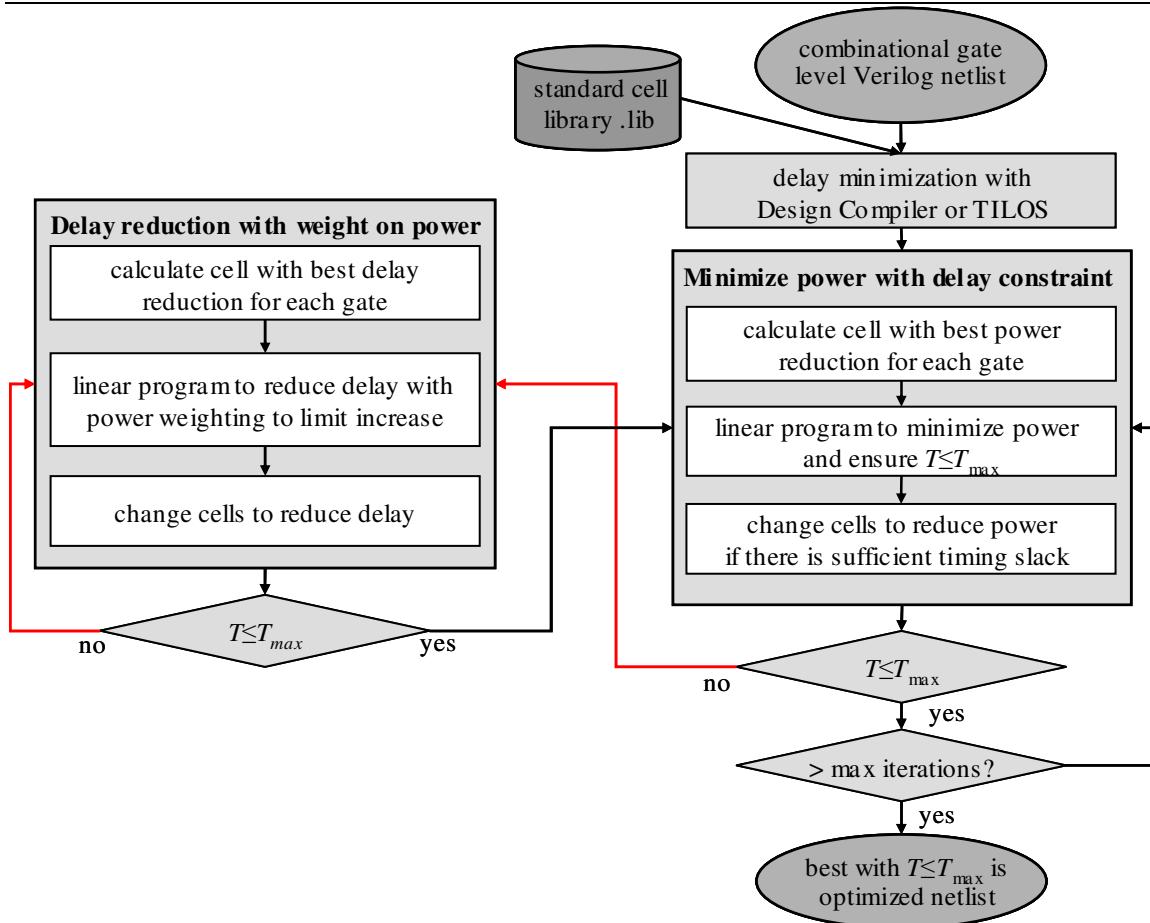


Figure 4.8 An overview of the optimization flow.

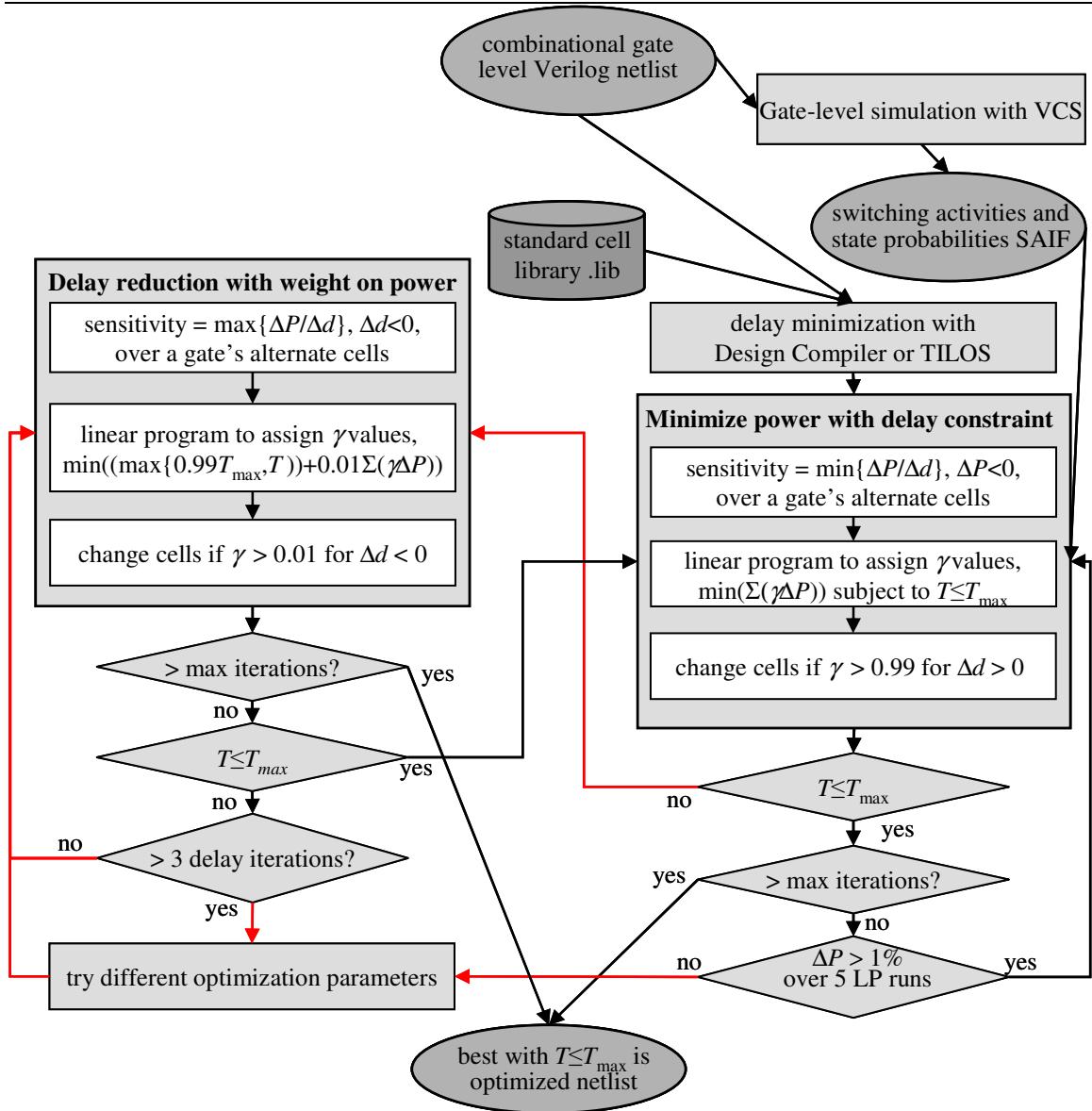


Figure 4.9 Detailed optimization flow diagram. The step “try different optimization parameters” is detailed in Appendix G.1.

4.5 Optimization Flow

A high-level overview of the optimization flow is shown in Figure 4.8, and Figure 4.9 shows further details. We start with a combinational gate-level netlist (in a Verilog file), accompanying switching activity and leakage state probabilities (in a SAIF file), and standard cell libraries (Liberty .lib format). Input drivers or input slew and output port load capacitance are user specified. Power minimization is performed subject to a delay constraint. If delay constraints are violated after optimization, delay reduction with a weighting on power is performed. The

optimization is iterated until the maximum number of iterations is reached. The initial approach to linear programming had a very similar flow to the right-hand side power minimization phase of Figure 4.8, but there was no delay minimization phase [154].

The starting point for optimization matters little providing that if a delay constraint is violated, for example after power minimization, we can reduce the delay to satisfy delay constraints. For example, multi-V_{th} experiments starting with all gates at low V_{th} rather than high V_{th} gave only marginally better results after optimization. However, at a fairly tight delay constraint, such as 1.1× the minimum critical path delay, the delay reduction phase may have trouble reducing delay, in which case it is essential to start with a delay minimized netlist (see Section 4.5.1).

Alternate cells for a logic gate are chosen by the best power/delay sensitivity as described in Section 4.4.3; we set up the linear program constraints; and then the open source COIN-OR LP solver [66] is used to choose γ values to minimize the power subject to delay constraints. Gates with γ close to 1 in the LP solution are then changed – a threshold of $\gamma > 0.99$ generally worked best. If $\gamma < 0.99$ then there is insufficient slack in the circuit for the cell to be changed without violating a delay constraint, assuming that changing the cell results in a delay increase.

Gates are changed simultaneously without fully considering the impact of other gate changes. If a gate is upsized increasing C_{in} and its fanin is downsized, then the fanin delay is larger than modeled in the LP which may lead to violating T_{max} . If this occurs, we perform delay reduction to satisfy T_{max} . For delay reduction, a threshold of $\gamma > 0.01$ worked well – if a gate has to be upsized to satisfy the delay constraint, we do so.

Thus the solution converges iteratively to reduce power and satisfy T_{max} . At a tight delay constraint, the delay reduction may fail to satisfy T_{max} . Design Compiler with greedy delay_reduction/power_increase reduces delay better than our tool. The output of our

optimization is the optimized Verilog netlist, for which the static timing and power analysis can then be verified in Design Compiler.

As the solution converges, the power reduction that is achieved per iteration decreases. To ensure that the solution is close to the optimal that can be achieved by the linear programming approach, several different parameter settings can be tried to see if any additional savings may be achieved. Code for trying different parameter settings is detailed in Appendix G.1 and results with different parameter settings are in Appendix G.5.

The next subsection describes the importance of the delay reduction phase.

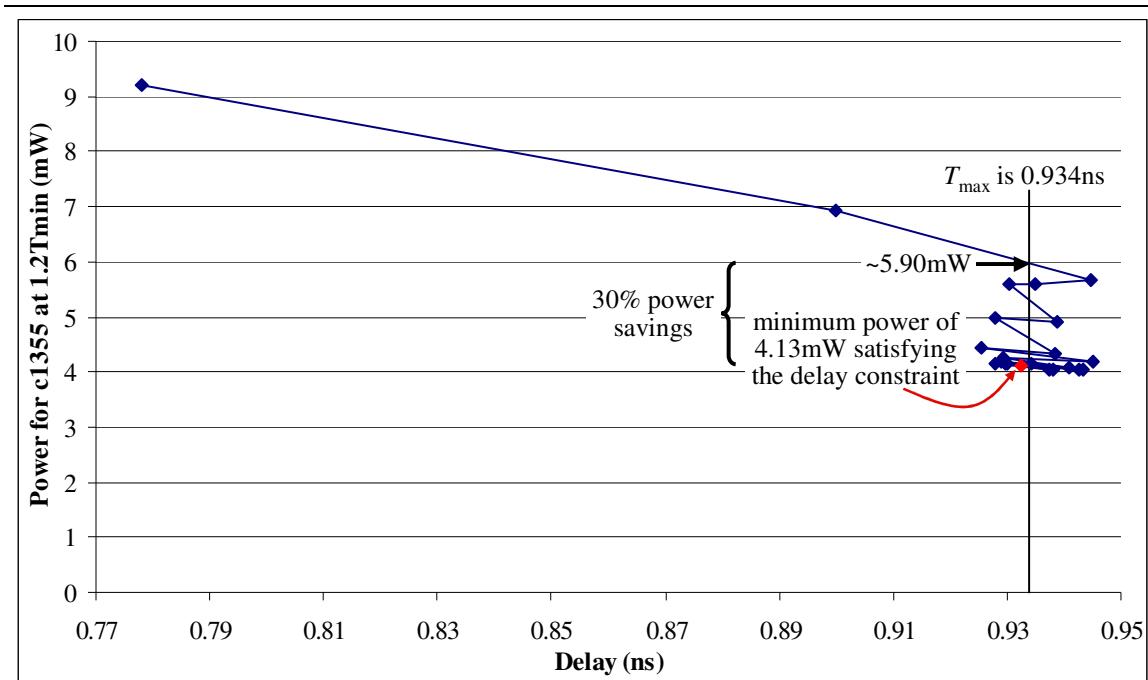


Figure 4.10 This graph shows the power and delay after each iteration of the optimization flow. This is for the c1355 benchmark at a delay constraint of $1.2T_{\min}$ for the Design Compiler delay-minimized netlist which is the starting point. The $V_{dd}=1.2V/V_{th}=0.23V$ PowerArc characterized 0.13um library was used for this gate sizing optimization.

4.5.1 The importance of the delay reduction phase

The importance of the delay reduction phase is illustrated by Figure 4.10, which shows the typical progress of the optimization flow. After a couple of iterations of power minimization, the timing slack in the initial delay-minimized circuit has been used to perform gate downsizing. If cell

changes were not allowed to cause the delay constraint to be violated, the optimization would stop at a solution with total power of about 5.9mW. Alternatively, we can allow the power minimization flow to make cell changes that may end up violating delay constraints, changing the cells of gates that have $\gamma > 0.99$ from the LP solution, without additional computational overheads to double-check that the delay constraints are not violated.

After cells have been changed and static timing analysis reports a circuit delay that violates the delay constraint T_{\max} , we then perform delay reduction, but we add a weight on the power in the objective to ensure that the power does not increase too much. This helps ensure that power minimization phase steps have steeper $\Delta P / \Delta T$ than the delay reduction steps, thus multiple iterations of power minimization with delay reduction can achieve further power savings. In the example shown in Figure 4.10, a substantial 30% power savings are achieved beyond the 5.9mW point, with a minimum power of 4.13mW at a point that meets the delay constraint.

The delay reduction phase allows hill climbing, by allowing the T_{\max} constraint to be violated by power minimization to see if some additional power savings may be achieved. The cell changes from delay reduction may be different to those performed in delay minimization of the initial netlist, thus providing some slack back into the circuit for power minimization, without reversing all the power minimization steps that resulted in violating T_{\max} . Without this hill climbing, the results would be substantially worse, and would not be as good as Design Compiler or the TILOS power minimization results. Thus the lack of a delay reduction phase is one of the major limitations of the initial approach to linear programming that was proposed in [154], and the same limitation is present in the second order conic programming statistical optimization approach used by Mani, Devgan and Orshansky in [140].

Coudert observed that “ping-ponging” back and forth from the infeasible region, where the delay constraint is violated, to the feasible region produced good results [51]. This “ping-ponging”

approach was dismissed as being too slow and applicable only to circuits of less than 500 gates. However, our LP approach that proceeds in this manner has been applied to benchmarks of up to 33,000 gates and runtime scales between linearly and quadratically, so it can be applied to significantly larger circuits.

Unfortunately, the delay reduction phase can sometimes perform poorly at a tight delay constraint. For a delay constraint of $1.1T_{\min}$, starting with TILOS-optimized netlists sized for $1.1T_{\min}$ results in 4.6% worse results on average than starting with delay minimized netlists, as detailed in Appendix G.3. Delay reduction can perform poorly for several reasons.

Firstly, the weight on power in the delay reduction objective may prevent certain cell changes that are essential to reduce delay below T_{\max} , but cause too large an increase in power. This can be solved by reducing the weighting on power, and allowing additional delay reduction. Setting $\tau = 0.98$ and $k = 0.001$ in Equation (4.21) does this, and is one of the parameter changes tried as detailed in Appendix G.1.

Secondly, the aggressive slew analysis setting may underestimate the delay increase due to a slew increase and overestimate the delay reduction due to a slew decrease. In contrast, the conservative slew analysis setting is pessimistic and often provides better delay reduction results.

Thirdly, in both phases of the optimization, all cells are changed simultaneously after the optimization. However, the linear program is a linear approximation, and there are no second order terms of the form $\gamma_u \gamma_v$ to directly account for the delay impact of say a cell being downsized while its fanout is upsized. This case can result in delay actually getting worse after delay reduction is attempted. This is more difficult to solve without additional computation overheads, as cells need to be sized individually. For example, a cell being downsized but its fanout upsized could be disallowed, though either one of these on its own would be acceptable, and analysis

along the lines of TILOS (Equation (4.3)) would be required to decide which of the two cells is more important to resize.

When delay reduction performs badly, some of the delay reduction steps actually increase delay and power, as shown in Figure 4.11. Slew analysis settings were set aggressively, which lead to the delay impact of slew being under-estimated and problems meeting the delay constraint in the delay reduction phase. This problem didn't occur with conservative slew analysis settings.

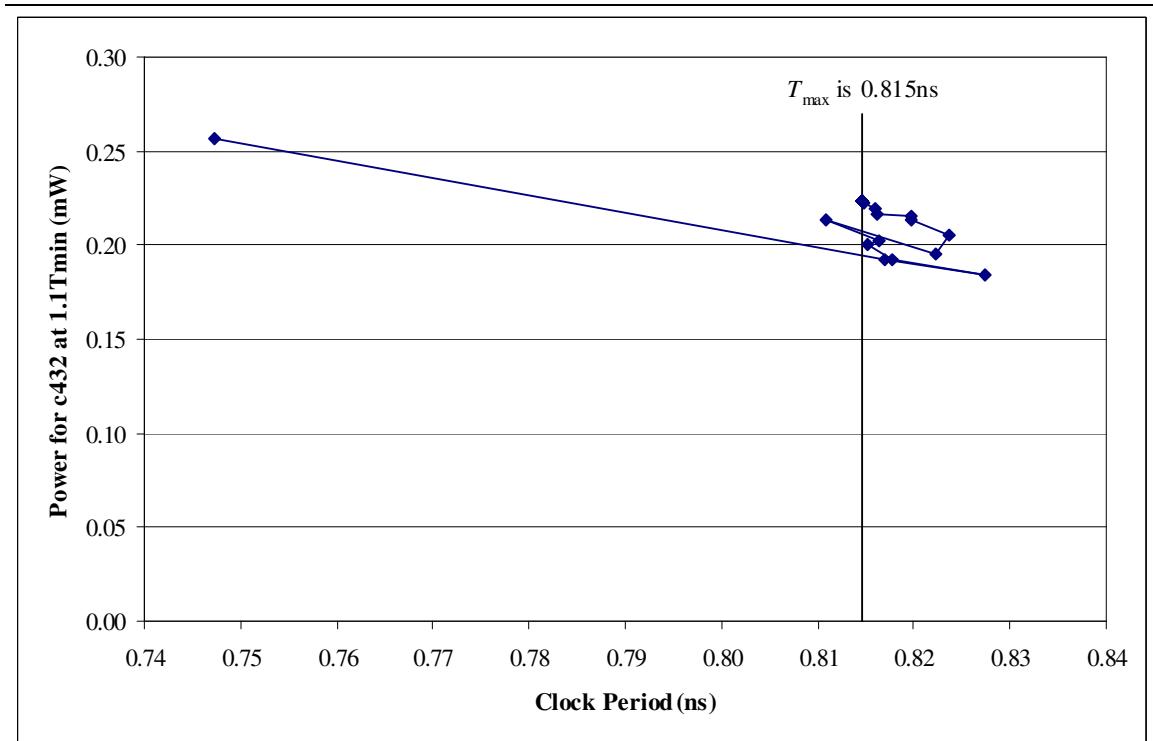


Figure 4.11 This graph shows the progress of the optimization flow in a case where delay reduction performs very poorly. Each point is the power and delay after another iteration of the optimization flow. This is for the c432 benchmark at a delay constraint of $1.1T_{\min}$ for the TILOS delay-minimized netlist which is the starting point. The $V_{dd}=1.2V/V_{th}=0.12V$ PowerArc characterized 0.13um library was used for this gate sizing optimization, and switching activity was multiplied by a fraction such that leakage is about 20% of total power. Slew analysis settings were set aggressively, which lead to the delay impact of slew being under-estimated and problems meeting the delay constraint in the delay reduction phase.

4.6 Comparison of gate sizing power minimization results

We shall compare our results versus the commercial synthesis tool Design Compiler [201], which is most commonly used in industry today and has a gate sizing approach that is based on TILOS. Design Compiler is generally considered to produce high quality results compared to other

commercially available EDA tools [72]. Section 4.6.1 discusses the combinational benchmarks on which we compare results.

The linear programming optimizer consistently saved power versus the University of Michigan TILOS-like optimizer, with average power savings of 14.4% at $1.1 \times T_{\min}$, 17.8% at $1.2 \times T_{\min}$, and 22.1% at $1.5 \times T_{\min}$. As delay constraints are relaxed and more slack is available to distribute in the circuit to reduce power, the LP optimization approach provides more benefit. The power savings for the linear programming approach versus TILOS are detailed in Appendix G.4.

While the results versus the University of Michigan TILOS-like sizer were good, the real test is versus a commercial synthesis tool. Section 4.6.2 compares our results versus Design Compiler, which performs gate sizing based on a TILOS sizing approach, but no doubt has additional tweaks to improve the results. Section 4.6.3 then discusses how optimal the linear program sizing results are, and whether any additional improvements can be made.

The next section discusses the circuit benchmarks used to compare the linear programming approach against the TILOS-like sizers.

Table 4.1 The function performed by the ISCAS'85 benchmark circuits and the number of functional blocks in each. The functional blocks are blocks in the high-level model for the circuit. For example, c880 consists of two multiplexers; a module to produce generate, propagate and sum adder signals; two 4-bit carry-look-ahead adder units; and an 8-bit XOR [83]. The number of inputs, outputs, logic gates, and logic levels in the synthesized circuits are listed in Table 4.2. The smallest benchmark, c17, wasn't in [83].

Circuit	Function	# of Functional Blocks
c17	not detailed in their paper	-
c432	27-channel interrupt controller	5
c499	32-bit single-error-correcting circuit	2
c880	8-bit ALU	7
c1355	32-bit single-error-correcting circuit	2
c1908	16-bit single-error-correcting/double-error-detecting circuit	6
c2670	12-bit ALU and controller	7
c3540	8-bit ALU	11
c5315	9-bit ALU	10
c6288	16x16 multiplier	240
c7552	32-bit adder/comparator	8

4.6.1 Benchmarks

Two sets of circuit benchmarks were used in this chapter. The first set of benchmarks was the combinational ISCAS'85 benchmark set [25]. Besides these netlists being small, one of the criticisms of them has been that they are not realistic circuit benchmarks. In particular, what circuits they represent and how to stimulate them properly with input vectors is not detailed in the benchmark set. The ISCAS'85 benchmarks were reverse-engineered by Hansen, Yalcin and Hayes [83], and the functions that they determined for these circuits are listed in Table 4.1. The behavioral Verilog netlists for these reverse-engineered netlists are available [82], and they were synthesized and delay minimized using Design Compiler with the PowerArc characterized Vdd=1.2V/Vth=0.23V 0.13um library. Assignment statements and redundant outputs were removed manually in the synthesized netlists. These gate-level synthesized netlists were simulated with VCS using independent random inputs with equal probabilities of 0 or 1 to produce SAIF (Switching Activity Interchange Format) files with switching activity and gate input state probabilities for power analysis. For comparison to our linear programming power minimization approach, the delay minimized netlists were then power minimized in Design Compiler restricted to sizing only changes. See Appendix F for example Design Compiler and VCS scripts.

The second set of three benchmark circuits was provided by Professor Nikolić's group at the Berkeley Wireless Research Center. The SOVA EPR4 circuit is an enhanced partial response class-4 (EPR4) decoder [243]. There is also a Huffman decoder [155]. These are typical datapath circuits that appear on chips for communication systems. These circuits were mapped to the PowerArc characterized $V_{dd}=1.2V$ / $V_{th}=0.12V$ 0.13um library by Sarvesh Kulkarni and Ashish Srivastava, who provided the combinational gate-level netlists, and we provided them with the corresponding SAIF files from VCS simulation. The SOVA EPR4 and R4 SOVA benchmarks are substantially larger than the ISCAS'85 benchmarks.

4.6.2 Comparison versus Design Compiler

For results in this section, the PowerArc characterized 0.13um library at 25°C with V_{dd} of 1.2V and V_{th} of 0.23V was used. The channel length was 0.13um. This was characterized for STMicroelectronics 0.13um HCMOS9D process. The library consisted of nine inverter sizes; and four sizes of NAND2, NAND3, NOR2 and NOR3 logic gates. The output port load capacitance was set to 3fF, which is reasonable if the combinational outputs drive flip-flops, and in addition there is a wire load to the port. The wire load model used was $3+2\times\text{num_fanout}$ fF. The input slew was 0.1ns for the 1.2V input drive ramps – typical slews within the circuits ranged from 0.05ns to 0.15ns. The same wire load, input slew, and load conditions were used in [124] and [188]. The switching activities used were directly from the SAIF files. Leakage was about 0.1% of total power, due to characterization at 25°C and high V_{th} . This avoids the problem with versions of Design Compiler before 2004.12 where either dynamic power or leakage power had to be prioritized, rather than total power – as leakage is so small, prioritizing dynamic power is equivalent to total power, which is minimized by the linear programming approach.

The starting point for LP optimization was the netlists that were synthesized and delay minimized using Design Compiler, and that was also the starting point for sizing_only power minimization in Design Compiler. Results were verified in Design Compiler.

The linear programming approach does better than Design Compiler in all cases, except for c880 at a delay constraint of $1.1 \times T_{\min}$ where the LP power is 2.4% higher. This is not surprising as the LP approach is heuristic and may still get stuck in a local minimum. The LP approach performs better in most cases because it has a global view, rather than the greedy peephole approach of TILOS. We achieved 12.0% and 16.6% average power savings versus Design Compiler at delay constraints of $1.1 \times T_{\min}$ and $1.2 \times T_{\min}$ respectively – see Table 4.2. We achieved lower power even on the smallest ISCAS’85 benchmark, c17, as was illustrated in Figure 4.1. This illustrates the suboptimal choices made by greedy optimization approaches that only consider individual gates, rather than the whole circuit. These results were versus the 2003.03 version of Design Compiler – version 2004.12 was also tried, but it produced worse power results on average.

Table 4.2 Here we compare our sizing results (LP) with sizing only power minimization results from Design Compiler (DC), at delay constraints of $1.1T_{\min}$ and $1.2T_{\min}$, where T_{\min} (shown in column 7) is the critical path delay after delay minimization by Design Compiler. Circuit statistics such as the number of logic levels, the numbers of inputs and outputs, the number of gates, and the number of edges between gates in the circuit are also listed. The “LP then DC” results are discussed in Section 4.6.3.

Netlist	# logic levels	# inputs	# outputs	# gates	# edges	Min Delay (ns)	Power (mW)					
							1.1T _{min}			1.2T _{min}		
							DC	LP	LP then DC	DC	LP	LP then DC
c17	4	5	2	10	17	0.094	1.11	0.96	0.95	0.86	0.76	0.76
c432	24	36	7	259	485	0.733	2.78	2.21	2.18	2.22	1.74	1.70
c499	25	41	32	644	1,067	0.701	5.83	4.59	4.48	4.98	3.73	3.64
c880	23	60	26	484	894	0.700	3.37	3.45	3.13	2.83	2.60	2.54
c1355	27	41	32	764	1,322	0.778	6.88	5.42	5.26	5.97	4.12	4.04
c1908	33	33	25	635	1,114	0.999	3.26	3.08	3.01	2.67	2.40	2.36
c2670	23	234	139	1,164	1,863	0.649	9.23	8.42	8.28	8.08	6.87	6.79
c3540	36	50	22	1,283	2,461	1.054	6.69	5.79	5.70	5.60	4.64	4.53
c5315	34	178	123	1,956	3,520	0.946	10.39	9.48	9.15	8.82	7.81	7.66
c6288	113	32	32	3,544	6,486	3.305	6.91	6.07	5.89	6.08	4.69	4.61
c7552	31	207	86	2,779	4,759	0.847	18.02	16.65	16.34	15.60	13.44	13.23
Huffman	29	79	42	774	1,286	0.845	6.02	4.81	4.62	5.07	3.72	3.61
SOVA_EPR4	110	791	730	15,686	27,347	3.039	17.07	15.82	15.61	15.28	13.89	13.73
R4_SOVA	144	1,177	815	33,344	59,178	4.811	24.26	21.81	21.22	20.82	19.16	18.69
							Minimum power savings vs. Design Compiler:			-2.4%	7.0%	
							Average power savings vs. Design Compiler:			12.0%	14.5%	
							Maximum power savings vs. Design Compiler:			21.4%	23.5%	
										30.9%	32.3%	

4.6.3 Post-pass cleanup with Design Compiler

After our linear programming optimization returns the lowest power solution that satisfies the delay constraint, there is a little timing slack left, up to about 0.6% of the delay target, which can be used to further downsize individual gates. Each LP optimization pass sizes multiple gates, whereas to fully utilize the remaining slack, individual gates should be sized. A power_reduction/delay_increase sensitivity approach such as provided by Design Compiler's sizer is appropriate for this.

On average, a post-pass with Design Compiler on the LP power minimized netlists achieved another 2% to 3% power savings versus the LP results, as listed in the “LP then DC” columns in Table 4.2. Interestingly, for c880 where the LP results were worse than Design Compiler at $1.1T_{\min}$ by 2.4%, the post-pass by Design Compiler improves the result by 9.3%, giving 7% overall power reduction with “LP then DC” versus the Design Compiler result. After running Design Compiler, there is typically at most 0.001ns slack, i.e. less than 0.1% of the delay constraint. The average power savings of the “LP then DC” results versus Design Compiler were 14.5% and 18.1% at delay constraints of $1.1T_{\min}$ and $1.2T_{\min}$ respectively.

It should be noted that multiple passes of power minimization sizing by Design Compiler on its own does not provide any significant benefit (<1%) over a single incremental power minimizing sizing compilation in Design Compiler. Design Compiler gets stuck in a local minimum where it has greedily downsized the wrong gates.

Table 4.3 Here we compare our sizing results with the linear programming approach (LP) and after a post-pass cleanup with Design Compiler (LP then DC) with sizing results from integer linear programming with CPLEX’s solver (CPLEX ILP), at delay constraints of $1.1T_{\min}$ and $1.2T_{\min}$, where T_{\min} (shown in column 2) is the critical path delay after delay minimization by Design Compiler. CPLEX was restricted to 1,000 seconds of computation per ILP run to limit computational runtime to something reasonable.

Netlist	Min Delay (ns)	Power (mW)					
		1.1Tmin			1.2Tmin		
		LP	LP then DC	CPLEX ILP	LP	LP then DC	CPLEX ILP
c17	0.094	0.96	0.95	1.06	0.76	0.76	0.76
c432	0.733	2.21	2.18	2.53	1.74	1.70	1.73
c499	0.701	4.59	4.48	4.64	3.73	3.64	3.72
c880	0.700	3.45	3.13	3.62	2.60	2.54	2.59
c1355	0.778	5.42	5.26	5.38	4.12	4.04	4.14
c1908	0.999	3.08	3.01	3.14	2.40	2.36	2.42
c2670	0.649	8.42	8.28	8.81	6.87	6.79	6.88
Minimum power savings vs. LP:		1.1%	-14.4%			0.0%	-0.7%
Average power savings vs. LP:		3.0%	-5.3%			1.7%	0.0%
Maximum power savings vs. LP:		9.2%	0.8%			2.4%	0.6%

4.6.4 Comparison versus integer linear programming

Rather than choosing a threshold for when a gate v should be assigned a different cell, we can restrict the cell choice variable γ_v to zero or one, $\gamma_v \in \{0,1\}$. This changes the linear programming formulation in equations (4.20) and (4.22) to a mixed integer linear programming problem. The integer linear programming (ILP) problem is NP-complete and takes much more computation time to solve than the polynomial-time solvable linear programming problem.

As an example of the huge ILP run times, a single ILP iteration with the CPLEX solver [101] for benchmark c432 terminates after 47,000 seconds when CPLEX runs out of memory with a branch and bound tree size of more than 3GB. In comparison, a single LP iteration with the COIN-OR CLP solver takes less than 2 seconds to complete. At the point when CPLEX terminates, there is a duality gap of 2% between the best integer solution found and the upper bound on an integer solution.

As the ILP runtime grows rapidly with circuit size, we limited computational runtime in CPLEX to 1,000s, resulting in about a total runtime of around 20,000s for 20 ILP iterations. This is a large runtime for these small benchmarks, but might be justified if ILP gives large power savings over

the LP approach with thresholds. At the time limit for larger circuit sizes, there is often a duality gap of 50% or more between the best integer solution found by the ILP solver and the upper bound on an integer solution. Consequently, the benefits of ILP may be limited.

With ILP runtime limited to 1,000s per iteration, the largest power saving that we saw with ILP versus LP was 0.8%, as detailed in Table 4.3. Due to the large ILP runtimes and increasing ILP duality gap for larger benchmarks, we have only results versus ILP for the smaller ISCAS’85 benchmarks. In all the cases where ILP provided a marginal benefit over LP, the results after performing a Design Compiler post-pass to clean up the LP results provided greater power savings. Thus, there is no advantage to using ILP instead of our thresholding heuristic with LP.

In some cases the results with ILP are actually worse than LP. The optimization parameters for the outer loop heuristic were tuned to produce the best results with the LP approach, which may disadvantage the ILP results in some cases. For example at a delay constraint of $1.1 T_{\min}$, the ILP approach fails to reduce the delay below the delay constraint after the sixth iteration and thus there are no further power savings – successful delay reduction may have been possible by imposing a tighter delay constraint for delay reduction using a smaller value of τ in Equation (4.22). In addition, the duality gap for an ILP iteration may lead to the results of multiple ILP iterations being suboptimal.

More gates are assigned to a different cell on early ILP iterations in comparison to LP iterations, as the ILP solver manages to find solutions where more cells can be changed to reduce the power further. For example for benchmark c432, 160 cells are changed on the first LP iteration and 174 cells are changed on the first ILP iteration. Using up additional timing slack to change more cells on early iterations is not necessarily advantageous, as the timing slack might be better used on later iterations after other cells have been changed. The LP and ILP solvers cannot take into account the net effect of a cell being changed multiple times over several iterations, nor the

second order impact of multiple cells being changed simultaneously. Thus, the LP approach may also give better results because fewer cells are changed, so there are less second order effects, and less timing slack is used up on each power minimization iteration.

Table 4.4 Number of iterations for gate sizing with the linear programming optimization flow to find a solution that satisfies the delay constraint and is within 1% of the minimum power found in 40 LP iterations. Runtimes for 20 iterations at a delay constraint of $1.2 \times T_{\min}$ with the 0.13um Vdd=1.2V, Vth=0.23V library and conservative slew analysis using the Design Compiler delay minimized netlists as a starting point are also listed. The number of iterations to get within 1% does not depend on the circuit size, whereas the runtime grows roughly quadratically.

Benchmark	# logic levels	# gates	# iterations to get within 1%		Runtime for 20 iterations (s)		
			At 1.1xTmin	At 1.2xTmin	Total	LP Solver	Total - LP Solver
c17	4	10	14	4	0.6	0.2	0.4
c432	24	259	15	21	28.2	8.8	19.4
c880	23	484	16	14	49.8	16.4	33.4
c1908	33	635	5	12	70.5	26.6	44.0
c499	25	644	21	14	63.7	23.7	40.0
c1355	27	764	13	15	79.4	30.5	49.0
Huffman	29	774	16	20	67.7	27.3	40.4
c2670	23	1,164	12	17	105.4	42.4	63.1
c3540	36	1,283	12	9	223.5	114.6	108.9
c5315	34	1,956	9	9	231.6	96.7	134.9
c7552	31	2,779	13	12	383.8	165.2	218.6
c6288	113	3,544	21	15	1,811.2	1,450.5	360.7
SOVA_EPR4	110	15,686	8	7	3,427.1	2,106.1	1,321.0
R4_SOVA	144	33,344	5	5	20,078.2	17,679.1	2,399.0

4.7 Computational runtime

This section examines the runtime for the linear program, and then compares it to the TILOS-like sizing runtimes in Design Compiler. We then briefly discuss the memory requirements.

4.7.1 Theoretical runtime complexity and analysis of actual runtimes

It is not straightforward to determine the theoretical worst case runtime complexity. The open source COIN-OR LP solver [66] uses the simplex method to solve the linear program, which has exponential runtime growth with problem size in the worst case. There are linear programming methods with guaranteed polynomial runtime; however, typically the simplex method is a fast method for solving linear programs.

Each vertex in the directed acyclic graph representation has one “free” cell choice variable, and edges between vertices (i.e. wires between gates) determine the constraints. As a result, our linear program constraint matrix is sparse, because the number of edges is of similar order to the number of vertices ($O(|E|)$ is $O(|V|$ for our benchmarks). Thus we might expect that runtime growth with circuit size will be reasonable.

To measure the actual runtimes, we need to consider when optimization should be terminated. The linear programming optimization flow consists of two approaches: power minimization subject to a delay constraint, and delay reduction with a weighting on power. In both of these, a linear program is posed and solved to determine which cells to change. These alternating optimization phases shift back and forth in the power-delay space about the delay constraint, as was illustrated in Figure 4.10. In addition, the more sophisticated optimization approach changes parameter settings if optimization progress is slow. Consequently, there is no clearly defined optimization endpoint. However, the point at which any further power savings are minimal can be measured. To do this, we can run a large number of iterations, where each iteration refers to a run of setting up the LP, solving it, and changing cells, whether this is for power minimization or delay reduction.

We examined the number of iterations required to get within 1% of the best solution found in 40 iterations. It appears that the number of iterations to get a good solution with gate sizing is not dependent on the circuit size or circuit depth in terms of logic levels. This is because gates are sized simultaneously on each iteration. From the data in Table 4.4, about 20 iterations is sufficient to get good results. Fewer iterations were required for the two largest benchmarks (SOVA_EPR4, R4_SOVA). If fewer iterations are required for larger benchmarks, the growth of runtime with circuit size will be less.

The runtime for twenty iterations for sizing the benchmarks on a 2GHz Athlon XP with 512KB of L2 cache is shown in Figure 4.12. The total runtime and the runtime just for the linear program solver appear to grow quadratically with circuit size. The runtime for static timing and power analysis, posing the linear program, and changing cells using the LP solution scales linearly with the circuit size. The linear program solver runtime is the dominant portion of the total runtime for the larger circuits.

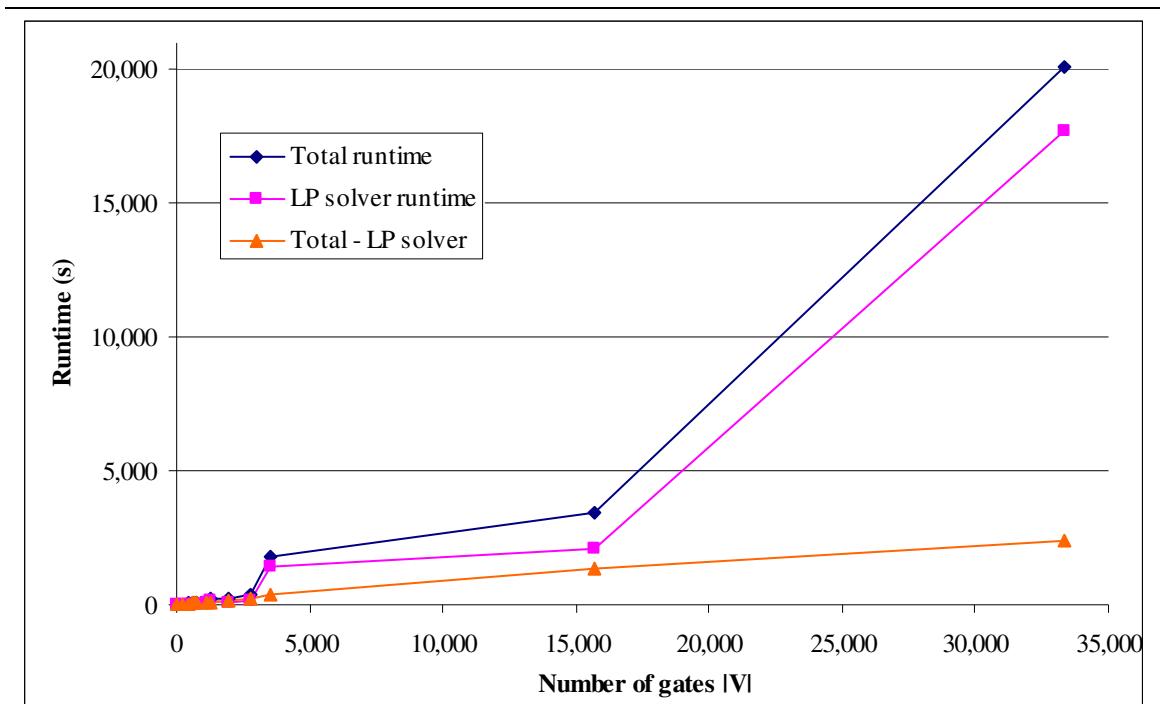


Figure 4.12 Runtime for 20 iterations of the linear programming gate sizing optimization versus circuit size. This is at a delay constraint of $1.2 \times T_{\min}$ with the 0.13um Vdd=1.2V, Vth=0.23V library, with conservative slew analysis. The total runtime, runtime for the linear program solver, and runtime without the linear program solver are shown. The linear program solver runtime appears to scale quadratically, whereas the runtime of the rest of the code grows linearly with circuit size. These runtimes were on a 2GHz Athlon XP with 512KB of cache.

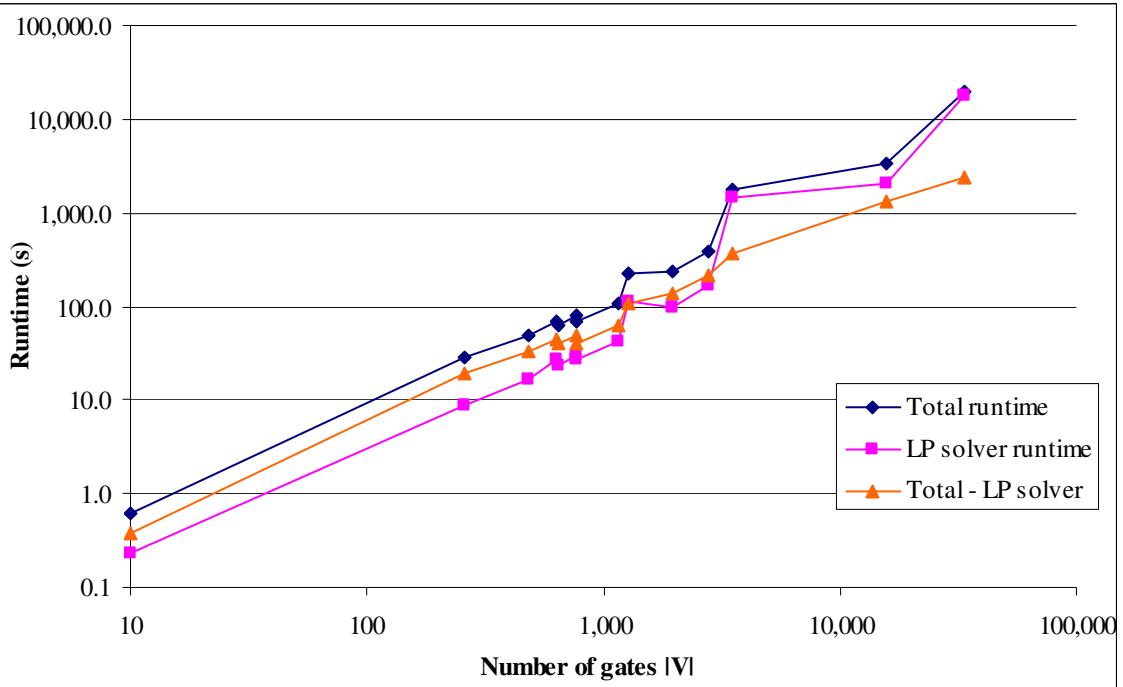


Figure 4.13 Runtimes in Figure 4.12 shown on a log-log scale.

Due to the wide range of circuit sizes, over three orders of magnitude, we can examine the runtime in more detail on a log-log scale in Figure 4.13. This shows that the runtime for static timing and power analysis, posing the linear program, and changing cells using the linear program solution dominates the total runtime for the smaller netlists, below about 1,000 gates. If the runtime growth had a constant exponent (say m , referring to growth of the form $|V|^m$), such as linear or quadratic growth, we would expect a straight line for the linear program solver runtimes on the log-log scale. However, it becomes apparent that the LP solver runtime is substantially larger at three points (benchmarks c3540, c6288 and R4_SOVA) than what might be expected from the other runtimes.

To examine this in more detail, we compare the LP solver runtime on a log-log scale versus lines that correspond to growth with a constant exponent in Figure 4.14. The smallest benchmark c17 is omitted as that runtime is very small and primarily consists of initialization routines for the solver. It is apparent from Figure 4.14 that the increase in runtime for the LP solver varies between $O(|V|)$ and $O(|V|^2)$.

The runtime for examining different cell choices to pose the linear program increases linearly with the number of cell choices available for gates, though this can be reduced by pruning – for example, limiting changes to increasing one size up or down. With a single supply voltage, the number of linear program iterations required to converge did not change significantly with different libraries, including using multiple threshold voltages. The runtimes do not vary significantly with the delay constraint.

One interesting observation was that the LP solver runtimes for power minimization are almost always slower than the LP solver runtimes for delay reduction with a weighting on power. On average LP solver power minimization runs were about twice as slow as delay reduction runs for the larger benchmarks (c6288, c7552, SOVA_EPR4 and R4_SOVA). It is not clear whether this is an artifact of the COIN-OR LP solver [66] that was used, or whether it is a genuine feature of the different way the problem is posed for power minimization versus delay reduction. Also, the LP solver runtimes can vary by up to a factor of 2.4 for both power minimization and delay reduction, with runtime standard deviation of between 1% and 19% depending on the benchmark.

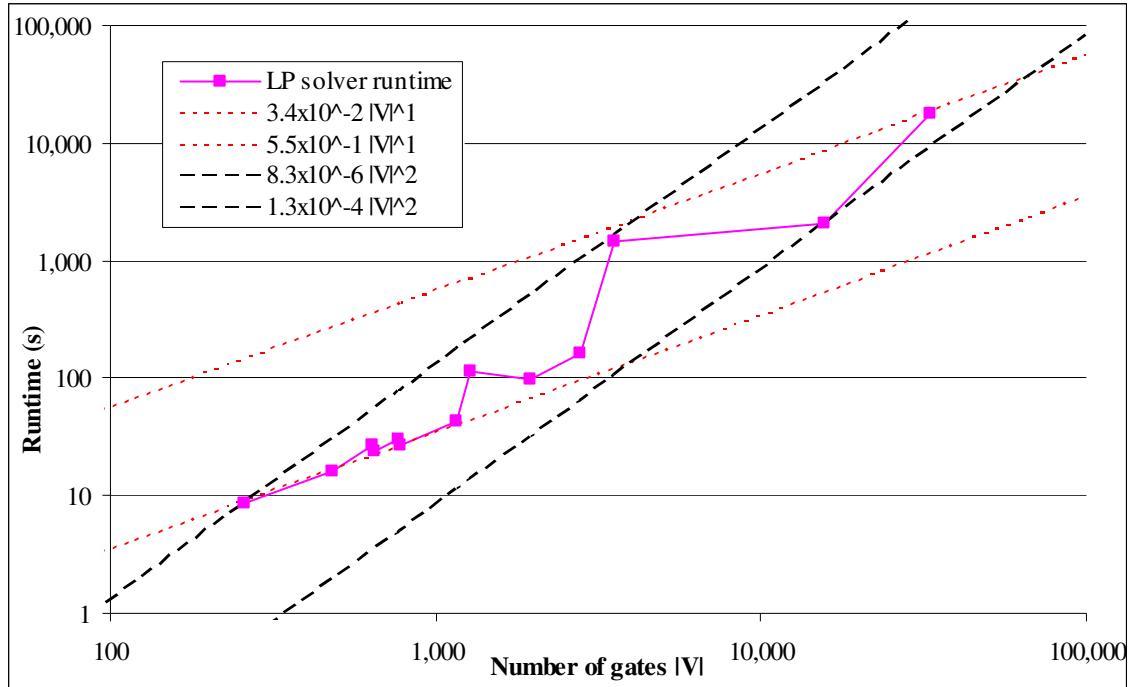


Figure 4.14 Linear program solver runtimes from Figure 4.12 shown on a log-log scale. The LP solver runtime growth varies between $O(|V|)$ and $O(|V|^2)$. Note this is the total for 20 LP solver iterations.

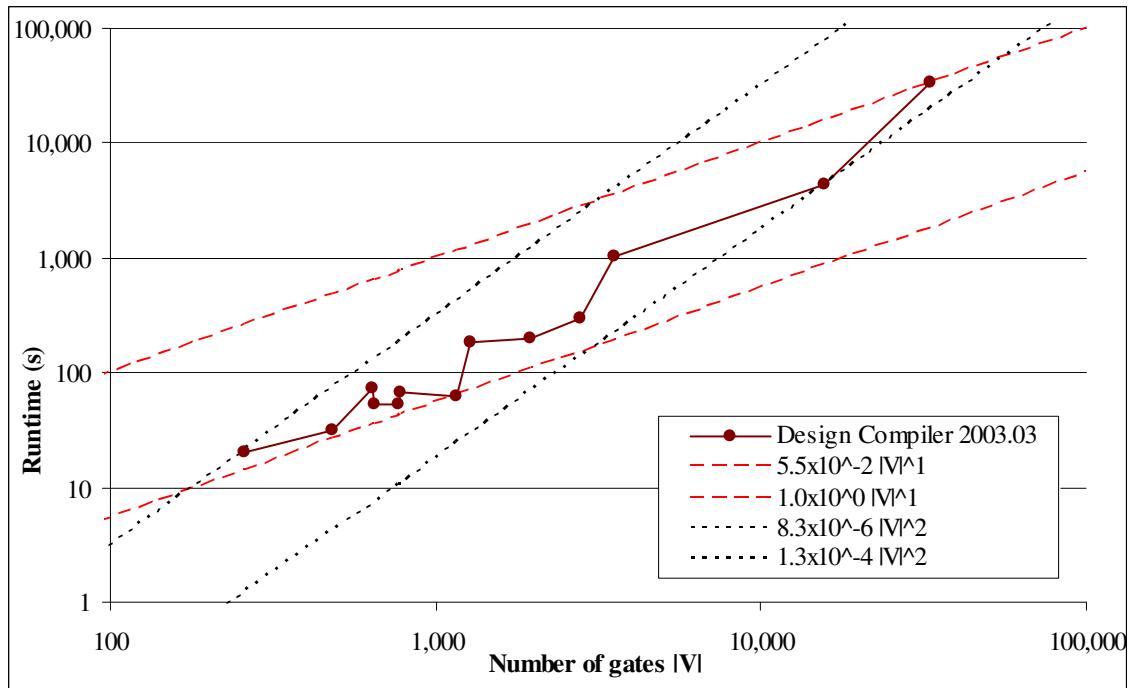


Figure 4.15 Runtime for running power minimization in Design Compiler on the best solution found by linear programming, that is for the “LP then DC” Design Compiler run. These runtimes were on a 300MHz Sun Ultra II. Design Compiler’s runtimes grow faster than $O(|V|)$, but slower than $O(|V|^2)$.

4.7.2 Runtime comparison versus other optimization approaches

In benchmarks using 0.13um libraries, the LP runtimes were about the same as the simpler linear programming approach [154] that we tried initially. Note that runtimes in [154] are about 10× faster with linear 0.18um delay models versus interpolating 0.13um library data. Computational overheads for more accurate delay analysis and more iterations were avoided by incremental analysis with caching.

It is also interesting to compare versus Design Compiler’s runtimes, run on a 300MHz Sun Ultra II, shown in Figure 4.15. This was for Design Compiler performing power minimization on the netlists after linear programming optimization, where the analysis is similar to what occurs in a single iteration of linear programming power minimization. Accounting for the much slower computer used to run the benchmarks, power minimization in Design Compiler is about an order of magnitude faster than our runtimes. Our runtimes in Figure 4.14 and Design Compiler’s runtimes in Figure 4.15 have quite similar shapes – performing faster on certain benchmarks and slower on others.

Several methods should be considered to improve the linear programming runtimes. Firstly, the standard settings for the linear program solver have been used, which performs analysis with the simplex algorithm to converge to a precise local minimum. However, we don’t need the same degree of precision, and relaxing the accuracy requirement would reduce the number of simplex iterations and speed up the LP solver. Secondly, the first few LP iterations provide the biggest power savings. Later iterations tend to bounce around the optimal solution as too many gates are being changed simultaneously. For c432 at a delay constraint of $1.1 \times T_{\min}$ in Table 4.2, we found the Design Compiler could use the LP result as a starting point to find a good solution. Thus it may be possible to run fewer LP iterations and then use a TILOS sizing post-pass to clean up the result. Thirdly, additional analysis could be performed with the solution from the LP solver to avoid changing gates that would cause a delay constraint violation, or to use an alternate cell that

reduces power but avoids the delay constraint violation. Lastly, a better delay minimization approach, getting closer to the delay constraint, may help speed up convergence. These latter suggestions have computational overheads too, so experiments are needed to see what benefit they offer.

Using a commercial linear programming solver could also reduce runtimes. For example, the commercial CPLEX LP solver [101] averaged about 40% lower runtime compared to the open source COIN-OR LP solver. However, CPLEX was about 30% slower for the largest benchmark, R4 SOVA.

4.7.3 Memory requirements

The memory requirements are reasonable for today's computers. Memory usage is linear with circuit size. The linear program solver takes about 400MB of memory on the largest benchmark R4_SOVA, and the rest of the software takes up to 2GB on this benchmark. Note that a substantial portion of the memory requirements are due to the verbose naming conventions to aid human readability in the LP solver input and output files. Additionally, caching by a verbose naming scheme is used to avoid repeating analysis to find the best alternate cell for a gate each iteration if it and its surroundings have not changed. Cached conditions that haven't been used in a recent iteration could be removed to free up memory. The MPS constraint file (i.e. the linear program) for the LP solver for R4_SOVA is 230MB, but compressed it is only 13MB, which indicates the overheads due to the verbose naming scheme.

4.8 Summary

The gate sizing results in this chapter demonstrated that commonly used TILOS circuit sizing approaches are suboptimal. It was known that this traditional approach to gate sizing could be suboptimal for small circuit examples, but it was not clear how to address the problem, nor whether there was significant suboptimality on typical circuits.

Our linear programming optimization flow that simultaneously optimizes all gates in the circuit achieved on average 14.4%, 17.8% and 22.1% power reduction versus a TILOS-like sizer for delay constraints of $1.1 \times T_{\min}$, $1.2 \times T_{\min}$, and $1.5 \times T_{\min}$ respectively. Comparing to the commercial implementation of a TILOS sizer in Design Compiler, the power savings on average were 12.0% and 14.5% at $1.1 \times T_{\min}$ and $1.2 \times T_{\min}$ respectively. We achieved a power reduction of 31% on one circuit.

Iterating cycles of reducing power then reducing delay to meet the delay constraint provides more power savings than stopping power minimization when the delay constraint is reached: further cycles of delay reduction then power reduction get out of this local minimum. Results also demonstrated the importance of having accurate delay and power analysis within the optimization formulation. In particular, it is important to consider slew and separate timing arcs, which much academic optimization research tends to avoid.

The runtime for posing the linear program constraints and changing cells using the linear programming solution scales linearly with circuit size. The LP solver runtimes scale between linearly and quadratically with circuit size, so this approach is applicable for larger circuits. Some approaches that may be useful for reducing the runtime of the linear programming solver have been outlined.

There are two improvements that may be made to our approach. (1) A traditional sizing tool, like Design Compiler, is better for pure delay minimization. This greedy, one gate at a time, optimization approach is also useful for a slight further improvement after our optimization. It was observed that a post-pass sizing individual gates with Design Compiler improved on the linear programming results by a further 2% to 3%. (2) If $\gamma <$ threshold to change a gate's cell, other cells which require less slack could be considered. In particular, the current linear programming formulation is not applicable for a tight delay constraint, as the delay reduction

phase is incapable of meeting a very tight delay constraint. In practice, this will not generally be a major issue when power is a significant constraint, as the delay constraint is usually relaxed a little to allow a more energy optimal solution, rather than many gates being upsized at a tight delay constraint.

Given the computational complexity for this non-convex gate sizing optimization problem, it is not possible to compare the results found to the global minimum except for very small circuits. In comparison to other heuristic approaches, there was only one case where the linear programming approach was worse than the TILOS-like optimizers. This was for c880 at $1.1 \times T_{\min}$ in Table 4.2, where the power was 2.4% higher than Design Compiler. On average, starting the LP approach with conservative slew analysis provided better results than for other parameter settings. The best results were found by using the linear programming approach with a post-pass by Design Compiler. With conservative slew analysis initially, the LP result was 11.4% worse than the result found by LP then Design Compiler for c880 at $1.1 \times T_{\min}$, but was otherwise within 3.8% of the best solution found, averaging 2.4% higher power on average.

Our linear programming approach could be extended to a second order conic program (SOCP) to include the impact of process variation in the manner of [140]. We do not compare our results to those with statistical analysis in [140] as their delay models were inaccurate.

This chapter focused on power minimization subject to a delay constraint, but our approach is equally applicable to area minimization subject to a delay constraint.

The next chapter looks at what power savings may be possible by scaling the supply voltage and the threshold voltage. Chapter 7 analyzes the power savings that can be achieved with use of multiple threshold voltages and multiple supply voltages versus the strong gate-sizing approach provided in this chapter.

Chapter 5. Voltage Scaling

Scaling the supply voltage and threshold voltages to an optimal point for a design can provide substantial power savings, particularly at a relaxed performance constraint. This chapter provides a quick overview of power and delay trade-offs with voltage scaling. We focus on use of a single supply voltage (V_{dd}) and a single threshold voltage (V_{th}). We will examine later in Chapter 6 and Chapter 7 using multiple supply and multiple threshold voltages in comparison to using a single V_{dd} and single V_{th} . Throughout this chapter, we assume that the NMOS threshold voltage and the PMOS threshold voltage are of about the same magnitude, $V_{thn} = -V_{thp}$, and will generally just refer to the value as the threshold voltage.

We examine how supply voltage, threshold voltage and gate size affect the circuit delay and power with simple analytical models in Section 5.1. Then Section 5.2 provides a preliminary examination of the power and delay trade-offs with the 0.13um library at different single V_{dd} and single V_{th} values. Experimental fits are provided for the power and delay, and contrasted to the analytical models. A quick summary is provided in Section 5.3.

5.1 Effect of supply voltage, threshold voltage and gate size on power and delay

It is helpful to examine simple analytical models to discuss the trade-off between power and delay with gate sizing, scaling the supply voltage, and scaling the threshold voltage. The analytical models are also useful to check that the 0.13um standard cell libraries used in Chapter 4 and Chapter 7 were characterized correctly.

These subsections overview analytical models for switching power, short circuit power, leakage power and gate delay. In digital logic, switching power consumption occurs when logic switches from 0 to 1 or 1 to 0, and capacitances are charged and discharged. There is a short circuit current from supply to ground in a gate when both the pull-up PMOS network of transistors and pull-down NMOS network of transistors are conducting. Together, switching power and short circuit

power constitute the dynamic power. Signal glitches propagating also cause switching and short circuit power. Leakage power occurs when logic is idle, whether the circuit is in standby or simply that portion of the logic is not switching. Leakage currents from supply to ground occur through transistors that are nominally “off”: the pull-up PMOS transistors if the output is low, or the pull-down NMOS transistors if the output is high.

5.1.1 Switching power

As digital logic performs computations, logic transitions occur between 0 and 1, charging circuit capacitances to a high voltage, or discharging them back to a low voltage. The switching power for charging and discharging a capacitance C to a voltage V_{dd} and back to 0V with switching frequency f is

$$P_{switching} = \frac{1}{2} C V_{dd}^2 f \quad (5.1)$$

Capacitances in the circuit include internal “parasitic” capacitances within each gate, capacitances of the input pins of each gate, and wire capacitances. In standard cell library characterization, switching power for the internal gate capacitances is included with the short circuit power in the “internal power” of a gate. A gate’s internal capacitances and input pin capacitances depend on transistor width, that is the gate size.

With the quadratic dependence of switching power on supply voltage V_{dd} in Equation (5.1), reducing the supply voltage is often seen as the most effective way to reduce dynamic power. However at a tight delay constraint, reducing gate sizes can provide a greater power reduction: as each gate loads its fanins, if a gate is upsized to reduce delay, then its fanins must in turn be upsized to prevent their delay increasing. Consequently, the gate sizes and power increase rapidly as delay is reduced towards a tight delay constraint. Conversely, if timing slack is available near a

tight delay constraint, reducing a gate's size allows fanins to be reduced in size, and substantial power savings may be achieved.

Note that gate capacitance, and thus switching power, increases a little as the threshold voltage is reduced. As a higher supply voltage reduces the threshold voltage due to drain-induced barrier lowering (DIBL) [172], gate capacitance and switching power also increase when supply voltage is increased.

5.1.2 Short circuit power

Short circuit power is dissipated when there is a current from supply to ground in a gate when both the pull-up PMOS network of transistors and pull-down NMOS network of transistors are conducting. In 1984, Veendrick derived the short circuit current for an inverter without load. He assumed the saturation drain current has the form [227]

$$I_{\text{saturation drain current}} = c\mu \frac{\epsilon_{ox}}{t_{ox}} \frac{W}{L_{eff}} (V_{in} - V_{th})^2 \quad (5.2)$$

where c is a constant; μ is the charge carrier mobility; ϵ_{ox} is the electric permittivity; t_{ox} is the gate oxide thickness; L_{eff} is the effective transistor channel length; W is the transistor gate width (size); and V_{in} is the driving voltage to the NMOS transistor gate. It was assumed that threshold voltages are symmetric, which is approximately the case for our libraries, and that $V_{thn} = -V_{thp}$, where V_{thn} and V_{thp} are the NMOS and PMOS threshold voltage respectively. This gives average short circuit current of [227]

$$I_{\text{short circuit}} = c\mu \frac{\epsilon_{ox}}{t_{ox}} \frac{W}{L_{eff}} \frac{1}{V_{dd}} (V_{dd} - 2V_{th})^3 s_{in} \quad (5.3)$$

However, this derivation assumed that the drain current depended quadratically on the input voltage. In today's technologies, the saturation drain current dependence on input voltage is more accurately modeled with the Sakurai-Newton alpha-power law [173]

$$I_{\text{saturation drain current}} = c \mu \frac{\epsilon_{\text{ox}}}{t_{\text{ox}}} \frac{W}{L_{\text{eff}}} (V_{in} - V_{th})^{\alpha} \quad (5.4)$$

where c is a constant; and α is the velocity saturation index which depends on the technology, and is between 1 and 2. In this form, Veendrick used $\alpha=2$ to determine the saturation drain current. A value for α of about 1.3 is typical for today's technologies. Following a similar derivation to Veendrick, the short circuit power without load in terms of the optimization variables of direct interest to us is

$$P_{\text{short circuit}} = cfW(V_{dd} - 2V_{th})^{\alpha+1} \quad (5.5)$$

where c is a constant and f is the switching frequency.

Consequently, the short circuit power is between quadratically and cubically dependent on the supply voltage, depending on the value of α . It is also linearly dependent on gate size, and increases as the threshold voltage is reduced.

Wei et al. performed similar analysis for short circuit current, and the expression they derived had a similar dependence on gate size, V_{th} and V_{dd} . Their model had error of 19% to 30% versus 0.25um SPICE data, compared to Veendrick's model which had error of 48% to 109% [233]. Thus Equation (5.5) provides a reasonable analytical model, but an empirical fit such as in Section 5.2.3 will be more accurate.

The impact of slew is not included in the above simplified equation. As noted by Veendrick [227], the short circuit power contributes only a minor portion of the dynamic power when the input slews to the gate and the gate's output slew are similar. This minimizes the duration when both PMOS and NMOS transistor chains are conducting, which creates a short circuit current path from supply to ground. A circuit's short circuit power is minimized if the input slews to a gate and its output slews are equal, and the short circuit power increases rapidly as input slew

increases relative to output slew [29]. As input slews are usually similar to a gate's output slew, short circuit power typically contributes less than 10% of the dynamic power [32]. However, this can be an underestimate for low threshold voltages – see analysis in Section 5.2.3.

5.1.3 Leakage power

In deep submicron process technologies with low threshold voltages, the dominant sources of leakage power are subthreshold leakage and gate leakage. Subthreshold leakage occurs when the transistor gate-source voltage V_{GS} is below the threshold voltage V_{th} , and the minority carrier concentration varies across the MOSFET channel causing a diffusion current between drain and source. Gate leakage is due to a high electric field across the thin transistor gate oxide, which results in tunneling of electrons through the transistor gate oxide [172]. In 0.13um technologies such as STMicroelectronics' 0.13um process, subthreshold leakage is by far the most significant component of leakage power, but gate leakage is also included in the technology models. The subthreshold leakage current is [54]

$$I_{\text{subthreshold leakage}} = c \mu \frac{\epsilon_{ox}}{t_{ox}} \frac{W}{L_{eff}} \left(\frac{kT}{q} \right)^2 e^{q(V_{GS} - V_{th0} - \gamma V_b + \eta V_{DS})/mkT} \left(1 - e^{-qV_{DS}/kT} \right) \quad (5.6)$$

where c is a constant; k is Boltzmann's constant; q is the charge of an electron; T is the temperature; V_{GS} is the transistor gate-source voltage; V_{th0} is the zero bias threshold voltage, V_b is the body bias voltage; γ is the linearized body effect coefficient; m is the subthreshold swing coefficient; and η is the drain-induced barrier lowering (DIBL) coefficient.

The subthreshold leakage increases rapidly as the temperature increases. At 25°C, The “ideal” subthreshold slope is $\ln(10)kT/q = 60\text{mV/decade}$ if $m = 1$ in Equation (5.6), though in real processes the subthreshold slope is worse (more) than this [167].

Simplifying the expression in Equation (5.6) to the optimization variables of interest to us, the subthreshold leakage power for an NMOS transistor in an inverter is approximately

$$P_{\text{subthreshold leakage}} = c_1 V_{dd} W e^{c_2(-V_{th0} + \eta V_{dd})} \quad (5.7)$$

where c_1 and c_2 are constants; the input voltage $V_{GS} = 0V$ when it is “off” and leaking; $V_{DS} = V_{dd}$; and $V_b = 0V$ assuming there is no body bias applied. The subthreshold leakage increases exponentially as the threshold voltage V_{th} is reduced. The subthreshold leakage increases linearly with gate size (transistor width W) and with the supply voltage V_{dd} . However, the supply voltage increases leakage further due to the drain-induced barrier lowering term.

5.1.4 Delay

The delay for an input logic transition to cause a transition at the gate’s output is due to the time it takes to charge or discharge the load capacitance and the gate internal capacitances. There are also wire RC delays, resistance multiplied by capacitance, but these contribute less than 1% of the critical path delay for our small combinational benchmarks where wires are not that long.

Using the saturation drain current from Equation (5.4), the delay d for charging a capacitance C from 0V to V_{dd} may be approximated as¹ [193]

$$d = CV_{dd} / I_{\text{saturation drain current}} = k \frac{CV_{dd}}{W(V_{dd} - V_{th})^\alpha} \quad (5.8)$$

where k is a constant; W is the width of the transistor through which current is flowing to charge the capacitor; and α is a constant depending on the technology. Therefore, a gate’s delay is reduced as its size increases, but increases as the size of fanout gates and thus their capacitance increases. A gate also has internal capacitances which contribute an additional “parasitic delay” as it is termed in “logical effort” delay models [193][199]. The delay also increases as the supply voltage V_{dd} is reduced and as the threshold voltage V_{th} is increased. Excluding the driving gate from analysis by using a fixed input voltage ramp, as a gate is upsized to reduce the delay, the reduction in gate delay is less for larger gate sizes due to the parasitic delay. The dynamic power

¹ Note that W_{in} should not be in the numerator in Equation (1) of [193].

for charging and discharging the capacitances in the gate increases linearly with gate size. This produces the classic power-delay “banana” curves shown in Figure 5.1. Including the driving gate in analysis results in a delay increase for larger gate sizes due to the load on the driving gate, as shown in Figure 5.2. Consequently, if a gate is upsized, its fanins may also need to be upsized.

By comparing the impact of gate size, threshold voltage, and supply voltage on delay and these power terms, we see that there are significant delay-power trade-offs that must be carefully analyzed. To minimize power, it is not clear that reducing supply voltage to reduce switching power, or increasing threshold voltage to reduce leakage power, is more important than reducing gate size. All of these choices increase the circuit delay, except for gate downsizing, which may increase delay or reduce delay as the reduced load on preceding gates reduces their delay. Optimization approaches which favor one technique, typically Vdd, over the others will be suboptimal in situations where the power-delay sensitivity to this technique is less. As noted by Brodersen et al. [26], reducing the threshold voltage or increasing the supply voltage, subject to process constraints, to provide slack for gate downsizing can sometimes achieve better overall power savings.

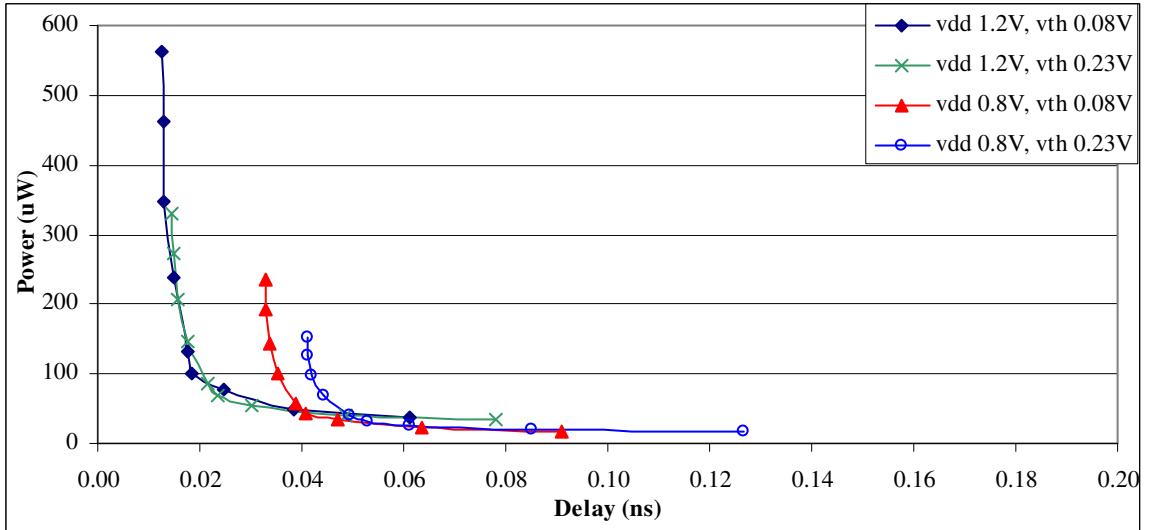


Figure 5.1 Power versus delay for inverter cells from the 0.13um PowerArc characterized libraries. The input signal was a ramp with 0.1ns slew. Each point on a curve is for a different gate size, and larger gate sizes have larger power consumption. The load capacitance was 8fF, and the switching frequency was 4GHz.

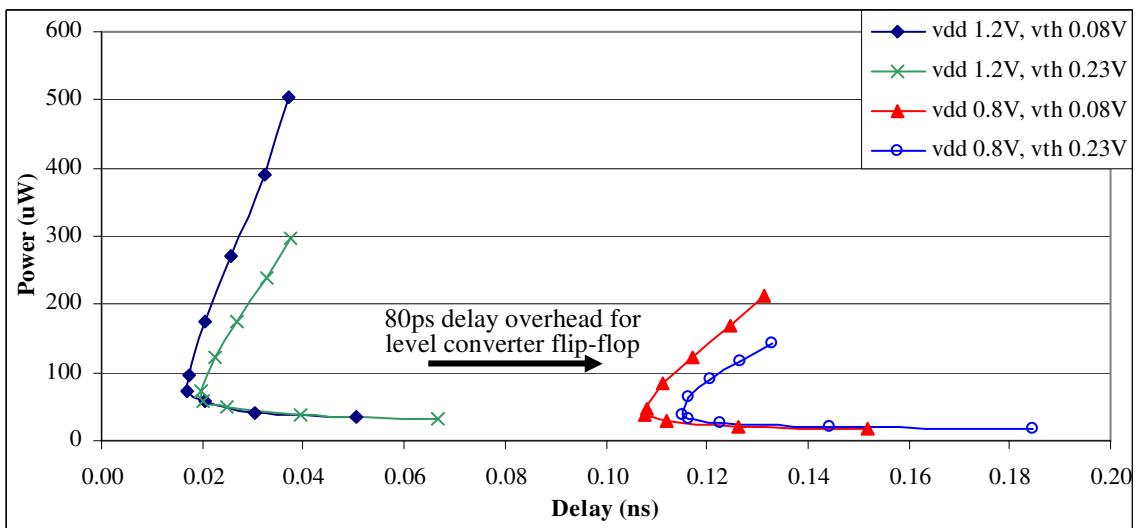


Figure 5.2 Power versus delay for inverter cells from the 0.13um PowerArc characterized libraries. The driving gate was included in the power and delay analysis. In addition for the 0.8V Vdd cells, there is a level converter flip-flop (LCFF) delay overhead of 80ps, but no power overhead, for voltage level restoration to drive the output at 1.2V. Each point on a curve is for a different gate size, and larger gate sizes have larger power consumption. The load capacitance was 8fF, and the switching frequency was 4GHz.

Table 5.1 Delay and total power consumption at full switching activity for the ISCAS'85 benchmark c7552, using PowerArc characterized 0.13um libraries with 1.2V supply voltage, and threshold voltage of 0.23V or 0.08V. Using the netlist delay minimized at 0.23V is sufficient to provide a delay minimized netlist when using the 0.08V library instead, but the power consumption is about 10% higher.

Netlist	Vdd (V)	Vth (V)	Delay (ns)	Total Power (mW)
c7552, delay minimized in dc_shell at vdd=1.2V/vth=0.23V	1.2	0.23	0.847	27.9
c7552, delay minimized in dc_shell at vdd=1.2V/vth=0.23V	1.2	0.08	0.695	47.9
c7552, delay minimized in dc_shell at vdd=1.2V/vth=0.08V	1.2	0.08	0.695	43.6

5.2 Delay and power for 0.13um libraries versus supply and threshold voltage

This section examines delay and power data for different Vdd and Vth values. The libraries are compared at the minimum delay achievable for the particular supply and threshold voltages. The netlists used were the ISCAS'85 benchmarks that were delay minimized in Design Compiler with the PowerArc characterized 0.13um library with Vdd=1.2V and Vth=0.23V. The wire loads were 3+2×#fanouts fF, and output port loads were 3fF excluding the load of the wire to the port.

The power and delay data were normalized to Vdd=1.2V/Vth=0.23V data, and then averaged across the netlists. There was little variation in the normalized data at the same Vdd and same Vth across the netlists. The standard deviation was less than 2%, except for internal power for which there was up to 5% standard deviation and standard deviations ranged from 3% to 9% for the 0.5V Vdd data.

To minimize the delay, it was not necessary to resize the circuits when using a different library. The gate sizes optimized for the Vdd=1.2V and Vth=0.23V library could be used. For example, Table 5.1 shows the delay and power consumption of benchmark c7552 for two different circuit configurations. There is no difference in the delay with Vth=0.08V between using the netlist delay minimized at Vth=0.23V versus the netlist delay minimized at Vth=0.08V. However, the power consumption at Vth=0.08V using the netlist that was delay minimized at 0.23V is 10% higher. For the purpose of providing an initial look at the power and delay at different Vth and

Vdd values, this is sufficiently accurate. We will examine netlists optimized with gate sizing and different Vdd and Vth values later in Section 7.6.

The channel length was 0.13um. To provide a range of threshold voltages, The zero bias threshold voltage parameter vth0 [14] in the SPICE technology files was adjusted to provide a range of threshold voltages. The 0.6V Vdd libraries had been incorrectly characterized using Vdd of 0.8V when characterizing delay and dynamic power. Thus for our work it was important to verify that the other libraries were characterized correctly. Detailed analysis of the libraries, errors in some of the characterized data with supply voltages of 0.5V and 0.6V, and corrections to that data are in Appendix B. As we use Vdd of 0.6V in Chapter 7, the next subsection discusses delay scaling from a supply voltage of 0.8V to 0.6V.

5.2.1 Delay dependence on Vdd and Vth for the 0.13um library

As the delay characterization for Vdd of 0.6V was incorrect, we wish to estimate the delay at Vdd of 0.6V by using an accurate fit to the characterized delays at Vdd values of 1.2V, 0.8V and 0.5V. To the best of our knowledge the library characterization with Vdd values of 1.2V and 0.8V was correct, but several cells were characterized incorrectly with Vdd of 0.5V, as detailed in Appendix B.5. Thus we present delay fits with and without the 0.5V Vdd library. We will now discuss these fits.

Using the analytical delay model in Equation (5.8), the delay scaling with Vdd and Vth is [193]

$$\text{Delay scaling factor from } V_{dd1} \& V_{th1} \text{ to } V_{dd2} \& V_{th2} = \frac{V_{dd2}}{V_{dd1}} \frac{(V_{dd1} - V_{th1})^\alpha}{(V_{dd2} - V_{th1})^\alpha} \quad (5.9)$$

Sarvesh Kulkarni proposed delay scaling using Equation (5.9) and a typical value of $\alpha=1.3$ for 0.13um technology to scale from Vdd of 0.8V to 0.6V. However, the derivation of Equation (5.8) ignores signal slew and other factors in the original delay derivation detailed in the Sakurai-Newton alpha power law paper [173]. The $\alpha=1.3$ delay scaling underestimates the delay by 10.5%

to 19.4% at Vdd of 0.8V and by 36.5% to 52.3% at Vdd of 0.5V. Versus the empirical “higher order” fit that includes the 0.5V Vdd data, the $\alpha=1.3$ delay scaling from Vdd of 0.8V underestimates the delay at Vdd=0.6V by between 17% at Vth of 0.08V to 26% at Vth of 0.23V.

The least squares fit of Equation (5.9) to the 1.2V and 0.8V Vdd delay data gives a value for α of 1.66. This delay scaling was used to scale from 0.8V to 0.6V for the results with Vdd of 0.6V in Chapter 7. The $\alpha=1.66$ fit underestimates the 0.5V Vdd delays by 14.2% to 24.4%. Versus the empirical “higher order” fit that includes the 0.5V Vdd data, the $\alpha=1.66$ delay scaling from Vdd of 0.8V underestimates the delay at Vdd=0.6V by between 6% at Vth=0.08V to 13% at Vth=0.23V. Despite this possible underestimate in delay at Vdd of 0.6V, multi-Vdd analysis in Chapter 7 concludes that using a low Vdd of 0.6V provides minimal benefits over using a low Vdd of 0.8V.

Including the 0.5V Vdd data in the least squares fit, gives a value for α of 1.83, but the fit errors are still up to 6.6%. Thus delay scaling with Equation (5.9) in [193] and other papers is at best somewhat inaccurate if α is fitted correctly, and otherwise wrong when a value for α of 1.2 to 1.3 is assumed typically. $\alpha=1.3$ may provide a reasonable fit for drain current, but accurate delay expressions are more complicated.

From analysis of the more complicated delay expression derived for the Sakurai-Newton alpha power law delay model [173] and experimenting with different fits, the best fit to normalized delay was given by

$$d = 0.587 \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha+1}} + 0.241(V_{dd} - V_{th})^\alpha \quad (5.10)$$

with a value for α of 1.101 which is closer to what might be expected for 0.13um technology. This fit had an average error magnitude of 0.8% and the maximum error is only 2.1%. This fit does not directly include the impact of slew on delay. However, as the net scaling of circuit delay

is accounted for, the scaling of slew is implicitly included. The accuracy of these fits is summarized in Table 5.2. See Appendix B.1 for more detail.

As the supply voltage is reduced, if the threshold voltage is kept constant, the delay begins to increase rapidly. Whereas if both Vdd and Vth are scaled down, then the delay does not increase as much. This is illustrated in the graph of the normalized delay given by Equation (5.10) in Figure 5.3. For example at $V_{dd}=0.8V/V_{th}=0.08V$, the delay is only 10% worse than at $V_{dd}=1.2V/V_{th}=0.23V$.

It is worth noting that the delay fit shown in Figure 5.3 is a smooth surface. Fits with more fitting variables and higher order terms may fit measured points nearly exactly, but can result in a bumpier surface between fitted points.

Table 5.2 This table lists the average error magnitude for the delay fits versus the delay data, with and without the 0.5V delay data. The fit with Equation (5.10) with α of 1.101 is the most accurate.

Delay Fit	Mean error magnitude versus	
	0.8V and 1.2V delay data	0.5V, 0.8V and 1.2V delay data
Equation (5.9) with $\alpha=1.30$	7.3%	17.0%
Equation (5.9) with $\alpha=1.66$	1.5%	6.0%
Equation (5.9) with $\alpha=1.83$	4.8%	4.3%
Equation (5.10) with $\alpha=1.10$	0.6%	0.8%

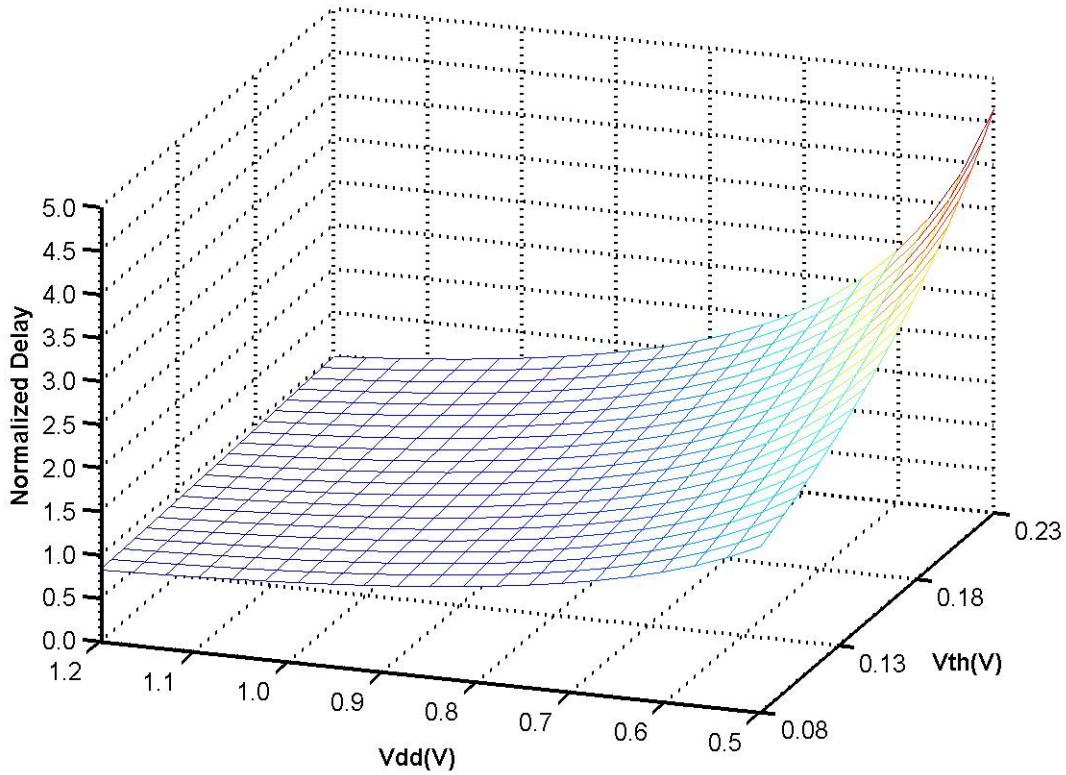


Figure 5.3 Graph of the fit in Equation (5.10) for the minimum delay versus supply voltage and threshold voltage, normalized versus the circuit delays with $V_{dd}=1.2V/V_{th}=0.23V$.

5.2.2 Switching power dependence on V_{dd} and V_{th} for the 0.13um library

In a combinational complementary CMOS circuit, the switching power is the power consumed charging and discharging the wire capacitances, output port capacitances and gate input pin capacitances. The power for (dis)charging gate internal capacitances is included in the internal power, which is covered in Section 5.2.3. Our circuit models assume that wire and port capacitances are fixed and do not vary with supply or threshold voltages. Gate input pin capacitances do vary somewhat with V_{dd} and V_{th} .

By setting the wire loads and output port loads to zero, we determine that wire loads contribute 37% and the output ports contribute 1% of the switching power, excluding switching of gate internal capacitances, on average across the ISCAS'85 benchmarks with the $V_{dd}=1.2V/V_{th}=0.23V$ library. The delay is also 25% more with wire loads versus no wire loads

without resizing gates. We measured the switching power at the original clock frequency to factor out the delay change. The contribution of wire loads to switching power is significant, given that we are performing a comparison at a minimum delay point where gates have been upsized to reduce delay. The significance of wire loads will be even greater at a relaxed delay constraint where gate sizes, and thus gate capacitances, are smaller. The switching power for the wires and output ports depends only on Vdd and the activity.

Given the switching power due to the output port and wire loads, the remainder of the switching power is due to the gate input pin capacitances. We can then determine how gate input pin capacitance C_{in} varies with V_{th} and V_{dd} , by dividing by the V_{dd}^2 term in Equation (5.1). To determine the input pin capacitance dependence on supply and threshold voltages, we perform a least squares fit of a first order Taylor series. This gives

$$\text{Normalized gate input pin capacitance } C_{in} = 0.957 + 0.200V_{dd} - 0.859V_{th} \quad (5.11)$$

where C_{in} was normalized to 1.0 at $V_{dd}=1.2V$ and $V_{th}=0.23V$. This fit has an average error magnitude of 1.0% and maximum error of 2.6%. See Appendix B.2 for more details.

The dependence of C_{in} on V_{dd} has not generally been identified nor quantified in other multi-Vdd optimization research. However, the reduction in V_{th} with increased V_{dd} due to drain induced barrier lowering (DIBL) [172] is well known from a process standpoint. From the data in Appendix B.2 for gate input pin capacitance, there is up to a 22% increase in C_{in} if V_{dd} is increased from 0.5V to 1.2V, and up to a 20% increase in C_{in} if V_{th} is reduced from 0.23V to 0.08V.

5.2.3 Internal power dependence on V_{dd} and V_{th} for the 0.13um library

The internal power has two components: the short circuit power, and the switching power for internal “parasitic” capacitances. From Equation (5.5), for a fit to the internal power we expect terms of the form $(V_{dd} - 2V_{th})^\alpha$ for the short circuit power. From Equation (5.1), we also expect a

term of the form aV_{dd}^2 for the switching power of the gate internal capacitances. A good fit was provided by

$$\begin{aligned} P_{internal} &= P_{switching \ internal \ capacitances} + P_{short \ circuit} \\ &= f(0.413V_{dd}^2 + 0.958(V_{dd} - 2V_{th})^{3.183} + 0.039(V_{dd} - 2V_{th})^{0.162}) \end{aligned} \quad (5.12)$$

The fit has an average error magnitude of 0.4% and maximum error of 1.0%. See Appendix B.3 for more details.

The first V_{dd}^2 term in Equation (5.12) accounts for the switching of internal capacitances. Multiplying the V_{dd}^2 term by a first order Taylor series in Vdd and Vth to consider the dependence of internal capacitance on Vth and Vdd, in the manner of Equation (5.11), does not improve the fit substantially. This suggests that gate internal capacitance is independent of Vth and Vdd.

The second term in Equation (5.12) accounts for the majority of the short circuit power, while the last term provides a slight correction to the short circuit power. The fitted exponents for the short circuit power terms do not correspond to what we might expect from Equation (5.5), if α is about 1.3, but Equation (5.5) does not include the effect of input slew and load capacitance which introduce higher order terms.

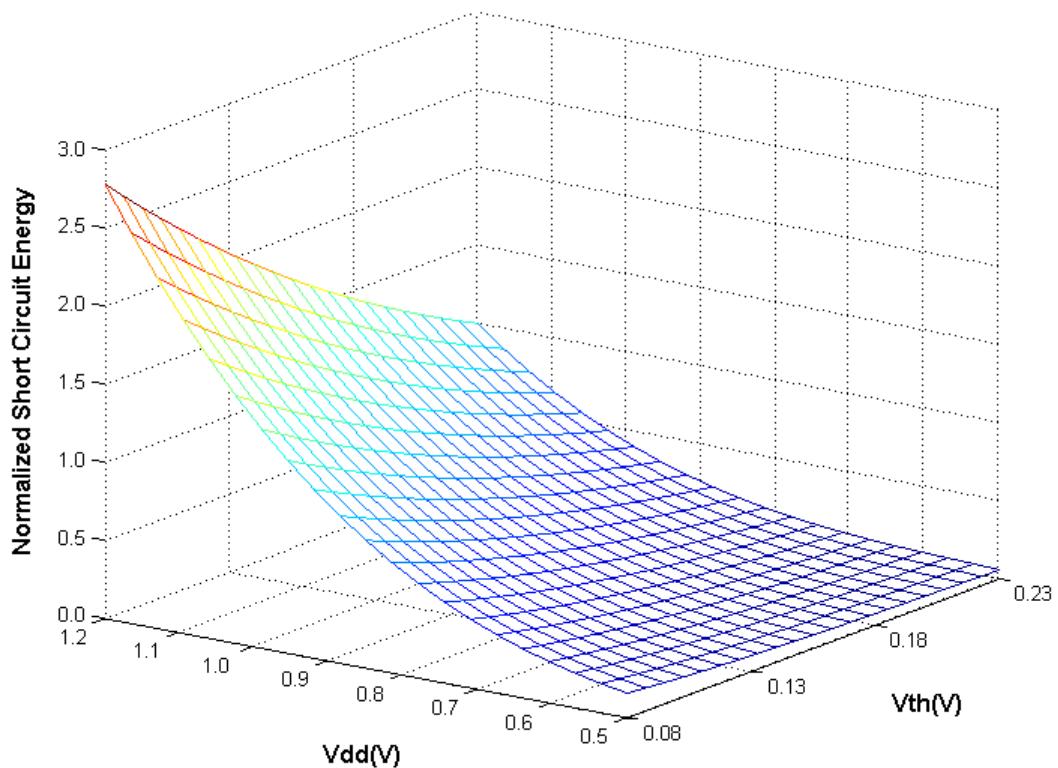


Figure 5.4 Graph of the short circuit energy ($V_{dd} - 2V_{th}$) terms in Equation (5.12), normalized versus the short circuit energy at $V_{dd}=1.2V/V_{th}=0.23V$.

The short circuit energy is graphed in Figure 5.4. As the threshold voltage is reduced, the short circuit power grows significantly. The short circuit energy approaches zero as V_{dd} approaches $2V_{th}$, as predicted by Veendrick [227], because at $V_{dd} = 2V_{th}$ NMOS and PMOS transistors cannot be “on” (i.e. $V_{GS} > V_{th}$) simultaneously, so there is no short circuit current. The short circuit power reduces substantially as Vdd decreases, by about a factor of 10× as Vdd is reduced from 1.2V to 0.6V at $V_{th}=0.08V$. The short circuit power is reduced by this large factor due to the smaller period of time when both pull-up and pull-down transistor chains are conducting.

Table 5.3 The leakage power averaged across the ISCAS'85 benchmarks is shown in the third column for different Vdd and Vth values, normalized to the leakage power at Vdd=1.2V/Vth=0.23V.

Vdd (V)	Vth (V)	Normalized Leakage Power
1.2	0.23	1.00
1.2	0.14	11.40
1.2	0.12	19.38
1.2	0.08	52.43
0.8	0.23	0.47
0.8	0.14	5.56
0.8	0.12	9.58
0.8	0.08	26.61
0.6	0.23	0.30
0.6	0.14	3.55
0.6	0.12	6.15
0.6	0.08	17.33
0.5	0.23 - 0.08	leakage power was characterized incorrectly

5.2.4 Leakage dependence on Vdd and Vth for the 0.13um library

The leakage power increases exponentially as the threshold voltage is reduced. From the normalized leakage power data shown in Table 5.3, there is about a 56× increase in leakage as Vth is reduced from 0.23V to 0.08V. Leakage increases with Vdd, by about 3× as Vdd is increased from 0.6V to 1.2V. Using the analytical model in Equation (5.7) for the leakage power, we get the following fit

$$P_{leakage} = 158V_{dd}e^{-26.9V_{th}+0.770V_{dd}} \quad (5.13)$$

This fit to the leakage data at Vdd of 1.2V, 0.8V and 0.6V has an average error magnitude of 2.6% and the maximum error is 6.0%. This is quite a good fit considering that the function is exponential. The 0.5V Vdd leakage data was determined to be incorrect, because the analytical model in Equation (5.7) gave a bad fit as detailed in Appendix B.4.

From the fitted V_{th} coefficient in Equation (5.13), we can determine the subthreshold leakage slope at 25°C

$$\text{Subthreshold leakage slope} = \frac{\ln 10}{26.9} = 86\text{mV/decade} \quad (5.14)$$

The subthreshold slope of 86mV/decade is about what we expect.

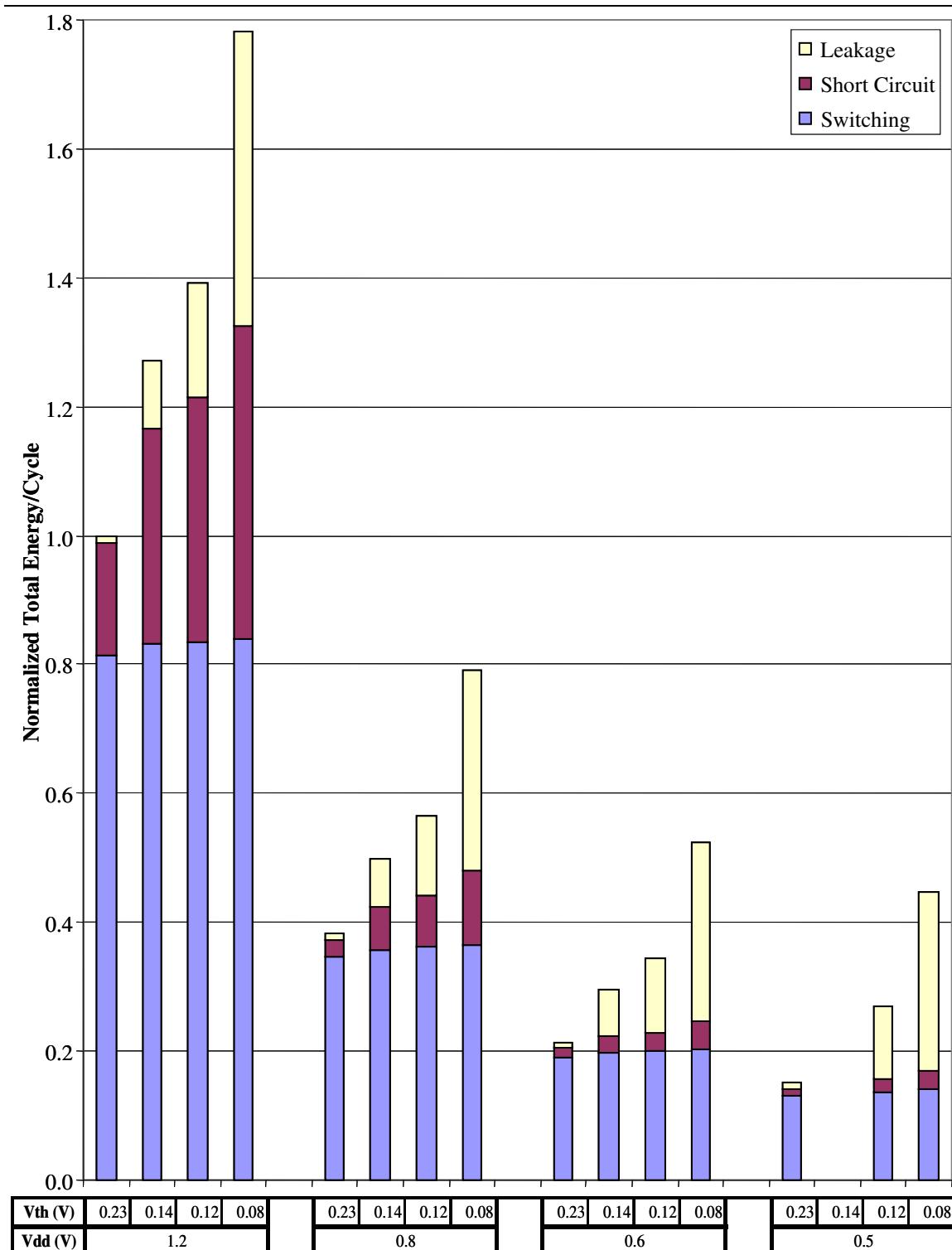


Figure 5.5 This graph shows the normalized total energy per cycle at different supply and threshold voltages, with the breakdown into leakage, short circuit and switching energy. Data at 0.5V Vdd and 0.14V Vth is not included as we did not have that library.

5.2.5 Overall dependence of power on Vdd and Vth for the 0.13um library

From the empirical fits and the earlier analysis of the components of switching energy and internal energy, we can now determine the individual contributions to the total energy, as shown in Figure 5.5. At high Vdd and high Vth, the majority of the energy is to switching of capacitances and short circuit current. Vdd may be scaled down to reduce the dynamic power and then Vth must also be reduced to avoid excessive delay, but the leakage power can become excessive.

The majority of the dynamic power is due to the switching power for the gate internal capacitances, gate input pin capacitances and wire loads. About 42% of the switching power is for (dis)charging the gate input pin capacitances, 32% is due to the gate internal capacitances, and 26% is due to the wire loads. These percentages vary up to $\pm 3\%$ with Vdd and Vth, as gate input pin capacitance varies with Vdd and Vth. Switching of the output ports contributes less than 1% of the switching power. The switching of output port capacitances is not significant as we assumed only a small load of 3fF on the output ports. However in circuits with buses or chip outputs, these capacitances can be much larger and then I/O, receiving/sending input/output data, contributes a significant portion of the chip power. A detailed breakdown of the total power is provided in Appendix B.5.

Leakage power is very significant at low threshold voltages. If threshold voltages are reduced too much, the majority of the power consumption is due to leakage, as leakage increases exponentially with reduction in Vth.

In some cases, the short circuit power can be quite significant, contributing up to 27.2% of the total power at $Vdd=1.2V/Vth=0.08V$. Typically, circuit designers say that short circuit power contributes around 10% of total power, but note that we have not performed power minimization on these circuits. The short circuit power may be larger than typical, but as we are using these

netlists as a starting point for power minimization, it is essential that we model short circuit power as its contribution can be significant. This is contrary to the assumption to ignore short circuit power that many researchers make.

While the power savings by reducing Vdd and increasing Vth can be huge, the accompanying delay increase must also be considered. For example, there is a 20 \times reduction in power going from Vdd=1.2V/Vth=0.14V to Vdd=0.6V/Vth=0.23V, but the delay increases by 3.3 \times . Assuming no delay constraint, the appropriate metric to use is the energy. There is only a 6 \times energy reduction going from Vdd=1.2V/Vth=0.14V to Vdd=0.6V/Vth=0.23V, as can be seen in Figure 5.5. Note that this comparison is somewhat simplistic as we haven't considered power minimization with gate sizing yet. Instead of reducing Vdd and increasing Vth, the timing slack could be used for gate downsizing to reduce gate internal capacitances and gate input pin capacitances. Thus the actual benefits of Vdd=0.6V/Vth=0.23V may be significantly less. In practice, circuits usually have a fixed delay constraint, or designers weigh delay more heavily by looking at PT^2 , where P is the total power and T is the clock period, and similar metrics.

5.2.6 Optimal supply and threshold voltages to minimize total power

We can now determine the optimal supply and threshold voltages to minimize power consumption when meeting a given delay constraint from the delay and power fits for the 0.13um technology. Combining equations (5.10), (5.11), (5.12) and (5.13), we minimize the total power P_{total} with

$$\begin{aligned}
& \text{minimize} && P_{total} \\
& \text{subject to} && T = T_{max} \\
& && T = \frac{0.587V_{dd}}{(V_{dd} - V_{th})^{2.101}} + 0.241(V_{dd} - V_{th})^{1.101} \\
& && P_{leakage} = 158V_{dd}e^{-26.7V_{th}+0.770V_{dd}} \\
& && E_{internal} = 0.413V_{dd}^2 + 0.958(V_{dd} - 2V_{th})^{3.183} + 0.039(V_{dd} - V_{th})^{0.162} \\
& && E_{switching} = (0.62(0.957 + 0.200V_{dd} - 0.859V_{th}) + 0.37 + 0.01)V_{dd}^2 \\
& && P_{total} = 0.011P_{leakage} + \frac{1}{T}(0.430E_{internal} + 0.559E_{switching})
\end{aligned} \tag{5.15}$$

where T is the critical path delay; T_{max} is the delay constraint; $P_{leakage}$ is the leakage power; $E_{internal}$ is the energy/cycle from short circuit currents and switching of internal capacitances; and $E_{switching}$ is the sum of the switching energy of transistor gates (62%), wire loads (37%) and output port loads (1%).

While the optimal Vdd and Vth depend on the process technology, *conventional wisdom based on theoretical analysis by Nose and Sakurai [156] suggests that leakage should contribute 30% of total power to minimize the total power consumption, independent of the process technology, switching activity and delay constraint.* There were a number of incorrect assumptions in their analysis:

- assuming that delay scales as the Sakurai-Newton alpha power law [173] expression for the saturation drain current, Equation (5.8) with velocity saturation coefficient α of 1.3, which underestimates delay by up to 50% at low Vdd as detailed in Section 5.2.1 earlier
- ignoring short circuit current, which contributes from 13% of the total power at a tight delay constraint to 3% at a relaxed delay constraint
- ignoring the reduction in transistor gate capacitance with lower Vdd and higher Vth
- ignoring leakage decreasing with less drain-induced barrier lowering as Vdd is reduced

In minimizing the total power, we find that leakage contributes from 8% to 21% of the total power depending on the delay constraint and how much leakage there is versus dynamic power.

To model the effect of different activities and process technologies, thus the amount of leakage, we consider three scenarios for leakage power: the base case in Equation (5.15); $0.1 \times$ the leakage in Equation (5.15); and $10 \times$ the leakage in Equation (5.15). For each order of magnitude increase in the weight on leakage, the optimal V_{th} averages $0.095V$ higher and the optimal V_{dd} averages $0.17V$ higher, reducing leakage (Figure 5.6) by a factor of $9.0 \times$ which mostly cancels out the $10 \times$ weight increase, in exchange for a 40% increase in dynamic power (Figure 5.7) on average. The total power consumption (Figure 5.8) is also 40% higher on average.

Leakage contributes more power at a tight delay constraint, as shown in Figure 5.9, because a lower V_{th} must be used to meet the delay constraint. As the delay constraint is relaxed, the timing slack enables an exponential reduction in the leakage power by increasing V_{th} (Figure 5.10). The minimum contribution from leakage occurs at a delay constraint of about 0.9. When the delay constraint is relaxed further, V_{dd} (Figure 5.11) is reduced faster than the increase in V_{th} , because of the exponential dependence of leakage power on V_{th} . The contribution of leakage slowly increases as the delay constraint is increased beyond 0.9, because the switching activity decreases as $1/T$, reducing dynamic power but not leakage. In the scenario where there is more leakage, at a tight delay constraint the percentage of leakage is larger because V_{th} must be lower, but at more relaxed delay constraints the percentage of leakage is less because V_{dd} is higher and there is more dynamic power.

In sequential circuitry, the optimal portion of total power from leakage may be higher as dynamic power in idle units can be avoided by clock gating, but power gating and other leakage reduction methods can only be used in standby.

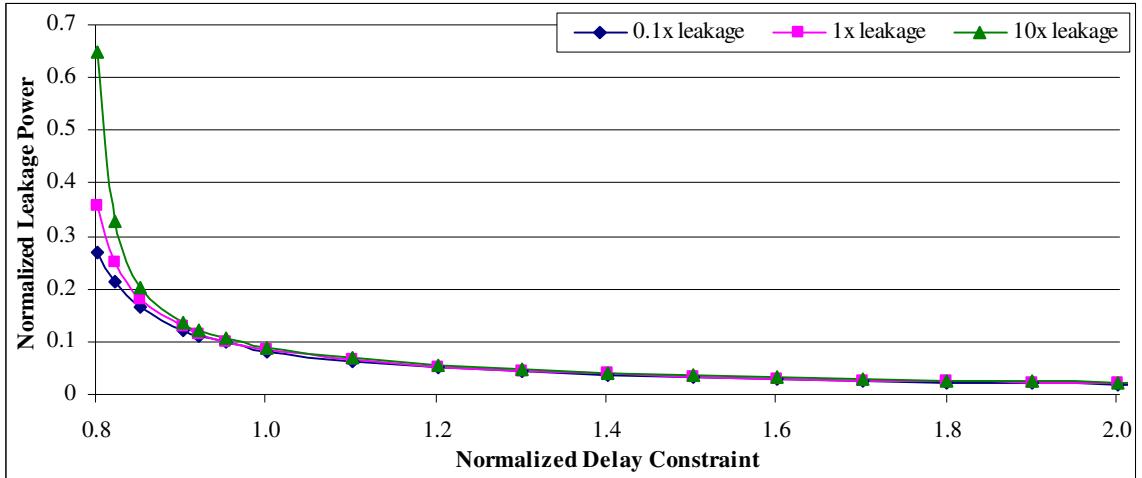


Figure 5.6 The leakage power versus the delay constraint.

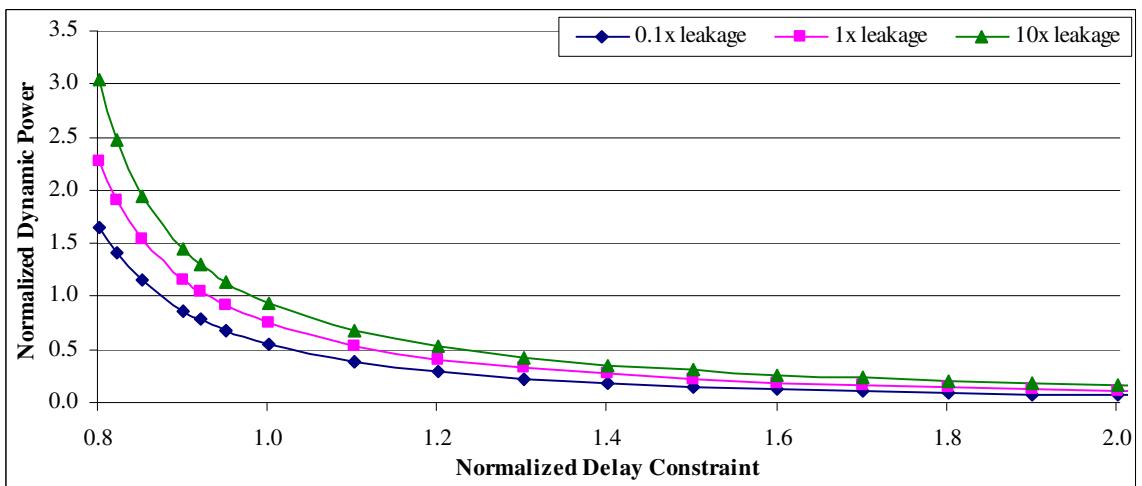


Figure 5.7 The dynamic power versus the delay constraint.

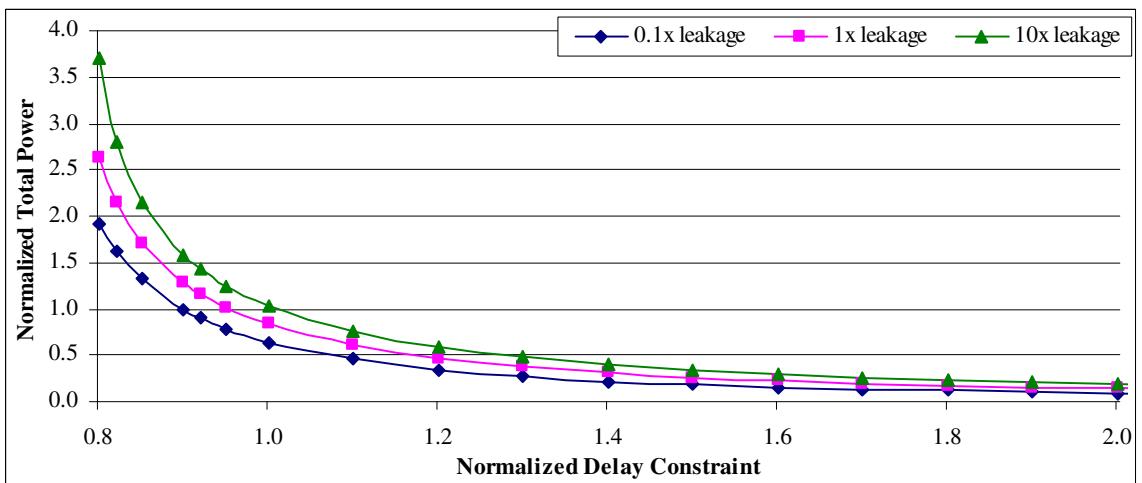


Figure 5.8 The minimum total power consumption versus the delay constraint. Delay and power are normalized versus the values with Vdd of 1.2V and Vth of 0.23V.

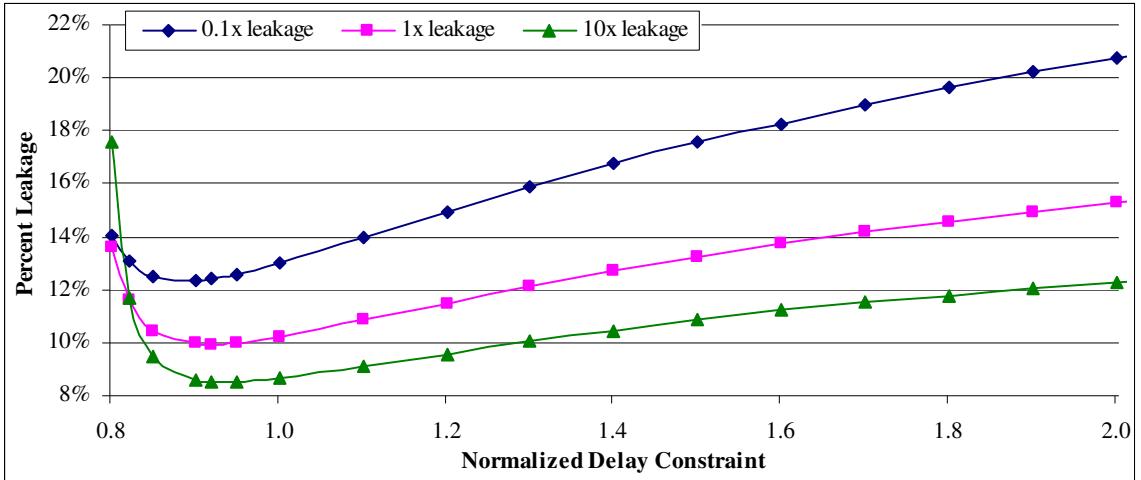


Figure 5.9 Percentage of total power due to leakage versus the delay constraint.

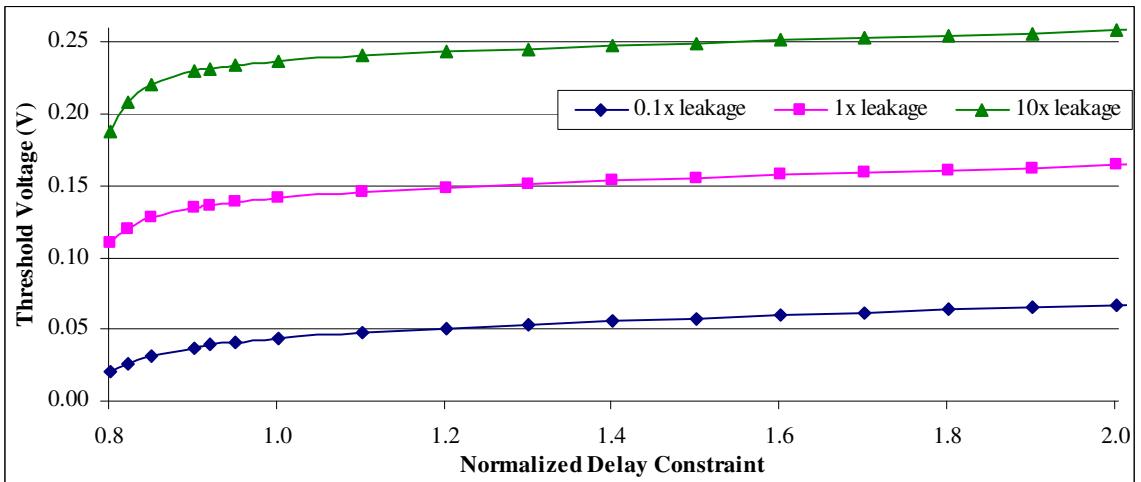


Figure 5.10 Optimal threshold voltage to minimize the total power versus the delay constraint.

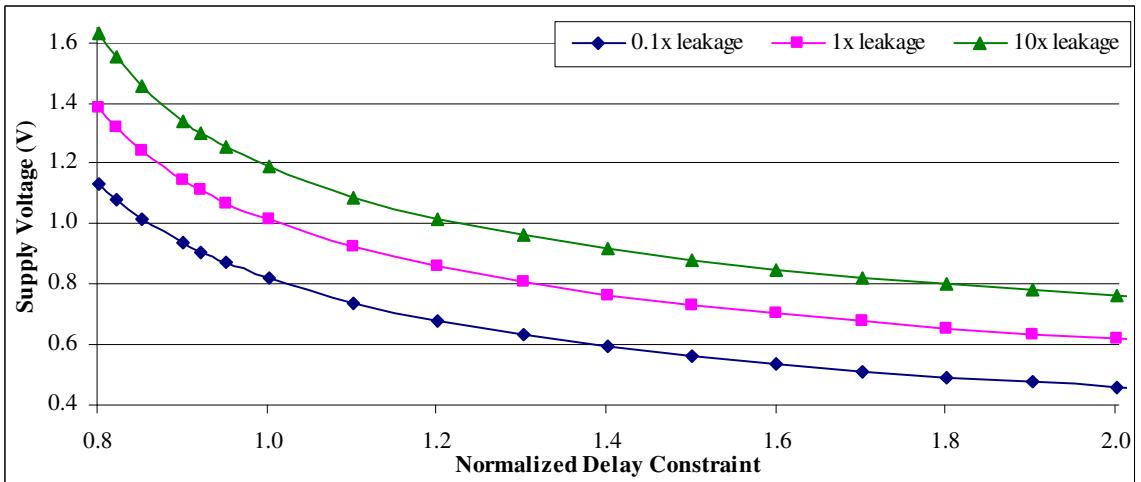


Figure 5.11 Optimal supply voltage to minimize the total power versus the delay constraint.

5.3 Summary

Having examined single Vdd and single Vth libraries, let us summarize the power minimization approaches. *Gate downsizing* reduces the gate internal capacitances and gate input pin capacitances, thus reducing switching power. Reducing the gate size also gives an approximately linear reduction in leakage and short circuit power, due to the higher transistor resistances.

Reducing the supply voltage provides a quadratic reduction in switching power, and also provides substantial reductions in short circuit power and leakage power. *Increasing the threshold voltage* exponentially reduces the leakage power and also reduces the short circuit power. In the 0.13um technology, the leakage at Vth of 0.08V is about $56\times$ the leakage at 0.23V Vth, and the leakage at Vdd of 1.2V is about $3\times$ the leakage at 0.6V.

Both an *increase in Vdd* and a *decrease in Vth* increase the gate input pin capacitance C_{in} , which increases the delay and the switching power. C_{in} increases by 22% if Vdd is increased from 0.5V to 1.2V at Vth of 0.23V, and it increases by 20% if Vth is reduced from 0.23V to 0.08V at Vdd of 0.5V. The dependence of C_{in} on Vdd has not generally been mentioned in other multi-Vdd optimization research, though it can be as significant as the affect of Vth on C_{in} . Other optimization research has noted the impact of Vth on C_{in} [183][229]. In contrast, we found that gate internal capacitances did not depend significantly on Vth or Vdd. Thus reducing supply voltage and increasing threshold voltage provide some additional power reduction by reducing the gate input capacitance.

Except for gate downsizing in some situations, these power minimization approaches come with a delay penalty. If the difference between supply voltage and threshold voltage ($Vdd - Vth$) scales with Vdd then delay is inversely proportional to Vdd. However, Vth scaling is limited by the exponential increase in leakage power as Vth is reduced. Thus the delay may increase substantially when Vdd is reduced. To avoid the delay penalty for low Vdd and high Vth, we can

use high Vdd and low Vth on critical paths, and use low Vdd and high Vth elsewhere to reduce the power. Chapter 6 and Chapter 7 will examine the power savings that can be achieved with use of multiple supply voltages and multiple threshold voltages.

Optimization researchers often exclude wire loads and short circuit power to simplify analysis; however, we found that wire loads can contribute 24% of the total power, and short circuit power can account for 27% of the total power at high Vdd and low Vth. The wire loads also increase the critical path delay, by 25% on average with the $V_{dd}=1.2V/V_{th}=0.23V$ library. Typically, circuit designers say that short circuit power contributes around 10% of the total power, but we did not perform power minimization on these circuits. The short circuit power may be larger than typical, but as we use these netlists as a starting point for power minimization in Chapter 4 and Chapter 7, it is essential that we model short circuit power as its contribution can be significant.

As we use a 0.6V supply voltage in Chapter 7 and there was no 0.6V delay characterization, we discussed how to scale the delays from 0.8V to 0.6V in Section 5.2.1. The net inaccuracy in the power consumption from the 0.8V Vdd characterization ranges from a 0.6% underestimate at $V_{dd}=0.6V/V_{th}=0.08V$ to at most a 1.3% overestimate at $V_{dd}=0.6V/V_{th}=0.23V$, and thus is minor and may be ignored.

From the empirical fits for the delay and power, the optimal Vdd and Vth to minimize the total power consumption can be determined. For example, the optimal Vdd is 1.0V and the optimal Vth is 0.14V for a delay constraint of 1.0; and the optimal Vdd is 0.86V and the optimal Vth is 0.15V for a delay constraint of 1.2, where delays have been normalized to the delay with Vdd of 1.2V and Vth of 0.23V in the 0.13um process technology.

The analysis for the optimal Vdd and Vth does not consider that additional timing slack may be used for gate downsizing. In Chapter 7, we consider selection of Vdd and Vth with gate sizing, but Vdd is limited to 0.6V, 0.8V or 1.2V, and Vth is limited to 0.08V, 0.14V or 0.23V. Without

gate sizing, our analysis would predict that Vdd of 1.2V and Vth of 0.23V are the best choice for delay of 1.0, and that Vdd of 0.8V and Vth of 0.08V are the best choice for delay of 1.2. The optimal Vdd is still 1.2V at a delay constraint of 1.0, but the optimal Vth is lower, 0.14V (see Table 7.3), providing timing slack of 12% of the clock period for gate downsizing, compared to no timing slack with Vth of 0.23V. At a delay constraint of 1.2, the optimal Vdd and Vth remain respectively 0.8V and 0.08V (see Table 7.7 with 0.8V input drivers) as there is sufficient timing slack, 8% of the clock period, for gate downsizing.

Conventional wisdom based on theoretical analysis by Nose and Sakurai [156] suggests that leakage should contribute 30% of total power when Vdd and Vth are chosen optimally to minimize the total power consumption, independent of the process technology, switching activity and delay constraint. Minimizing total power with our more accurate empirical fits, we found that leakage contributes from 8% to 21% of the total power depending on the delay constraint and how much leakage there is, thus depending on the process technology and switching activity. However, the possible Vdd and Vth values depend on the particular process technology and standard cell libraries available. For example for the delay constraint of 1.2 with the best library choice with Vdd of 0.8V and Vth of 0.08V above, leakage contributed on average 40% of the total power.

Chapter 6. Geometric Programming for Optimal Assignment of PMOS and NMOS Widths, Threshold Voltages, and Gate Supply Voltage

Assigning the supply voltage, transistor threshold voltages, and transistor sizes of each gate to a choice of several discrete values to minimize circuit power is a combinatorial problem. Optimization approaches that have been proposed for this problem are heuristic and do not guarantee finding the global minimum, nor do they quantify how suboptimal a solution may be. Using convex models with a continuous range of values, instead of a discrete range, the global optimum can be found in $O(|V|^3)$ polynomial time, where $|V|$ is the number of inputs and gates in the circuit.

We show that posynomials can model variable supply and threshold voltages, with skewed gate drive strengths. Fitted to 0.13um SPICE data, these convex models have root mean square (RMS) errors of 10% or less. With these models, geometric programming gives a convex problem formulation with globally optimal solutions for minimum delay, or minimum power subject to a delay constraint.

For ISCAS'85 benchmarks c17, c432nr and c499, dual voltage and continuous voltage solutions reduce power by less than 9% compared to the optimal solution using single supply voltage with single NMOS threshold voltage and single PMOS threshold voltage. However for benchmark c880, the results with continuous voltages are up to 22.2% lower power than the single voltage results.

Continuous voltage results may be heuristically discretized with $O(\log_2|V|)$ iterations to a discrete dual supply voltage and dual threshold voltage solution, which is at most 5.4% higher power than the continuous voltage lower bound and on average only 2% worse. The largest power saving with dual supply and dual threshold voltages is 18.0% lower power than the single voltage upper

bound for c880; for other benchmarks the small gap between the single and continuous voltage solutions limits the benefit of multiple supply and threshold voltages.

6.1 Introduction

Power minimization is a primary design goal for circuits fabricated with deep submicron technologies. Reducing the power consumption extends battery life, and reduces heat dissipation and packaging costs. Existing commercial tools minimize the power consumption by optimizing the gate size, subject to delay constraints.

Critical paths are assigned high supply and low threshold voltages to reduce the delay. Timing slack on other paths can be used for power minimization. Lowering supply voltage (V_{dd}) quadratically reduces dynamic power. Increasing transistor threshold voltage (V_{th}) exponentially reduces leakage power. Reducing transistor width linearly decreases dynamic and leakage power. We seek an optimal trade-off between these approaches.

Each additional threshold voltage requires an additional mask, adding substantially to fabrication costs and reducing yield due to the additional process complexity. Using multiple supply voltages adds design complexity, requiring level converters and layout of separate supply power rails, increasing the circuit area and hence cost per die. To justify these additional costs, significant power savings must be demonstrated versus using a single supply voltage and single NMOS and PMOS threshold voltages.

A number of algorithms have been proposed for multiple threshold voltage and multiple supply voltage assignment, as summarized later in Section 7.3. Most of these approaches have not performed global optimization, instead adopting a greedy heuristic choosing to change the gate with the maximum power/delay sensitivity on each iteration. Power savings reported vary widely depending on delay and power models, and the baseline which results have been compared to. A quick summary of results with accurate models and a reasonable baseline is presented here.

Optimization with multi-Vth and sizing can give a $\times 6$ reduction in leakage power [183] and a 14% reduction in total power [233] compared to sizing with only low Vth. Using dual supply voltages with gate sizing can reduce power by 14.7% [125] versus sizing with high Vdd.

Global optimization approaches for dual supply voltage assignment have used linear programming and Fiduccia-Matheyses style algorithms, but the gate models were very simple [34][198]. Our geometric programming approach globally optimizes gate sizes, with PMOS and NMOS threshold voltages (V_{thn} and V_{thp}) and gate supply voltage V_{dd} , with more accurate power and delay models.

Accurate analytical models for current technologies are not attractive for optimization. The models are not convex in transistor width due to drain-induced barrier lowering (DIBL) and other deep submicron issues. Furthermore, restrictions imposed by process design rules and standard cell layout result in discontinuities in the underlying data. For example, wider transistors in standard cells must be “folded” into multiple transistors to fit compactly. Without convexity, there is no guarantee that a locally optimal solution is globally optimal. An alternative is to fit convex models to the data.

Single variable, linear piecewise models for delay and leakage in terms of threshold voltage were used in [187]. An iterative heuristic was used to assign thresholds to a choice of several global threshold voltages, and the global values for thresholds were optimized. Even using four threshold voltages resulted in 6% to 18% higher leakage power than using a continuous range of threshold voltages [187]. In contrast, our results with dual Vth, dual Vdd and sizing are at most 5.4% higher total power than the continuous lower bound.

Multivariable convex models are required to simultaneously optimize several variables. Ketkar et al. have shown convex posynomial models have an error of within about 10% for 0.25um data for sizing NMOS and PMOS transistors in gates [117]. Geometric programming with these convex

models was used for transistor sizing to minimize power. We extend previous research by showing that posynomial models can be fit with reasonable accuracy to delay and other gate characteristics, over a wide range of transistor widths, transistor threshold voltages, and gate supply voltage.

We can find exact bounds on solutions with discrete voltages. The upper bound has single supply voltage, single NMOS threshold voltage, and single PMOS threshold voltage. The lower bound allows all variables to be continuous. Heuristically, we find a discrete solution near the global minimum of the continuous solution. Given assignments of gate supply voltage and threshold voltages to discrete global values, for example high Vdd and low Vdd for supply voltage, we can optimally choose the global values while simultaneously performing sizing.

The general problem of gate sizing to minimize a circuit's power consumption is NP-complete, as proven in Appendix C. To simplify the problem, we examine using convex posynomial fits to SPICE data in Section 6.2. Using the posynomial models, we pose the problem as a geometric program, which has polynomial runtime, in Section 6.3. However, the solution of the geometric program gives variable in a continuous range. A heuristic method for finding a discrete solution near the continuous solution is described in Section 6.4. The results are detailed in Section 6.5 and computation runtimes are discussed in Section 6.6. We present our conclusions in Section 6.7.

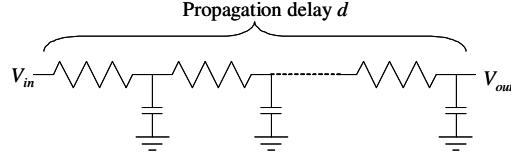


Figure 6.1 An RC chain of resistors and capacitors.

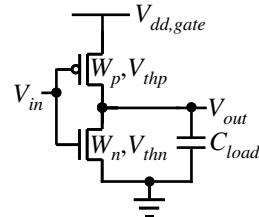


Figure 6.2 An inverter driving a capacitive load with capacitance C_{load} . Transistor widths W and transistor threshold voltages V_{th} are indicated respectively for the PMOS and NMOS transistors.

6.2 Posynomial Models

Posing transistor sizing as a convex optimization problem with posynomial models to minimize area or delay was first suggested by Fishburn and Dunlop in 1985, though the TILOS algorithm that they used was a greedy heuristic [64]. Minimizing the power in a circuit subject to delay constraints was first posed as a convex optimization problem by Mark Matson in 1986 [142].

This early work used simple circuit models based on the Elmore delay [60]. Logic gates and wires were modeled as resistive and capacitive loads, and the delay d for a change in the input voltage V_{in} to propagate to the output voltage V_{out} was simply $d = R_{total}C_{total}$ for an RC chain as shown in Figure 6.1. Models based on the Elmore delay are convex, but very inaccurate for today's technologies.

Accurate analytical models exist for current technologies; however these models are not convex. For example, the falling delay of an inverter is approximately inversely proportional to the falling transistor drain current, which is approximated by the Sakurai and Newton power law model [173]. The falling saturation drain current for an inverter is at first appearance a linear function of NMOS transistor width W_n [81]:

$$I_{saturation\ drain\ current} = kW_n(V_{in} - V_{thn})^\alpha(1 + \lambda V_{out}) \quad (6.1)$$

where k , α and λ are constants, and the other circuit parameters are shown in Figure 6.2. However, the transistor threshold voltage V_{th} is affected by drain-induced barrier lowering (DIBL) and other effects. These effects can be analytically modeled [81], giving

$$I_{\text{saturation drain current}} = kW_n \left(V_{in} - V_{thn} \left(1 + \frac{\eta}{W_n} \right) \right)^{\alpha} (1 + \lambda V_{out}) \quad (6.2)$$

where η is a constant. This is not a convex expression for inverter fall delay in terms of the transistor width W_n due to the subtraction.

Rather than using analytical models, we can fit convex posynomial models to the data of interest. Ketkar et al. have shown this can be done with modeling error within about 10%, using posynomial models to optimize transistor widths W to minimize area or power subject to delay constraints [117]. We extend previous transistor sizing approaches, and allow gate supply voltage $V_{dd,gate}$ and transistor threshold voltages V_{thn} and V_{thp} to be optimized simultaneously. We show that posynomial models can be fit with reasonable accuracy to delay, slew and other gate characteristics of interest, over a wide range of transistor widths, threshold voltages, and gate supply voltages. Posynomial models have the form [117]

$$p(\mathbf{x}) = \sum_{j=1}^m \gamma_j \prod_{i=1}^n x_i^{\alpha_{ij}} \quad (6.3)$$

where the components x_i of variable vector \mathbf{x} are positive, and the positive coefficients γ_j and real-value exponents α_{ij} are fitted to the data. Each product term is a monomial.

The general form of a geometric program utilizing posynomials is [22]

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \leq 1, \quad i = 1, \dots, m \\ & && h_j(\mathbf{x}) = 1, \quad j = 1, \dots, p \end{aligned} \quad (6.4)$$

where f_0, \dots, f_m are posynomials and h_1, \dots, h_p are monomials. The logarithm is taken to transform this to convex form, replacing variables x_i by $y_i = \log x_i$. Thus the posynomial models are limited to modeling positive data.

We consider optimizing a combinational circuit. The circuit can be represented by a directed acyclic graph $G(V,E)$. Figure 6.3 shows a simple example comprising 2-input NAND gates. Each node $v \in V$ represents a logic gate or input driver. The delay and power consumption are determined by the gate characteristics listed in Table 6.1, which depend on the parameters listed in Table 6.2. We need convex models to fit these characteristics over the range of the parameters. In the geometric program, the independent variables to be optimized are the supply voltage $V_{dd,gate}$, NMOS width W_n , PMOS width W_p , NMOS threshold voltage V_{thn} , and PMOS threshold voltage V_{thp} .

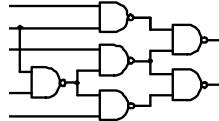


Figure 6.3 ISCAS'85 benchmark c17 [25]. It has six NAND gates. Note that this mapping differs from the mapping for c17 used in Chapter 4 and Chapter 7, for example see Figure 4.1, but it is logically equivalent.

Table 6.1 This table summarizes the data that needs to be characterized for a logic gate.

Gate Characteristic	Represents
$C_{in,fall}$	Gate input pin capacitance when input is falling
$C_{in,rise}$	Gate input pin capacitance when input is rising
d_{fall}	Gate delay when the output falls
d_{rise}	Gate delay when the output rises
E_{dyn}	Energy for switching capacitances and due to short circuit current
$P_{leak,high}$	Gate leakage when output is high
$P_{leak,low}$	Gate leakage when output is low
$S_{out,fall}$	Gate output fall slew
$S_{out,rise}$	Gate output rise slew

Table 6.2 The characteristics of a logic gate depend on these parameters and the SPICE characterization is listed. The transistor width ranges depended on the type of gate – more detail is provided in Appendix D.1.

Parameter	Represents	Range	
		Minimum	Maximum
C_{load}	Sum of fanout input pin capacitances and wire capacitance	3fF	75fF
s_{in}	Maximum fanin slew	0.02ns	0.40ns
$V_{dd,gate}$	Supply voltage of the gate	0.6V	1.3V
$V_{dd,in}$	Minimum input voltage swing	0.6V	1.3V
V_{thn}	Threshold voltage of NMOS transistors in the gate	0.097V	0.357V
V_{thp}	Threshold voltage of PMOS transistors in the gate	0.078V	0.338V
W_n	Width of NMOS transistors in the gate	0.26um	varies
W_p	Width of PMOS transistors in the gate	varies	varies

6.2.1 Data Sampling

We wish to fit posynomial models to delay and power data for our library. The library consists of inverters, NAND2, NAND3, NAND4, NOR2, NOR3, and NOR4 gates characterized with SPICE in STMicroelectronics' 0.13um process technology. Except for the inverter, logic gates have multiple NMOS and PMOS transistors. This subsection details the most important issues in sampling the data; other details are provided in Appendix D.1.

6.2.1.1 Ensuring that the data is positive

Posynomial models are limited to modeling positive data, so we must ensure our data is positive. In particular, care must be taken when fitting the dynamic energy and when choosing how to sample delay data. Note that PMOS threshold voltage is negative, but we use V_{thp} to represent its absolute value.

The energy for an output transition is measured in SPICE as the current drawn from the gate supply times the supply voltage Vdd. This includes any short circuit current that may occur from supply to ground during the transition, when both PMOS and NMOS transistors chains are conducting; and it includes the current to charge and discharge gate internal capacitances and the load capacitance. The rise transition energy is always positive in the complementary CMOS gates, as capacitances are charged up to Vdd. The fall transition energy can be negative, as on occasion some current can flow back to the supply due to coupling capacitances resulting in voltage

overshoot at the output node, where V_{out} exceeds V_{dd} before the output voltage starts to drop [167].

However, the total dynamic energy $E_{dyn,fall} + E_{dyn,rise}$ is always positive, so we use that.

Gate delays are often measured from when the input is at 50% of the maximum input voltage $V_{dd,in}$ to when the gate output is at 50% of the gate voltage $V_{dd,gate}$. If the input voltage changes slowly, it can occur that the gate output voltage transitions past 50% before the input. A slow input transition occurs when the fanin gate is heavily loaded relative to its size, whereas the gate being driven may be driving only a small load and may have a fast output transition. This results in a negative measured delay. $V_{dd,in} > V_{dd,gate}$ increases the occurrence of negative 50%-50% delays. For example, consider a rising input and falling output with $V_{dd,in} = 2V_{dd,gate}$: a fast falling output transition will pass below 50% $V_{dd,gate}$ well before the input voltage passes above 50% $V_{dd,in} = 100\%V_{dd,gate}$.

To avoid negative gate delays, we measured 10%-90% delays: from 10% of a rising input or 90% of a falling input, to 10% of a rising output or 90% of a falling output. The input transition must have started to cause the output to change, so by measuring near when the transitions start, we ensure that a positive time has elapsed between the two. 10%-90% delays were positive over our characterization range. Static timing analysis performed with 50%-50% delays or 10%-90% delays gives essentially the same result for circuit path delays.

6.2.1.2 Number of sample points

Standard cell libraries for a process are typically limited to a couple of supply and threshold voltages. For example, a high speed library with high supply voltage and low threshold voltages, and a low power library with low supply voltage and high threshold voltages. Whereas, intermediate voltages may be optimal. SPICE simulations were used to characterize the gates over the range of parameters. For reasonably accurate fits, we should sample at least 5 points in each variable's range. Typical libraries may have delay lookup tables over five input slews and

five load capacitances. For an inverter there are 8 variables, and hence $5^8 = 390,625$ sample points. For a 2-input NAND gate with 2 NMOS and 2 PMOS transistors, there are 244,140,625 sample points if we wish to assign individual transistor thresholds and widths – this is computationally intractable for our computer resources. Thus we limit it to 8 variables: all NMOS threshold voltages in a gate are equal; all PMOS thresholds in a gate are equal; all NMOS widths in a gate are equal; and all PMOS widths in a gate are equal. There was no tapering of widths in a series chain of NMOS or PMOS transistors – they all have the same width.

The number of data samples to fit is further reduced by several requirements. If the input to a gate has $V_{dd,in} \leq V_{dd,gate}$ there will be significant static current, so we require that $V_{dd,in} \geq V_{dd,gate}$. This is a clustered voltage scaling (CVS) methodology, where it is assumed that voltage level restoration occurs at the registers. CVS is presented in more detail in Section 7.2. The convex geometric programming approach cannot incorporate discontinuous jumps in delay and power. For this reason, it was assumed that there was no delay nor power overhead for voltage level restoration at the combinational outputs, and insertion of asynchronous level converters between combinational logic was not considered. We also assume that the PMOS substrate is biased at $V_{dd,gate}$, which requires spatial isolation of the PMOS transistor wells that are connected to different supply voltages.

The sampling was also reduced by requiring that maximum output slew $s_{out} \leq 0.4\text{ns}$. In retrospect, allowing larger slews would have been reasonable, as slews of 1.0ns or more were seen with Vdd of 0.8V and 0.6V with the linear programming optimization results in Chapter 7. Limiting the maximum slew also ensures that the gate delays are not excessive.

The total number of sample points to fit varied between about 60,000 and 160,000. Performing SPICE characterizations for this large number of data points required several days computation

time for each logic gate on Pentium III 700MHz machines. Standard cell library can take several weeks of computation time, so the time taken is not unreasonable.

6.2.1.3 Channel length

The channel length L was chosen to be 0.18um to avoid short channel issues. This was overly conservative; 0.13um should have been used. Using a longer channel length reduces the leakage, but makes the delay worse and increases transistor capacitances, increasing dynamic power. As shown in Figure 6.4 with channel length of 0.18um, the ratio of NMOS to PMOS drain current varies between 2.9× at 0.6V to 3.7× at 1.3V. Consequently, for equal drive strengths, the PMOS transistor widths need to be three times the width of the NMOS transistors or more. In a typical technology, the ratio is about 2:1, but the choice of 0.18um channel length has a worse impact on the PMOS drain current and consequently rise times. Also note that the drive currents are 5.4× (NMOS) to 6.8× (PMOS) larger at 1.3V versus 0.6V, so at 0.6V the logic is much slower. As described in Chapter 5, the threshold voltage must be reduced to compensate for the reduction in drive current and increase in delay at lower supply voltage.

Driving a low Vdd gate with a high Vdd input increases the pull-down drive strength, as illustrated in Figure 6.4. With fixed drain-source voltage of 0.6V (i.e. for a gate with 0.6V supply voltage), the NMOS drain current with V_{in} of 1.3V is 4.9× that when V_{in} is 0.6V. With NMOS transistor gate voltage of 0.6V, the drain current saturates as the drain-source voltage V_{ds} increases to 0.6V. If V_{in} is 1.30V, the NMOS drain current is larger and has not saturated when the drain-source voltage reaches 0.6V. In contrast, when the pull-up network (PUN) is on, the input voltages are zero. Thus the drive strength of the PUN is not affected by $V_{dd,in}$. For a given drive strength when $V_{in} > V_{dd,gate}$, the NMOS transistors in the pull-down network can be smaller, reducing the power consumption and capacitive load on fanins. Thus skewing gates widths is advantageous when $V_{dd,in} > V_{dd,gate}$, that is the P:N ratio should be larger as the NMOS widths can be smaller. Taking advantage of the increased NMOS drive strength when $V_{dd,in} > V_{dd,gate}$, we

expect optimized multiple supply voltage circuits to have a higher mean PMOS to NMOS width ratio than single supply voltage circuits, which is demonstrated by the results in Section 6.5.

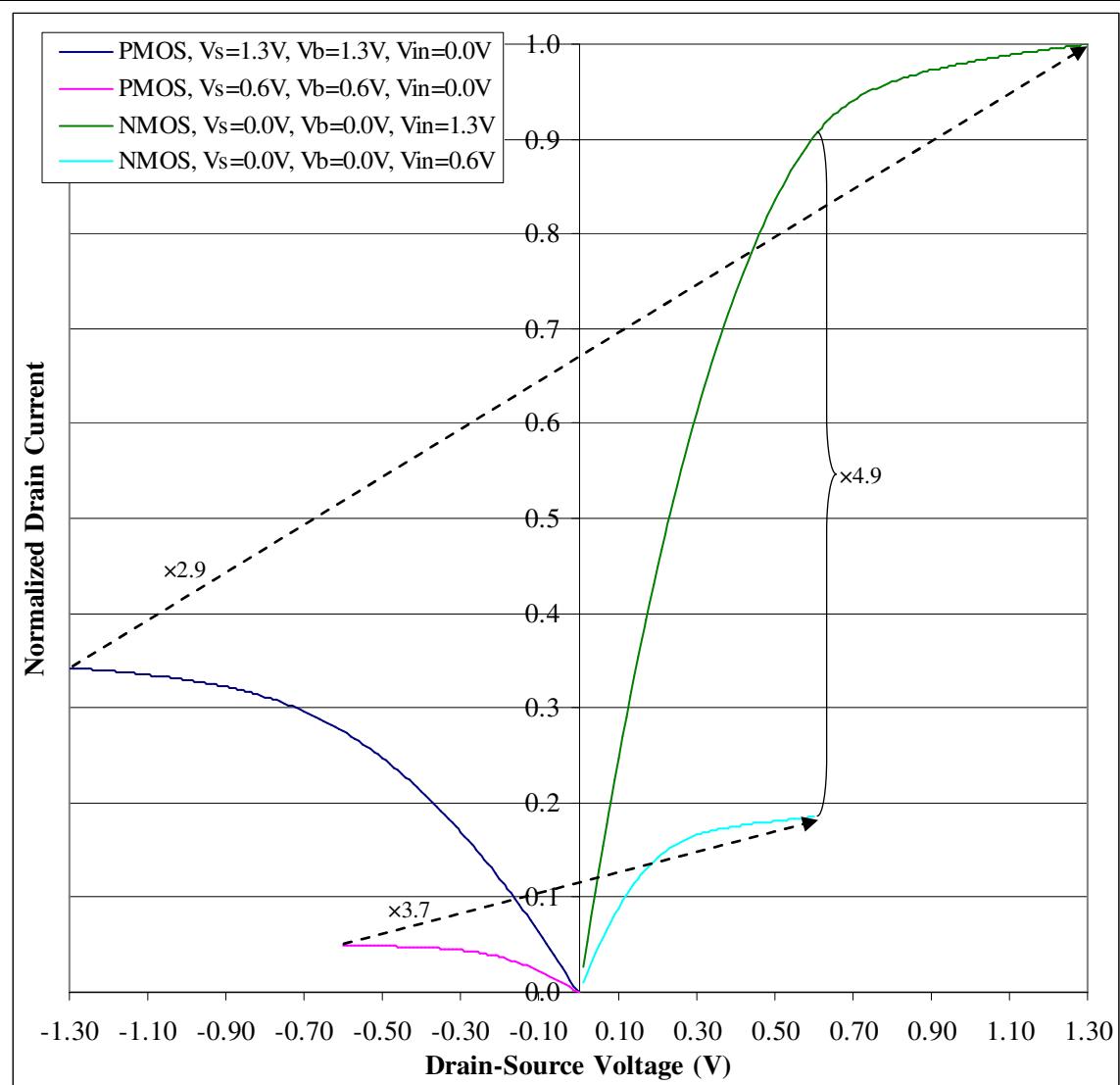


Figure 6.4 This graph shows the comparative drain currents of equally sized NMOS and PMOS transistors with 0.18um channel length and the nominal threshold voltage. Drain currents were normalized versus the maximum NMOS drain current with drain-source voltage of 1.3V. V_s is the source voltage, V_b is the substrate (bulk) voltage, and V_{in} is the transistor gate voltage. The NMOS drain current is 3.7× that of the PMOS drain current for a gate when Vdd is 0.6V, and 2.9× that of the PMOS drain current for a gate when Vdd is 1.3V. With drain-source voltage of 0.6V, the NMOS drain current is 4.9× larger with input voltage V_{in} to the transistor gate of 1.3V compare to V_{in} of 0.6V.

6.2.2 Posynomial Fits

Previous researchers have assumed that Vdd and Vth were fixed, and that transistor widths or gate sizes were the only optimization variables. The supply voltage and threshold voltages can be included as parameters in the posynomial models without substantially affecting the accuracy of the posynomial fits, as illustrated by the example in Appendix D.2.

To fit the data, posynomials summing three monomials were used. Adding more monomials did not generally improve the fit accuracy. The general form of the posynomial models for a gate k is

$$\sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} s_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (6.5)$$

$$\sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} s_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (6.6)$$

Except for the dynamic energy which has the form

$$E_{sw,k} = \frac{1}{2} \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} s_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} + \frac{1}{2} \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} s_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (6.7)$$

where α_{ij} and c_i are fitting constants for the posynomial expression for each type of logic gate; $C_{load,fall}$ and $C_{load,rise}$ are respectively the load capacitance for a falling or a rising output; and $s_{in,fall}$ and $s_{in,rise}$ are respectively the maximum fanin fall or rise slew. The other posynomial expressions are listed in Appendix D.3.

The gates in our library have negative polarity. For a negative polarity gate, the output rise slew and output rise delay are a function of the maximum input fall slew and the sum of the input capacitances of fanouts for a rising input. Whereas, the input capacitance for a rising input (i.e.

falling output) is a function of the maximum input rise slew and the sum of the input capacitances of fanouts for a falling input.

Note that input capacitance for a rising input and input capacitance for a falling input were quite close in value, but for accurate power analysis separate models were appropriate.

The dynamic energy E_{dyn} expression is different in order to avoid negative values. It is the sum of the energy for a 0→1 output transition and the energy for a 1→0 transition. The dynamic energy includes the energy for (dis)charging the gate's internal capacitances and load capacitance, and the short circuit current from supply to ground. Half the energy is attributed to the rising transition and half to the falling transition¹. For an input driver i with voltage swing from 0V to $V_{dd,i}$, the switching energy for charging and discharging the load capacitance is

$$E_{dyn,i} = \frac{1}{2} C_{load,fall,i} V_{dd,i}^2 + \frac{1}{2} C_{load,rise,i} V_{dd,i}^2 \quad (6.8)$$

The leakage power is an exponential function of the transistor threshold voltage. We use exponentials of the threshold voltage, as this improves modeling accuracy for the leakage power, and doesn't reduce modeling accuracy for other characteristics. In the geometric program, the leakage power is multiplied by the probability the output is high or low as appropriate.

The least squares fit with a monomial can be minimized optimally, as the logarithm is linear in coefficient and indices:

$$\log y_{fit} = c_1 + \sum_{j=1}^8 \alpha_{1j} \log x_j \quad (6.9)$$

where y_{fit} is the monomial fit for the data y_{data} . The least squares fit to $\log y_{data}$ minimizes the relative error for the N data points:

¹ As suggested by David Blaauw.

$$\sum_{i=1}^N (\log y_{fit} - \log y_{data})^2 = \sum_{i=1}^N \left(\log \frac{y_{fit}}{y_{data}} \right)^2 \quad (6.10)$$

In general, fitting a posynomial to the data is a nonconvex nonlinear problem. Thus solutions are liable to get stuck in local minima. In practice, the least squares fit for a single monomial provides a sufficiently good starting point for the posynomial to ensure a reasonably accurate posynomial fit to the data. For the two other monomials, we start with indices of 1 and -1. To fit the posynomials, we then use conjugate gradient descent¹ to minimize the least squares error between $\log(y_{data})$ and $\log(y_{fit})$. The Jacobian of the logarithm of the posynomial fit with respect to the fitting indices and coefficients is used to compute the gradient for steepest descent minimization. Note that the coefficients are squared to ensure that they are positive. The derivatives are determined as follows:

$$\log y_{fit} = \log \left(\sum_i c_i^2 \prod_{j=1}^8 x_j^{\alpha_{ij}} \right) = \log \left(\sum_i c_i^2 \prod_{j=1}^8 (e^{\alpha_{ij}})^{\log x_j} \right) \quad (6.11)$$

$$\frac{\partial \log y_{fit}}{\partial \alpha_{ij}} = \frac{(\log x_j) c_i^2 \prod_{j=1}^8 (e^{\alpha_{ij}})^{\log x_j}}{\log y_{fit}} \quad (6.12)$$

$$\frac{\partial \log y_{fit}}{\partial c_i} = \frac{2c_i \prod_{j=1}^8 (e^{\alpha_{ij}})^{\log x_j}}{\log y_{fit}} \quad (6.13)$$

¹ Conjugate gradient descent was performed with lsqcurvefit in MATLAB.

The circuit parameters of interest vary over a wide range depending on circuit conditions, so minimizing the relative error is essential. For example, gate delays may be as small as 20ps in a fast circuit configuration, and as slow as 400ps when driving a large load. Relative root mean square (RMS) error is often used to measure the accuracy of standard cell libraries [138], so we use this metric to analyze the accuracy of the posynomial fits. The relative RMS error is given by

$$\text{RMS error} = \sqrt{\sum_{i=1}^N \left(\frac{y_{\text{data},i} - y_{\text{fit},i}}{y_{\text{data},i}} \right)^2} / N \quad (6.14)$$

Relative root mean square errors for the posynomial models of the SPICE data are shown in Table 6.3. These accuracies are comparable to results reported elsewhere using posynomial models for transistor sizing alone [117][207]. Note that relative mean square error analysis tends to emphasize the larger errors when fitting small data values [138], while larger values can be fit more accurately. On a 700MHz Pentium III machine, about a day is required for the MATLAB scripts to fit posynomials to all the data.

Table 6.3 Relative root mean square error of posynomial fits for the SPICE data. When output low leakage is significant, $V_{dd,in}=V_{dd,gate}$, and the accuracy is within 10%.

	Symbol	Inverter	NAND2	NAND3	NAND4	NOR2	NOR3	NOR4
Rise delay	d_{rise}	7.6%	6.5%	5.7%	4.5%	5.2%	3.8%	3.7%
Fall delay	d_{fall}	6.9%	6.1%	5.1%	4.3%	5.2%	5.7%	4.6%
Rise slew	$s_{out,rise}$	7.4%	6.7%	5.9%	4.7%	6.6%	4.3%	2.4%
Fall slew	$s_{out,fall}$	6.8%	7.7%	7.8%	6.3%	6.1%	6.3%	4.7%
Input capacitance C_{in} for rising input	$C_{in,rise}$	1.3%	1.6%	1.2%	1.1%	1.5%	1.5%	1.5%
Input capacitance C_{in} for falling input	$C_{in,fall}$	1.2%	1.5%	1.3%	1.1%	1.3%	1.2%	1.1%
Dynamic switching energy and short circuit power	E_{sw}	6.1%	5.5%	5.8%	5.2%	3.2%	2.5%	2.1%
Leakage when pull-down network is off	$P_{leak,high}$	5.4%	10.2%	10.4%	8.3%	5.8%	5.4%	5.4%
Leakage when pull-up network is off	$P_{leak,low}$	34.0%	34.8%	31.7%	29.1%	4.6%	2.5%	1.8%

Modeling the leakage through the pull-up network when the output is low ($P_{leak,low}$) is particularly difficult when $V_{dd,in} > V_{dd,gate}$. The larger input voltage reverse biases the PMOS transistors, exponentially reducing the subthreshold leakage current. Conflictingly, as $V_{dd,in}$ increases, gate-induced drain leakage (GIDL) increases. The impact of GIDL is smaller, so overall there is a significant reduction in leakage¹. These conflicting effects are difficult to model accurately, leading to large RMS errors for $P_{leak,low}$ when $V_{dd,in} > V_{dd,gate}$. When $V_{dd,in} = V_{dd,gate}$, the relative RMS error for $P_{leak,low}$ is only about 10% for the inverter and NAND gates, which is precisely when leakage is larger and of significance to the total power consumption. With the small leakage currents, the SPICE simulations were not always sufficiently long for the leakage to stabilize due to voltage changing slowly on gate internal capacitances, which makes it more difficult to fit the leakage data. These issues do not apply to leakage through the pull-down network (when the output is high), as the input voltage is 0V to NMOS transistors that are off in the pull-down network. Also note that $V_{dd,in} = V_{dd,gate}$ for the majority of gates in a dual supply voltage circuit with clustered voltage scaling, as the transition from high supply voltage (VDDH) to low supply voltage (VDDL) can only occur once on a path through combinational logic. Thus the larger error for small PMOS leakage values when $V_{dd,in} > V_{dd,gate}$ has a negligible impact on the accuracy of the total circuit power.

Note that the SPICE models for gate characterization did not have transistor folding for wider transistors in larger gates. Instead, it was simply assumed that a standard cell could become increasingly wider to accommodate the wider transistors, rather than using transistor folding to compact the cell. Parasitic capacitances between gate nodes were determined by SPICE, rather than from a specific standard cell layout. Transistor folding and design rule limitations on transistor layouts result in dips and jumps in the value of the parasitic capacitances of SPICE netlists for the STMicroelectronics 0.13um HCMOS9D library. The resulting parasitic

¹ For example, an inverter with $V_{dd,gate}=1.125V$ has 64x larger leakage when $V_{dd,in}=1.125V$ versus when $V_{dd,in}=1.3V$.

capacitances are not piecewise convex. That is, there are dips and jumps in the data such that the capacitance can be double what would be expected from a convex fit at that point. For example for the NAND2 logic gate in Figure 6.5, the non-convex internal parasitic capacitances are shown versus gate size in Figure 6.6 with the number of transistor folds listed in Table 6.4. When the transistors are folded in two at gate size 4.0, the parasitic capacitances from pin A to GND and Vdd are double what might be expected. Including transistor folding and the irregular parasitic capacitances would result in the power and delay data for gates being more uneven and non-convex, and the convex models would have greater fitting error.

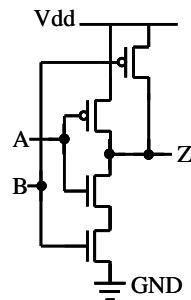


Figure 6.5 A transistor-level schematic showing the nodes (A, B, Z, Vdd, GND) between which there are parasitic capacitances.

Table 6.4 This table lists the number of NMOS and PMOS transistor folds versus the normalized gate size of NAND2 gates. As the number of transistor folds change, there are jumps and dips in the parasitic capacitances between gate nodes (see Figure 6.6) due to the change in layout.

Normalized Gate Size	Number of NMOS folds	Number of PMOS folds
1.0	1	1
2.0	1	1
4.0	2	2
6.0	3	3
8.0	4	3

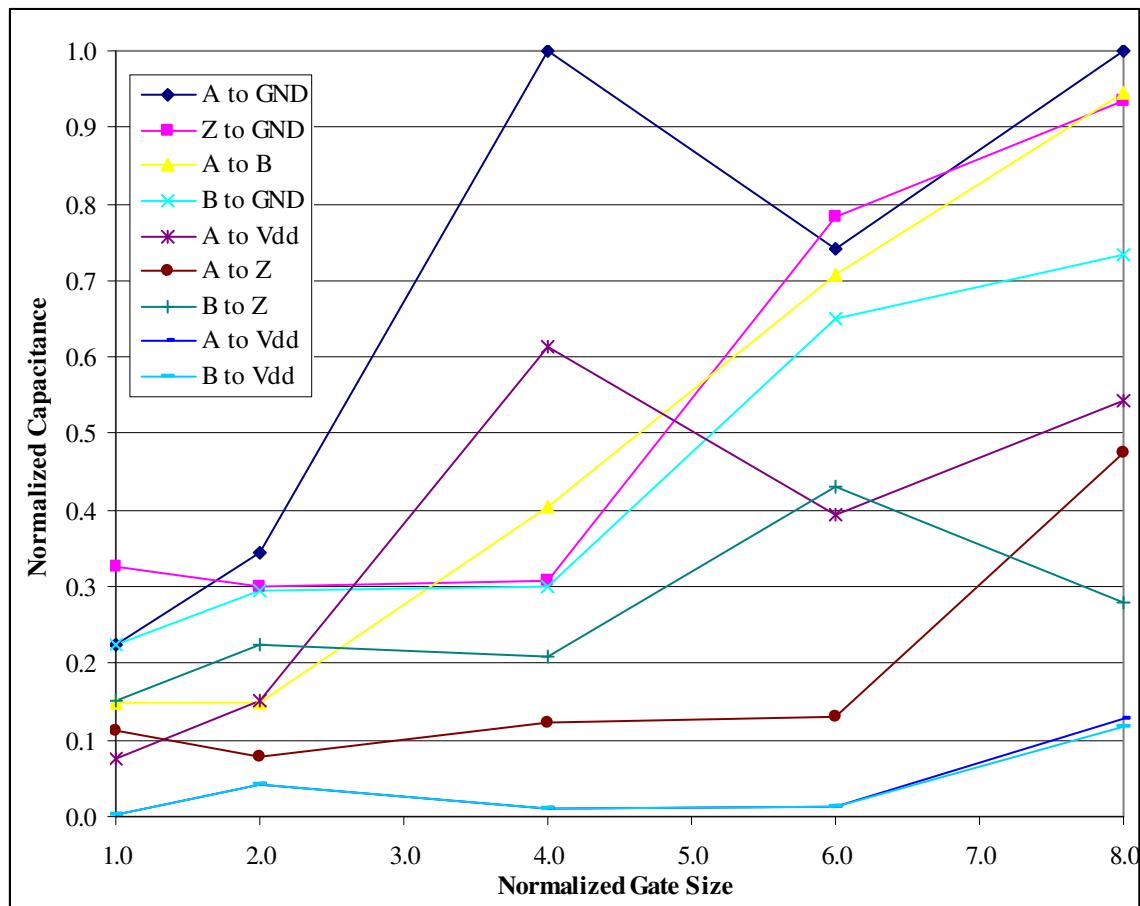


Figure 6.6 This graph shows the nine most significant parasitic capacitances between nodes in the NAND2 transistor-level SPICE netlists as gate size increases. Jumps and dips occur in the data as the standard cell layout changes when the number of transistor folds changes to accommodate larger transistors.

It is possible to achieve reasonably accurate posynomial fits with fewer parameters, as some characteristics are independent of some parameters. The number of parameters in the posynomials can be reduced by about 21% with at most 1% impact on the delay and power accuracy. This enables a 21% to 26% reduction in geometric program solver runtime as discussed in Appendix D.3.1.

We also examined whether the posynomial fits could be made conservative to ensure that delay and power were not underestimated. This results in fitting error of up to 23% for the inverter delay, so it doesn't make sense to use the conservative posynomial fits described in Appendix D.3.2. This limits us to using posynomial fits that may underestimate the delay and power in some cases.

The next section discusses the geometric program constraints with which we perform convex optimization using the posynomial fits with a reduced number of variables detailed in Appendix D.3.1.

6.3 Constraints

This section lists the constraints to formulate the geometric program with the posynomial models. The combinational circuit is represented as a directed graph $G(V,E)$, with directed edges in set E between vertices in set V . Each vertex represents a logic gate or an input driver. Specifying the constraints in a suitable form for the geometric program solver is detailed in Appendix D.4.

The independent variables to be optimized by the geometric program solver are the gate or input driver supply voltage $V_{dd,gate}$, and the NMOS and PMOS transistor threshold voltages and transistor widths for each gate. The range of these independent variables is limited by what is reasonable for the 0.13um process technology.

There are other dependent variables in the geometric program that are determined by edge constraints, or are characteristics of the logic gate (e.g. delay, output slew, power) for the given input slew, load capacitance, and independent variable values.

6.3.1 Process Technology Constraints

The process technology places upper and lower bounds on the gate supply and threshold voltages, as listed for the SPICE characterization range in Table 6.2. That is, the supply voltage for each logic gate or input driver was constrained in the range

$$0.6V \leq V_{DDL} \leq V_{dd, gate,v} \leq V_{DDH} \leq 1.3V \quad (6.15)$$

where the minimum and maximum gate voltages are V_{DDL} and V_{DDH} . The NMOS and PMOS threshold voltages for logic gates were constrained as

$$0.097V \leq V_{thn,min} \leq V_{thn,v} \leq V_{thn,max} \leq 0.357V \quad (6.16)$$

$$-0.338V \leq V_{thp,min} \leq V_{thp,v} \leq V_{thp,max} \leq -0.078V \quad (6.17)$$

As positive-valued variables are required for the geometric program, the absolute value of the PMOS threshold voltage and positive constraints are used in the geometric program, but we show the actual negative-valued range here. We actually used the exponential of the threshold voltage, that is $\exp(V_{thn})$ and $\exp(|V_{thp}|)$, in the posynomial models and geometric program to improve the accuracy of the posynomial fits to leakage.

With a continuous range of voltages, the minimum and maximum voltages V_{DDL} , V_{DDH} , $V_{thn,min}$, $V_{thp,min}$, $V_{thp,max}$ are simply used to inform the user of the lower and upper bounds on the voltage variables. Whereas when voltages have been discretized to give a “dual voltage” problem with these values, these values are global variables that may be optimized. In a single voltage problem, there is a single global V_{dd} , V_{thn} and V_{thp} that may be optimized.

Table 6.5 This table lists the maximum NMOS and PMOS width constraints in the geometric program. The minimum width was 0.26um for NMOS and PMOS for all gates types. Essentially for gates other than the inverter, the drive strength is up to 10 \times the drive strength of the minimum width (0.26um) NMOS transistor. For the inverter it is up to 20 \times , as inverters may be used for buffering.

Logic gate	Number of NMOS transistors in series	Number of PMOS transistors in series	Maximum NMOS width (um)	Maximum PMOS width (um)
Inverter	1	1	5.2	10.4
NAND2	2	1	5.2	5.2
NAND3	3	1	7.8	5.2
NAND4	4	1	10.4	5.2
NOR2	1	2	2.6	10.4
NOR3	1	3	2.6	15.6
NOR4	1	4	2.6	20.8

The minimum transistor width of 0.26um was also determined by the process technology.

Constraints on the maximum transistor widths are listed in Table 6.5. The maximum NMOS widths for the NOR gates are larger than the characterization range detailed in Appendix D.1, because the maximum NMOS widths for characterization were set erroneously low. Restricting to the smaller characterization range limited voltage scaling as detailed in our geometric programming paper [41]. The posynomial models for NAND gate characteristics will be less accurate when extrapolating beyond the characterized width range.

As NMOS transistors have about double the drain current of PMOS transistors of the same width, typical standard cell gates have PMOS to NMOS (P:N) width ratio of about 2:1 to balance rise and fall delays. However, skewed P:N ratios can be advantageous. To reduce power with smaller PMOS transistor capacitances, a ratio of as low as 1.5:1 may be better. Moreover, sometimes the rise and fall drive strengths needed are different. For example if a rising input signal to an inverter arrives much later than the falling input signal, then it is important to have a fast falling transition, but the rising output transition can be slower.

Fixing the P:N width ratio is suboptimal, particularly when multiple supply voltages are allowed, because $V_{dd,in} > V_{dd,gate}$ substantially reduces fall delay. Using skewed P:N width ratios can also be beneficial with mixed threshold voltages. For example, if a gate usually has a high output, that is the pull-down NMOS transistors are off most of the time, then higher NMOS threshold voltage

can substantially reduce leakage while NMOS transistor widths are increased to avoid increasing the delay. Thus when optimizing threshold voltages and gate size, we can expect a wide range of P:N width ratios. The transistor width ratio characterization range was from 4:1 to 3:4 of the equivalent inverter ratio. To reduce the number of constraints, gate width ratios were not constrained in the optimization. Solutions generally stayed within these bounds, except when $V_{dd,in}$ exceeded $V_{dd,gate}$ and smaller NMOS widths could be used, because the pull-down drive strength of the gate is much higher than the pull-up drive strength.

6.3.2 Graph Edge Constraints

To connect the nodes, we have the following edge constraints for worst case static timing analysis and for worst case power analysis. The posynomial models also have an implicit lower bound of zero ($p(\mathbf{x}) > 0$), so we do not need to impose constraints for the minimum capacitance or slew.

The load capacitance is the sum of the input capacitance of fanout gates plus the wire load, where the wire load was $3+2\times|fanout(v)|$ fF, where $|fanout(v)|$ is the number of fanouts of gate v .

$$C_{load,fall,v} = 3 + 2|fanout(v)| + \sum_{j \in fanout(v)} C_{in,fall,j} \quad (6.18)$$

$$C_{load,rise,v} = 3 + 2|fanout(v)| + \sum_{j \in fanout(v)} C_{in,rise,j} \quad (6.19)$$

The load capacitance of the combinational outputs was set to 3fF, and there is an additional 5fF wire load for connecting to the output. The wire load capacitance and output port load are the same as was used for the linear programming work in Chapter 4 and Chapter 7. To reduce runtimes, we did not set a constraint on the maximum load capacitance, as load capacitances only very rarely exceeded the maximum characterized load capacitance of 75fF.

The worst case input voltage for delay analysis is the minimum over fanin supply voltages for a rising input. The worst case input voltage for a falling input may be the maximum over fanin

supply voltages, as the input reaches 90% of $V_{dd,in}$ before 90% of $V_{dd,gate}$ if $V_{dd,in} > V_{dd,gate}$, as described in Appendix D.2. However, the maximum supply voltage is not always the worst case for a falling input, because the output slew is larger for a lower input voltage and $\partial d/\partial s_{in}$ is positive in most cases (see Table E.1). As the net impact of using $V_{dd,in}=V_{dd,gate}$ is pessimistic by only 1% to 3% as described at the end of Appendix B.5, using the minimum over fanin supply voltages is acceptable for worst case delay analysis for both rising and falling inputs. Using the minimum fanin supply voltage is also appropriate for worst case leakage power analysis, as the PMOS leakage is less when $V_{dd,in} > V_{dd,gate}$. Thus we have the constraint

$$V_{dd,in,v} = \min_{j \in \text{fanin}(v)} \{V_{dd,gate,j}\} \quad (6.20)$$

For worst case delay analysis as $\partial d/\partial s_{in}$ is generally positive, the worst case input slew is the maximum output slew of fanin nodes. Larger input slews are worse for short circuit power, because there is a longer period of time when there is a short circuit current path from supply to ground with both NMOS and PMOS transistors conducting. Thus the slew constraints for worst case power and delay analysis are

$$s_{in,fall,v} = \max_{j \in \text{fanin}(v)} \{s_{out,fall,j}\} \quad (6.21)$$

$$s_{in,rise,v} = \max_{j \in \text{fanin}(v)} \{s_{out,rise,j}\} \quad (6.22)$$

The rise and fall slew for input driver voltage ramps was set to 0.1ns, the same as was used for the linear programming work in Chapter 4 and Chapter 7. To limit the slew to the characterization range, we also restrict the maximum slew to 0.4ns:

$$s_{out,fall,v} \leq 0.4\text{ns} \quad (6.23)$$

$$s_{out,rise,v} \leq 0.4\text{ns} \quad (6.24)$$

The maximum rise and fall arrival times to a gate for worst case delay analysis are given by

$$t_{in,fall,v} = \max_{j \in fanin(v)} \{t_{out,fall,j}\} \quad (6.25)$$

$$t_{in,rise,v} = \max_{j \in fanin(v)} \{t_{out,rise,j}\} \quad (6.26)$$

Adding the gate delay, the rise and fall arrival times at the output of a negative polarity gate are given by

$$t_{out,rise,v} = t_{in,fall,v} + d_{rise,v} \quad (6.27)$$

$$t_{out,fall,v} = t_{in,rise,v} + d_{fall,v} \quad (6.28)$$

where $d_{rise,v}$ and $d_{fall,v}$ are gate v 's rise delay and fall delay respectively, as determined from the posynomial models. Note that wire delays were not included in static timing analysis, as they contributed only about 1% of the critical path delay for the linear programming results in Chapter 4 and Chapter 7.

We also have topological constraints on the supply voltage to avoid significant static current. We require that a gate has supply voltage less than or equal to the supply voltage of its fanins:

$$V_{dd,gate,v} \leq V_{dd,in,v} \quad (6.29)$$

This is necessary, because if a VDDL input drives a VDDH gate, the PMOS transistors are forward biased by $V_{DDL} - V_{DDH}$ which results in static current. Figure 6.7 illustrates this with two inverters. Thus it is important to have a voltage level converter to restore the input to full voltage swing. In this chapter, we assume a clustered voltage scaling scheme, where level converters appear only at the combinational outputs. Combining the level converters with the flip-flops reduces the power and delay overhead for voltage level restoration, as compared to inserting asynchronous level converters between combinational logic – see Section 7.2 for more detail.

We assumed no additional power or delay overhead for the level converter flip-flops, because these discrete jumps in power and delay for inserting a level converter cannot be handled in the geometric program as variables and posynomials must be continuous. The power and delay for a level converter flip-flop is comparable to a typical ASIC standard cell library D-type flip-flop, though it is slower than a high-speed pulsed flip-flop.

To include the level converter power and delay overheads, they would have to be approximated as a continuous function of $V_{dd,in} - V_{dd,gate}$, starting from no overhead if the voltages are the same. However, this is not realistic, because if the voltage difference is small it may be possible to avoid inserting a level converter and just accept the higher static power, whereas at some point there is a stepwise (delay if not power) discontinuity when it is essential to include a level converter to reduce the static power.

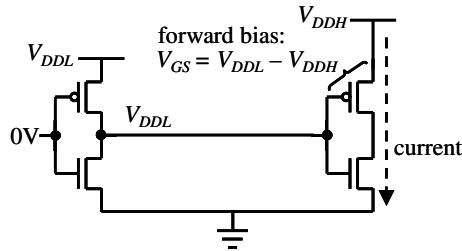


Figure 6.7 This diagram illustrates the need for voltage level restoration. The schematic shows an inverter with supply voltage of V_{DDH} , being driven by an inverter of supply voltage V_{DDL} . When the V_{DDL} inverter output is high, the driving voltage is only V_{DDL} , which results in a forward bias of $V_{DDL} - V_{DDH}$ across the PMOS transistor of the V_{DDH} inverter. The forward-biased PMOS transistor is not completely “off” resulting in large static currents.

6.3.3 Maximum delay constraint and total power

Considering only combinational delays, the minimum clock period T is the maximum output arrival time from the combinational outputs

$$T = \max_{v \in \text{combinational_outputs}} \{t_{out,fall,v}, t_{out,rise,v}\} \quad (6.30)$$

The total power consumption P_{total} for the circuit is the sum of dynamic power and leakage power

$$P_{total} = \left(\frac{1}{2T} \sum_{v \in V} \alpha_{sw,v} E_{dyn,v} \right) + \left(\sum_{v \in V} (1 - \alpha_{sw,v}) (\Pr(v=1) P_{leak,high,v} + \Pr(v=0) P_{leak,low,v}) \right) \quad (6.31)$$

where $\alpha_{sw,v}$ is the switching activity of gate v , that is the probability of a $0 \rightarrow 1$ or $1 \rightarrow 0$ transition at the output of v ; $\Pr(v=1)$ and $\Pr(v=0)$ are the probabilities that the output of gate v is high or low respectively; and only load switching power is included for input drivers. Activities are calculated from random simulation with independent inputs with equal probabilities of 0 or 1, and the switching activity of each input is 0.5. We assume that any leakage that occurs in a cycle when the gate switches, before or after the transition, is included in the dynamic energy.

The optimization problems of interest to us are minimizing the clock period T ; and minimizing the total power P_{total} subject to a constraint on the maximum delay T . We can also minimize the energy-delay product $P_{total}T^2$ and similar metrics if so desired. These can be solved to find the global minimum using geometric programming with the posynomial models and given constraints.

The geometric program has continuous variables. However, typical circuits have only single or dual supply and threshold voltages. For dual supply and threshold voltages, we need a method to pick discrete values. The next section discusses a heuristic to find a dual supply voltage and dual threshold voltage solution near the global minimum provided by the posynomial formulation. The discretization approach ensures the iterative solution is within the feasible solution space, preserving the topological supply voltage constraints.

6.4 Issues with Discrete Voltages

The geometric programming formulation using convex models assumes that the variables are continuous. We will refer to this solution as the “continuous voltage” solution. Typical circuits have only single or dual supply and threshold voltages, due to the additional fabrication costs. Each additional threshold voltage requires an additional mask. Each additional supply voltage

requires a voltage regulator and the increased wiring congestion and reduction in cell density increases the layout area. Thus a reasonable limit is at most two supply voltages, two NMOS threshold voltages, and two PMOS threshold voltages – the “dual voltage” solution. We must develop a discretization heuristic to find a dual voltage solution, given a continuous voltage solution.

The width W of a transistor in a cell may be continuous in “liquid cell” sizing methodologies, but usually a standard cell library with has logic gates characterized for discrete widths. We allow a continuous range of transistor widths, assuming a liquid cell methodology or a library with a very fine granularity of gate sizes.

The solution with all variables continuous provides a lower bound on the minimum delay, or minimum power for a delay constraint, that may be achieved with discrete voltages. If we limit the supply voltage, NMOS threshold voltage, and PMOS threshold voltage to being some global value for each, then we have a “single voltage” solution. In the single voltage solution, the global values for the supply voltage and the threshold voltages can still be chosen from within the allowed range for the process technology. The single voltage solution provides an upper bound on what can be achieved using multiple discrete voltages. The single supply voltage, single NMOS threshold voltage, and single PMOS threshold voltage problem can be solved optimally by the geometric program solver. The single voltage upper bound is realizable in today’s process technologies, whereas the continuous voltage lower bound is not.

Given voltage assignments to high and low V_{dd} et al., we can optimize these global variables and perform gate sizing for the circuit. This gives a globally optimal solution for the given heuristic choice of supply voltage and threshold voltage assignments.

6.4.1 A Discretization Heuristic

Assuming that the continuous voltage solution gives a fixed ordering to assign gates from high to low voltage and assuming that the power is convex versus the number of gates assigned to a low voltage, the optimal number of gates to assign low can be determined by repeatedly bisecting the interval¹ to find where the minimum occurs. These assumptions are not always true, but this discretization heuristic does find a dual voltage solution close to the continuous lower bound.

The procedure for supply voltage discretization is shown in Figure 6.8. The first two steps establish a single voltage upper bound on the power consumption, by assigning all gates to V_{DDH} or V_{DDL} . The third step is to perform continuous voltage optimization, which provides a lower bound on the dual voltage power consumption, and serves as a guide for which gates to assign to V_{DDH} or V_{DDL} .

When assigning supply voltage high or low, we must try to avoid violating the delay constraints and topological constraints on Vdd. Given a continuous solution, a gate j has voltage $V_{dd,gate,j}$ between V_{DDH} and V_{DDL} . If we increase $V_{dd,gate,j}$ to V_{DDH} , it will be faster, but it may violate topological constraints. Alternatively, we could reduce the gate voltage to V_{DDL} , but then delay constraints may be violated. The gates with the highest supply voltage in the continuous solution are most in need of higher Vdd to meet delay constraints, or gates in their transitive fanout need high Vdd to meet delay constraints. Thus we order candidate assignments to V_{DDH} by value from the continuous solution and secondly by topological order if the supply voltage values are the same. This ensures that a gate can only be assigned to V_{DDH} after its inputs have been assigned to V_{DDH} , as the continuous solution also has the constraint $V_{dd,in} > V_{dd,gate}$.

NMOS threshold voltage and PMOS threshold voltage candidates can be ordered in a similar manner, but they have no topological constraints. We refer to an assignment that will reduce the gate delay, trying to avoid violating delay constraints, as a *conservative assignment*. For $V_{dd,gate}$ a

¹ The interval from no gates assigned low to all gates assigned low.

conservative assignment is to high supply voltage V_{DDH} , whereas for threshold voltages it is conservative to assign them to low threshold voltage. As was described in Section 5.2.2, the input capacitance increases a little with increasing supply voltage and decreasing threshold voltage, so while these assignments do reduce the delay of the gate, they can increase the loading on fanins and in rare cases can cause a delay constraint violation.

Having assigned gates to V_{DDH} or V_{DDL} , we can perform power minimization optimizing the other variables. In particular, we can optimize the global variables V_{DDH} and V_{DDL} , if these are not fixed by the designer or process technology. Until we have examined the power consumption for a varying number of gates being assigned to V_{DDH} or V_{DDL} , we don't know what the optimal number of gates to assign to V_{DDH} and V_{DDL} is. If Vdd of a gate is optimized to V_{DDH} or V_{DDL} in the continuous solution, then it should be assigned to that value in the discrete solution, but for intermediate values it is not clear. For example, if we make some gates slower by assigning them to V_{DDL} , we may need to assign more gates to V_{DDH} to ensure delay constraints are met. After the continuous voltage run, V_{dd} of gates or input drivers within 1% of V_{DDH} or V_{DDL} in the continuous solution are assigned to the appropriate global variable.

Discretization then proceeds by bisecting the interval of intermediate values. This is illustrated in Figure 6.9. Our upper two bounds have 0% and 100% of gates at V_{DDH} . If all gates have intermediate values that need to be discretized, then the fourth, fifth, and sixth steps in Figure 6.8 perform optimization with 25%, 50% and 75% of gates at V_{DDH} . In the seventh step, we choose the best interval, say where 75% at V_{DDH} is the best result so far and the neighboring points are 50% and 100%. Then we perform bisection again, getting results with 62.5% and 87.5% of gates set to V_{DDH} . Bisection proceeds in this manner, taking $O(\log_2 N)$ iterations to converge to a solution. Finding the optimal solution using bisection assumes that there are no local minima of power versus the number of gates assigned conservatively, which may not be the case. For example, for NMOS threshold voltage discretization for c432nr at a delay constraint of 2.12ns,

there was a local minimum with 97 gates assigned to $V_{thn,min}$, but the best solution found had 154 gates at $V_{thn,min}$ (see Figure D.7). Thus this procedure is heuristic and does not guarantee an optimal solution.

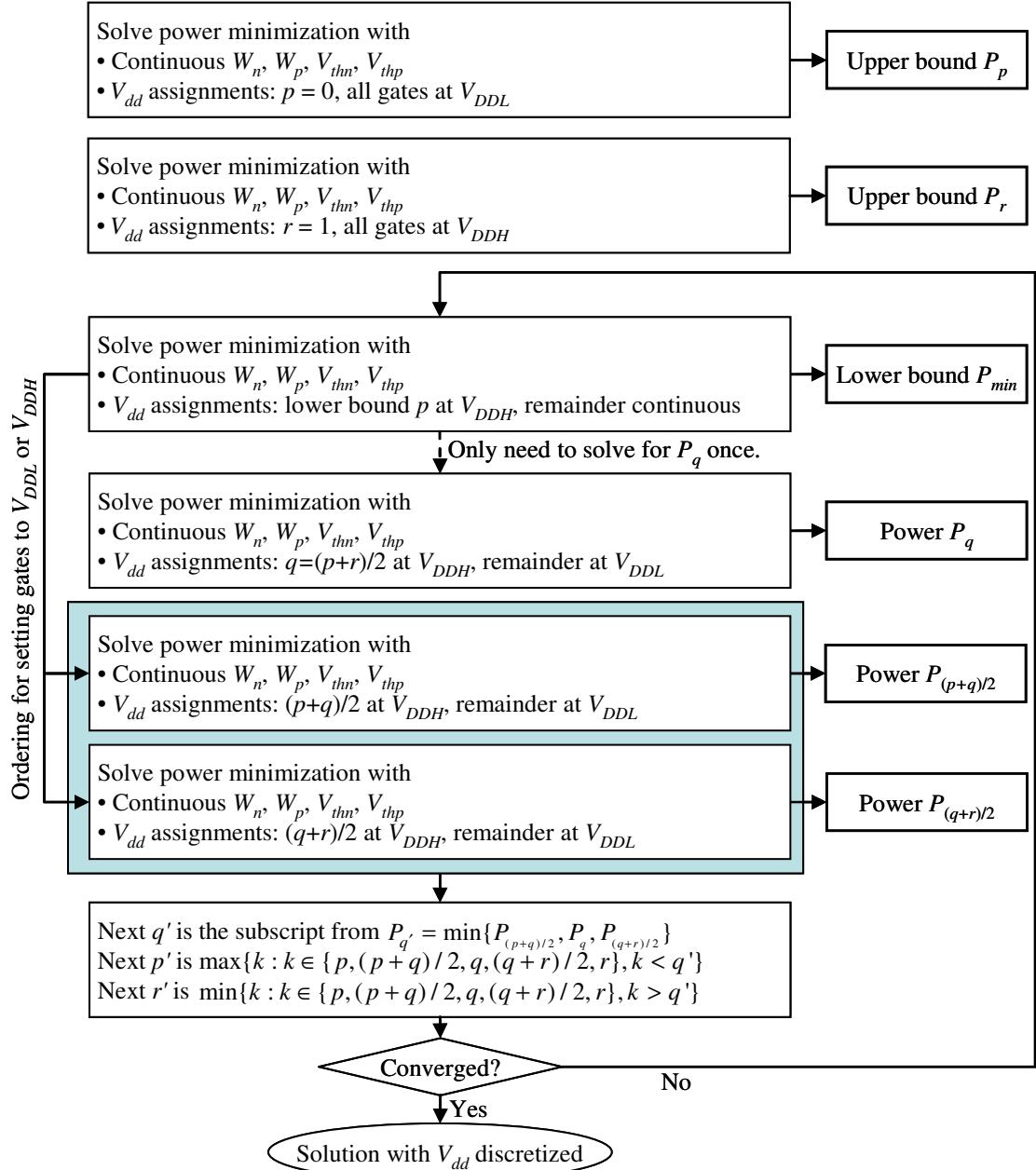


Figure 6.8 This flow chart shows the discretization procedure for V_{dd} . Note that the discrete global variables, V_{DDL} and V_{DDH} , can be continuously optimized or fixed as chosen by the user. If the problem is infeasible with a particular set of assignments, for example with V_{dd} of all gates and input drivers assigned to V_{DDL} , then the power is returned as $+\infty$ to ensure this solution is not used but the iterative method can proceed. The ordering for assigning V_{dd} to V_{DDH} and V_{DDL} is from the value order (largest to least) from the continuous V_{dd} solution (third step down), and any gates at the same V_{dd} are ordered topologically to preserve the topological constraint $V_{dd,in} \geq V_{dd,gate}$, which is not necessary for V_{thn} and V_{thp} discretization. The continuous V_{dd} optimization run on the first iteration with no gates pre-assigned to V_{DDH} may have a solution with V_{dd} of some gates at V_{DDH} and some at V_{DDL} – if so, V_{dd} of those gates is fixed to V_{DDH} or V_{DDL} , and discretization is performed only on the remaining gates with intermediate V_{dd} values. p , q , r , and intermediate fractions represent the fraction of gates assigned to V_{DDH} .

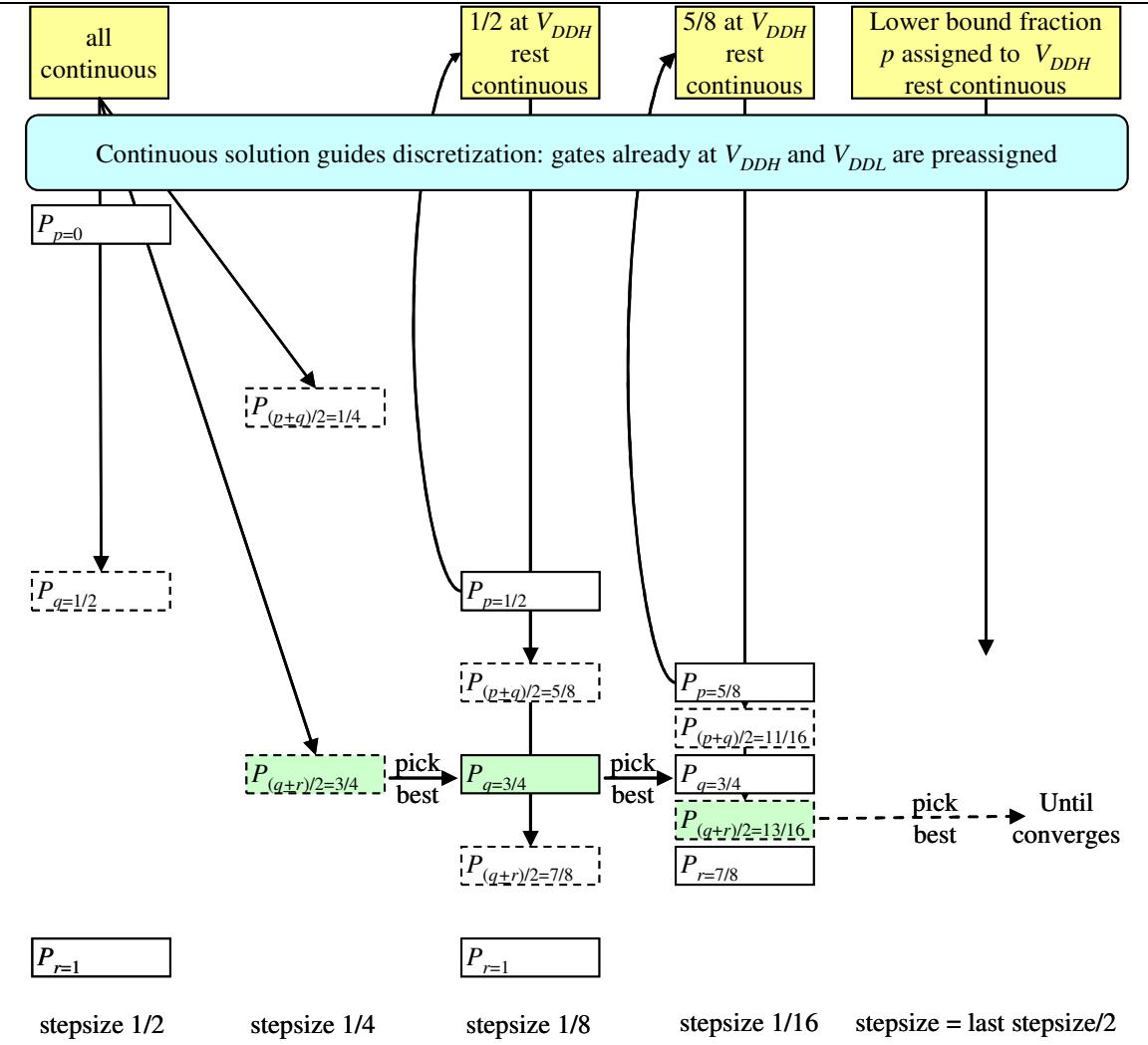


Figure 6.9 This diagram illustrates how the Vdd discretization procedure in Figure 6.8 proceeds. The continuous optimization on each iteration is shown in yellow, and the best discrete result on each iteration is shown in green. The new runs on each iteration are shown with dashed boxes. P_f is the power with fraction f of the gates and input drivers with intermediate $V_{dd,gate}$ values assigned to V_{DDH} and the rest assigned to V_{DDL} . This bisection method takes $O(\log_2 N)$ steps, where N is the number of intermediate $V_{dd,gate}$ values to be discretized. Note that this diagram does not show the impact on the fraction values of gates being pre-assigned to V_{DDH} or V_{DDL} from the continuous solution results – see Equation (6.32) for that detail.

While the discretization approach is heuristic, we can measure how far it is from the continuous voltage lower bound. This provides a measure of how good the discrete solution is. For example for c432nr at a delay constraint of 2.12ns, the dual voltage discretization result of 902.24uW is only 1.6% worse than the continuous voltage solution of 888.14uW. The discretized dual voltage result is surprising close to the solution which allowed a continuous range of voltages. Indeed, we find this to be the case for the benchmarks over a range of delay constraints, because optimization

of the global variables (V_{DDH} , V_{DDL} , $V_{thn,max}$, $V_{thn,min}$, $V_{thp,max}$, $V_{thp,min}$), NMOS widths, and PMOS widths allows sufficient flexibility for a discrete voltage solution to get close to the continuous voltage solution. For example, the worst power for discretization assignments tried at a delay constraint of 2.12ns is only 2.2% higher power than the continuous solution. See Appendix D.5 for more details of voltage discretization for c432 at a delay constraint of 2.12ns.

As more gates are assigned conservatively, we establish a lower bound to the number of gates that are assigned to V_{DDH} . We can perform continuous optimization again, with this lower bound fraction p of gates set to V_{DDH} and the rest variable. This gives us an updated lower bound for the power consumption of the continuous voltage solution with some gates assigned conservatively. If the discrete solution is sufficiently close to this lower bound, we can terminate discretization. Thus convergence was determined by the discrete solution being within 0.1% of the continuous lower bound, or when assigning one more or one less gate conservatively from the optimal solution had been tried (i.e. step size less than one gate).

When the updated continuous result assigns more gates to V_{DDH} and V_{DDL} , we need to rescale the fractions p , q , and r in the discretization procedure in Figure 6.8. Let h be the fraction of gates and input drivers assigned to V_{DDH} in the continuous solution, and let l be the fraction of gates and input drivers assigned to V_{DDL} in the continuous solution. h' and l' denote the respective fractions for the updated continuous solution. Then to rescale a fraction f of additional gates to assign to V_{DDH} , f' is given by

$$f' = \frac{f(1-h-l) + h - h'}{(1-h'-l')} \quad (6.32)$$

Here $(1 - h - l)$ is the fraction of unassigned gates in the original continuous solution, and $(1 - h' - l')$ is the fraction of unassigned gates in the updated continuous solution.

As gate supply voltage is subject to topological constraints, we discretize supply voltage before discretizing threshold voltages. A larger feasible solution space remains for threshold voltage. We discretize NMOS threshold voltage before PMOS threshold voltage as NMOS leakage was larger than PMOS leakage for our benchmarks. Thus discretizing PMOS threshold voltage last should have less impact on power. We could discretize all the variables simultaneously, but geometric programming is slow, so we want to minimize the number of additional runs required. Discretizing k variables for N gates simultaneously would take $O((\log_2 N)^k)$ geometric programming runs, whereas discretizing them in order requires only $O(k \log_2 N)$ runs. Likewise, discretizing a single variable to k global values would take $O((\log_2 N)^k)$ runs.

Feedback on this work pointed out that this binary search method is similar to Fibonacci search [63], but slower. Essentially, in the worst case the Fibonacci search splits the largest interval by fraction $1/\varphi = 0.618$, where φ is the Golden ratio. The number of geometric programming runs with discrete assignments for our approach is $2\log_2 N$. Whereas, Fibonacci search would take at most $\log_\varphi N$ iterations and on average $\log_2 N$ iterations [63]. Thus our method takes about twice the number of discretized geometric programming runs. However, the Fibonacci search method will be less robust in cases where the distribution is non-convex and there is more than one minimum – the additional sample points for our approach make it a bit more likely to avoid local minima.

Table 6.6 Summary of the ISCAS’85 benchmarks used with geometric programming optimization in this chapter. The benchmarks were mapped to NAND2, NAND3, NAND4, NOR2, NOR3, NOR4 and inverter logic gates. The mapping for c17 was six NAND2 gates, as illustrated in Figure 6.3.

Circuit	Number of inputs	Number of outputs	Number of gates	Number of logic levels	Average input and gate switching activity	Average fraction of cycles that gate outputs are high
c17	5	2	6	3	0.470	0.646
c432nr	36	7	222	24	0.377	0.580
c499	41	32	580	27	0.324	0.653
c880	60	26	555	29	0.329	0.537

6.5 Results

The MOSEK software package [146] was used to the geometric optimization problems, which were posed via scripts in MATLAB [141]. We will briefly discuss the circuit benchmarks for optimization then look at the results.

6.5.1 Benchmarks

Due to the long geometric programming runtimes, we only used the smallest circuits from the combinational ISCAS’85 benchmark set [25]. The ISCAS’85 benchmarks used in this chapter were c17, c432nr (c432 with logical redundancies removed, reducing the number of logic gates by three), c499 and c880.

A brief summary of the benchmarks mapped to our library is provided in Table 6.6. The number of logic levels is the maximum number of gates on any path from an input to a combinational output. The switching activity is the fraction of cycles on which an input on gate output changes from 0 to 1 or 1 to 0. The dynamic power of a gate is linearly proportional to the switching activity, α_{sw} in Equation (6.31). The fraction of time when a gate output is high, $\Pr(v = 1)$ in Equation (6.31), determines the proportion of leakage power contributed by leakage through the NMOS and PMOS transistors. If the gate output is high, that is the NMOS transistors are “off”, then the leakage is through the NMOS transistors; when the gate output is low, the leakage is through the PMOS transistors.

The switching probabilities and gate input state probabilities for leakage were determined by gate level logic simulation using a short Matlab script. In general, it would be preferable to have specific simulation benchmarks to stimulate a netlist, for example to avoid toggling a reset signal. Unfortunately, such vector sequences are not available for the commonly used benchmarks (i.e. ISCAS'85 and others). We assumed independent random primary inputs, with equal probabilities of 0 or 1. This seems a reasonable assumption for the combinational benchmarks that we used, as they don't have sequential elements that require a reset signal.

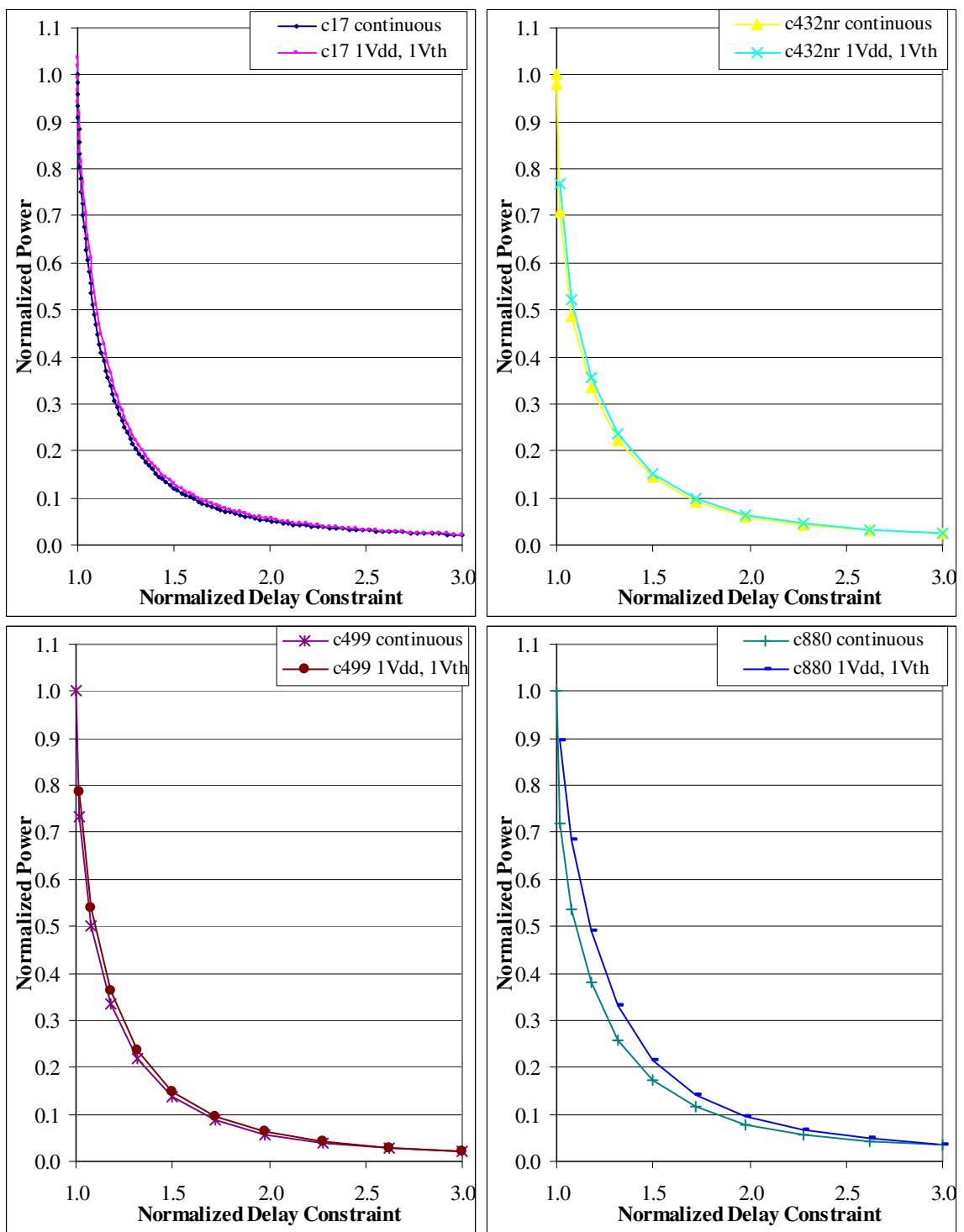


Figure 6.10 These graphs show the power versus the delay constraint for the single voltage and continuous voltage solutions for the different benchmarks. Power is normalized to the power at the minimum delay with continuous voltages. Delay is to the minimum delay achievable with continuous voltages. The power versus delay curve for c880 is not as steep, and there is a much larger gap between the single voltage upper bound and the continuous voltage lower bound.

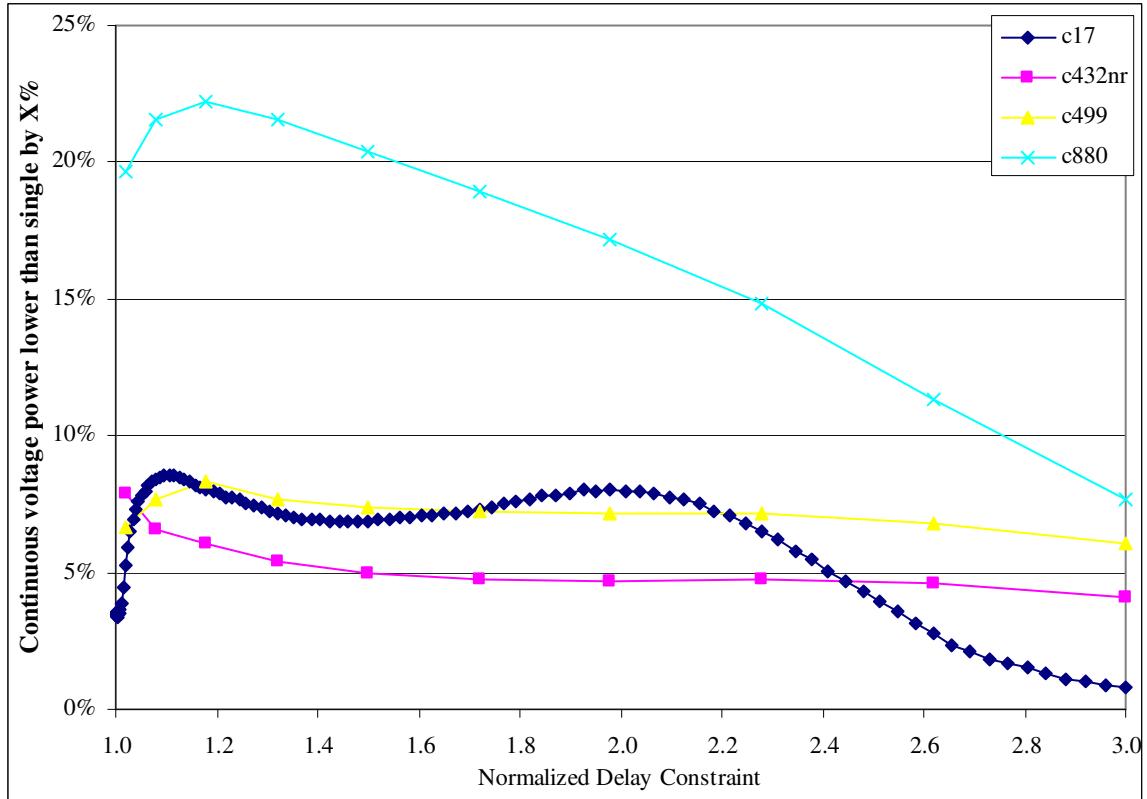


Figure 6.11 This graph shows how much lower power the continuous voltage lower bound is than the single voltage upper bound for the normalized data shown in Figure 6.10.

Table 6.7 This table shows that the minimum delay achievable using a single supply voltage, single NMOS threshold voltage and single PMOS threshold voltage (single voltage) is very close to that when voltages are assigned from a continuous range at the gate level (continuous voltage). Dual voltage discretization was not run for c499 and c880 due to the long run times.

Circuit	Minimum delay (ns) with			Dual voltage delay worse by	Single voltage delay worse by
	continuous voltage	dual voltage	single voltage		
c17	0.1629	0.1629	0.1629	0.00%	0.00%
c432nr	1.7953	1.7960	1.8018	0.04%	0.36%
c499	1.7923	did not run	1.7935	n/a	0.07%
c880	1.6523	did not run	1.6572	n/a	0.30%

6.5.2 Single voltage, dual voltage, and continuous voltage solutions

Performing optimization with a continuous range for supply and threshold voltages provides a lower bound on what can be achieved by discrete assignment of voltages between two possible choices, dual voltage solutions. Performing optimization with single supply voltage, single NMOS threshold voltage (V_{thn}), and single PMOS threshold voltage (V_{thp}) values that can be optimized at a global circuit level provides an upper bound for dual voltage solutions.

The minimum delay achieved for the single voltage solution is very close to the minimum delay possible with gate-level assignment of supply and threshold voltages from a continuous range, as shown in Table 6.7. With dual supply and dual threshold voltage values, there is a slight improvement over the single voltage minimum delay in some cases, bounded below by the continuous voltage minimum delay. We will not dwell further on delay minimization, as using multiple supply voltages or multiple threshold voltages provides no significant advantage in terms of delay versus the single voltage solution.

The power versus delay curves are quite similar for the different benchmarks, as shown in Figure 6.10. The gap between the single voltage upper bound and the continuous voltage lower bound is less than 9% for benchmarks c17, c432nr and c499 – see Figure 6.11. In contrast, the power versus delay curve for c880 is not as steep and the continuous voltage power is as much as 22.2% lower than the single voltage power.

We will examine the results for c17 and c432nr in detail, looking at the gap between the single voltage upper bound and the continuous voltage lower bound, and how much the dual voltage results can improve on the single voltage results. The results for c432nr are similar to those for c17. However, c432nr is a substantially larger benchmark, composed of more than just NAND2 gates, so it is interesting to also examine the c432nr results.

We will also examine how much higher the power is when voltage cannot be scaled optimally for the design by assuming that the supply and threshold voltages are fixed. To meet the minimum delay achieved by the continuous solution, the high supply voltage is the maximum supply voltage of 1.3V and low V_{th} is the minimum threshold voltage, 0.10V for V_{thn} and 0.08V for V_{thp}. The low supply voltage of 0.6V and nominal threshold voltage, 0.23V for V_{thn} and 0.21V for V_{thp}, were chosen to minimize power at large delay constraints.

We then take a quick look at the impact of not including wire loads in analysis, which is a common simplifying assumption used by researchers, though it gives quite inaccurate results.

The continuous and single voltage results for c499 are similar to those for c17 and c432nr. For c499, the continuous voltage power is at most 8.3% less than the single voltage power, so dual voltages may provide at most 8.3% benefit. Given the minimal benefits possible with dual voltages, we have not performed discretization for c499 due to the long run times. The gap between the single voltage and continuous voltage power for c880 is substantially larger, so we will examine the c880 results. More result details are provided in Appendix D.6.

6.5.2.1 Results for c17 and c432nr with voltage scaling

For c17 and c432nr the continuous voltage power is respectively at most 8.6% and 7.9% lower than the single voltage power. This leaves little room for improvement using dual supply voltages or dual threshold voltages, as shown in Figure 6.12 and Figure 6.13.

For c17, dual Vdd/single Vthn/single Vthp and single Vdd/dual Vthn/dual Vthp provide at most 3.5% power savings versus the single voltage solution. Using dual threshold voltages and dual supply voltages provides at most 5.5% power reduction versus the single voltage solution.

For c432nr, using dual Vdd/single Vthn/single Vthp provides at most 0.5% power savings versus the single voltage solution, as most gates get assigned to VDDH. Whereas, single Vdd/dual Vthn/dual Vthp provides up to 7.2% power savings versus the single voltage solution. Using dual threshold voltages and dual supply voltages provides at most 7.4% power reduction versus the single voltage solution.

These power savings are too small to justify use of dual Vdd or dual Vth.

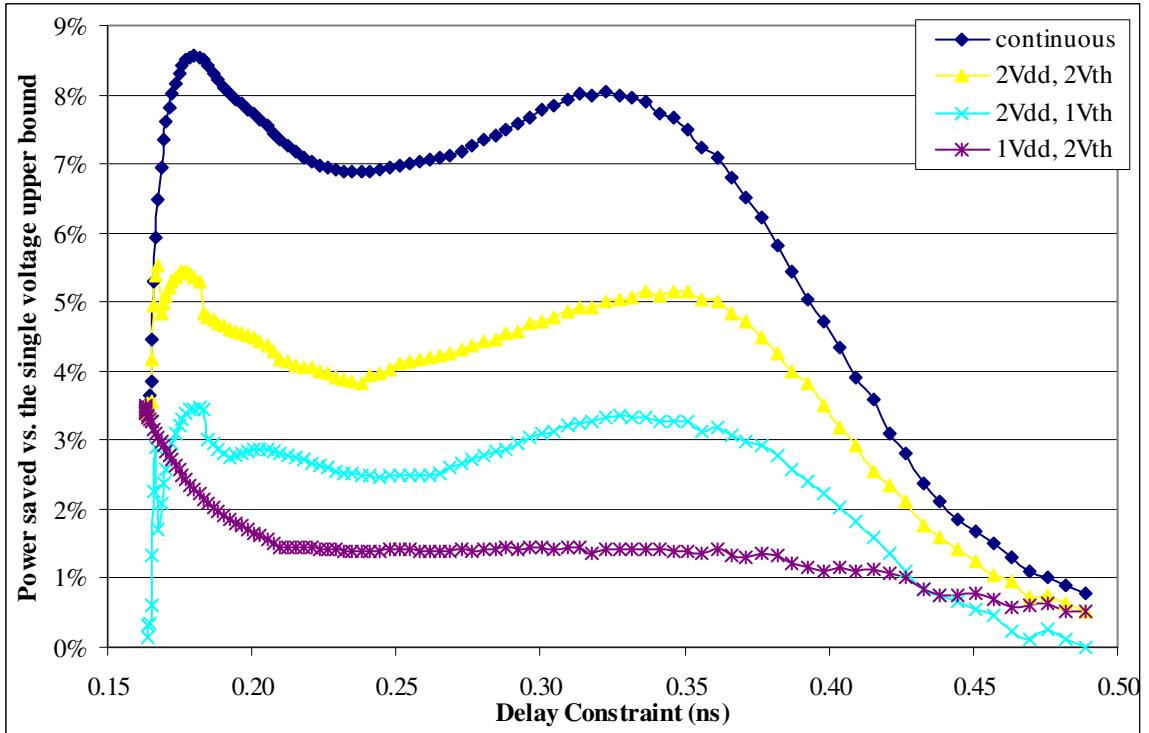


Figure 6.12 This graph shows how much power the dual voltage and continuous voltage solutions save versus the single voltage upper bound for c17.

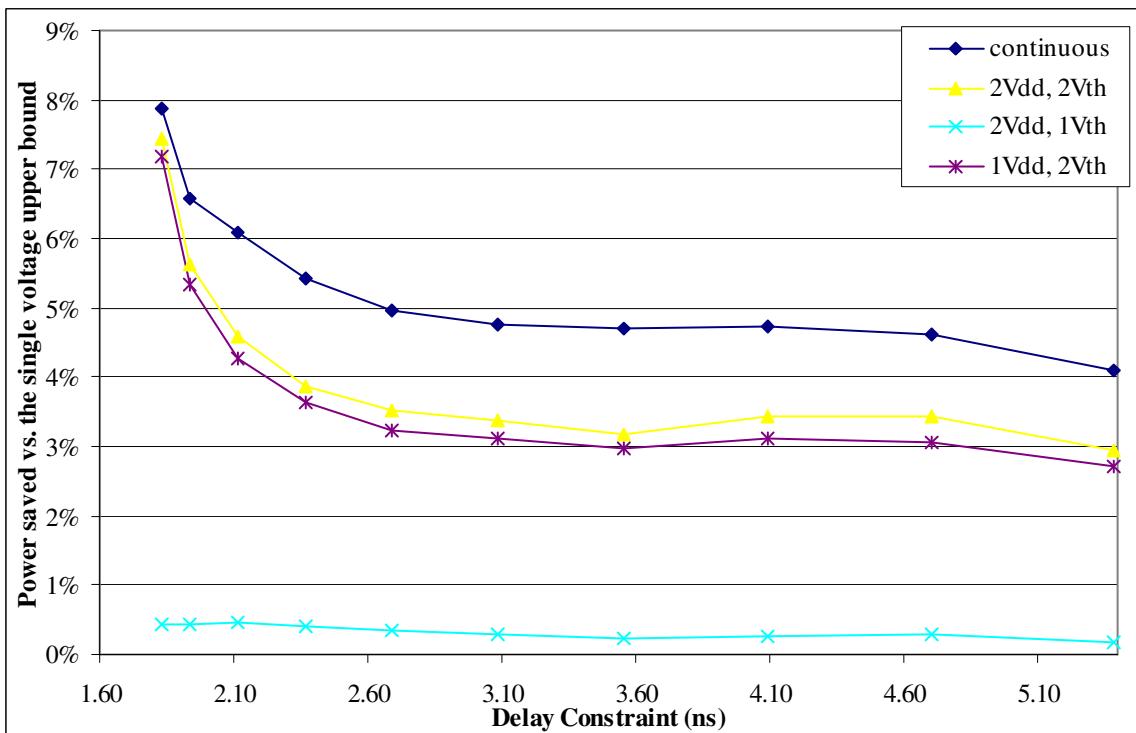


Figure 6.13 This graph shows how much power the dual voltage and continuous voltage solutions save versus the single voltage upper bound for c432nr.

6.5.2.2 Results for c17 and c432nr with fixed global voltage values

If the global voltage values cannot be optimized, then the situation changes. We see that a poor choice of global voltage values can result in 2 \times the power compared to the continuous solution, as shown in Figure 6.14 and Figure 6.15. While even using dual supply and dual threshold voltages with fixed global voltage values can result in about 50% worse power for both c17 and c432nr at relaxed delay constraints: at a delay constraint of $1.77 \times T_{\min} = 0.288\text{ns}$ for c17, and at a delay constraint of $1.98 \times T_{\min} = 3.55\text{ns}$ for c432nr. This illustrates the importance of optimizing Vdd and Vth for a design for a given performance constraint. See Appendix D.6.3 for power versus delay curves with fixed global voltage values.

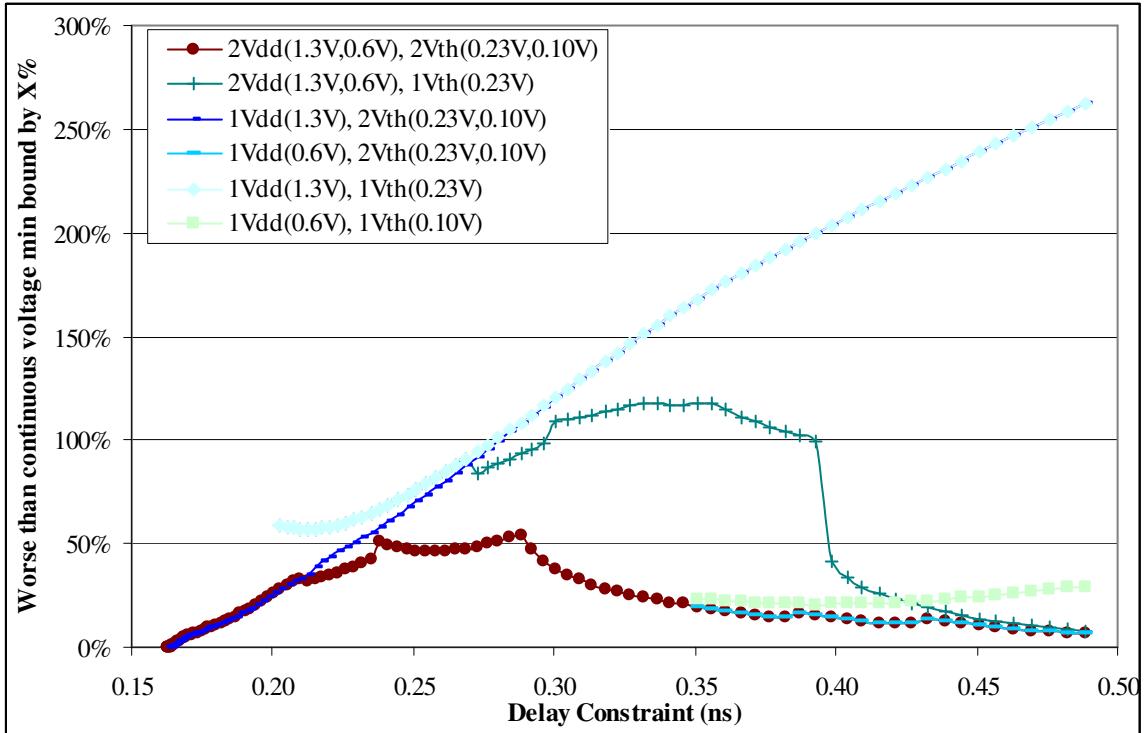


Figure 6.14 This graph shows how much worse the dual voltage and single voltage results with fixed global voltage values for c17 are relative to the continuous lower bound.

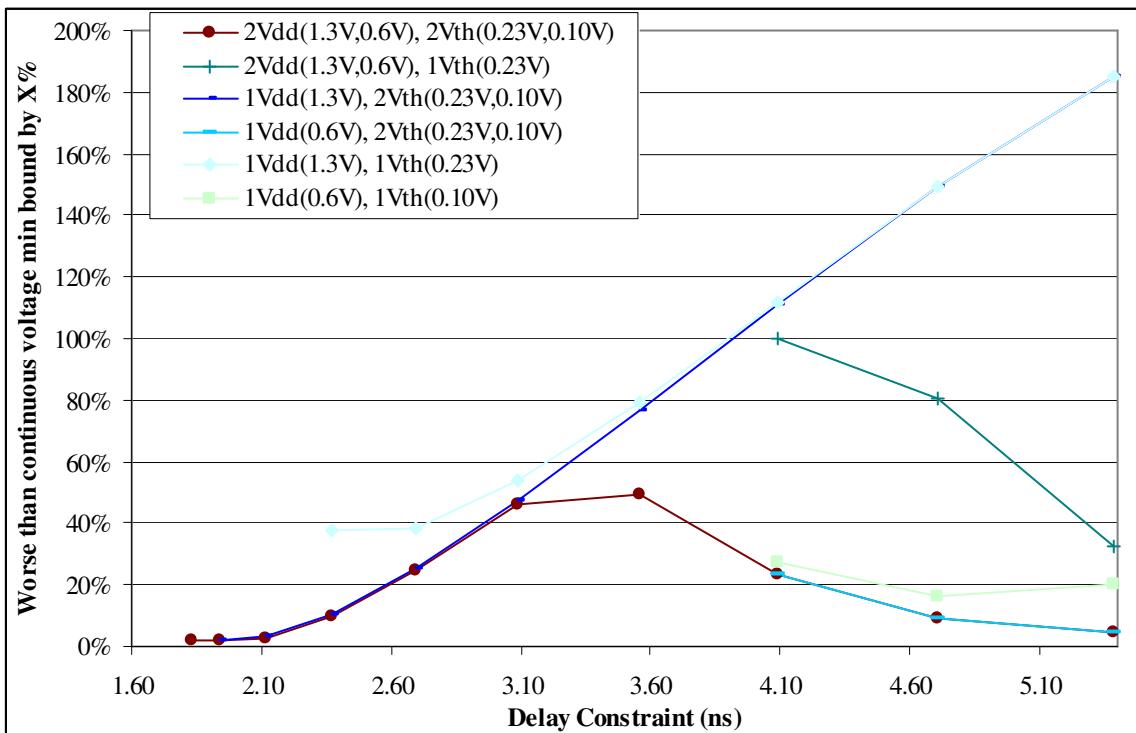


Figure 6.15 This graph shows how much worse the dual voltage and single voltage results with fixed global voltage values for c432nr are relative to the continuous lower bound.

6.5.2.3 Results for c17 and c432nr without wire loads

It is interesting to note the impact of wire loads on the gap between the single voltage and continuous voltage solutions. Not including wire loads in analysis is a common simplifying assumption used by researchers, though we will see that it gives quite inaccurate results.

For c17 without wire loads, the continuous voltage power is between 4.1% and 21.6% lower power than the single voltage solution, as shown in Figure 6.16. This is a much larger gap between the upper and lower bounds on the dual voltage solutions. Versus the single voltage solution, single Vdd/dual Vthn/dual Vthp provides up to 6.1% savings, dual Vdd/single Vthn/single Vthp provides up to 12.3% savings, and dual supply and threshold voltages provides up to 15.7% power savings. The power savings are up to about 10% more for the dual voltage solution, when wire loads are set to zero. This indicates the importance of including wire loads in circuit optimization problems involving gate sizing.

For c432nr without wire loads, the continuous voltage power is between 11.7% and 13.9% less than the single voltage power, as shown in Figure 6.17. The gap between the upper and lower bounds is larger without wire loads for c432nr, but not as much as in c17. We did not look at dual voltage results for c432nr without wire loads due to the longer run times.

Without wire loads, the minimum delay achievable is 12% lower for c17 and 16% lower for c432nr. At the minimum delay achievable for the result with wire loads, the power without wire loads is 85% lower for c17 and about 91% lower for c432nr. See Appendix D.6.4 for power versus delay curves with and without wire loads for c17 and c432nr. In comparison for the 0.13um channel length, rather than the 0.18um channel length used in this chapter, with the PowerArc Vdd=1.2V/Vth=0.23V library and the Design Compiler delay minimized ISCAS'85 benchmarks, we found that setting wire loads to zero reduced delay by 20% on average and reduced switching power by 37% when measured at the original clock frequency, but this was

without gate resizing. Ignoring wire loads when gates are sized introduces an even more substantial error in the power.

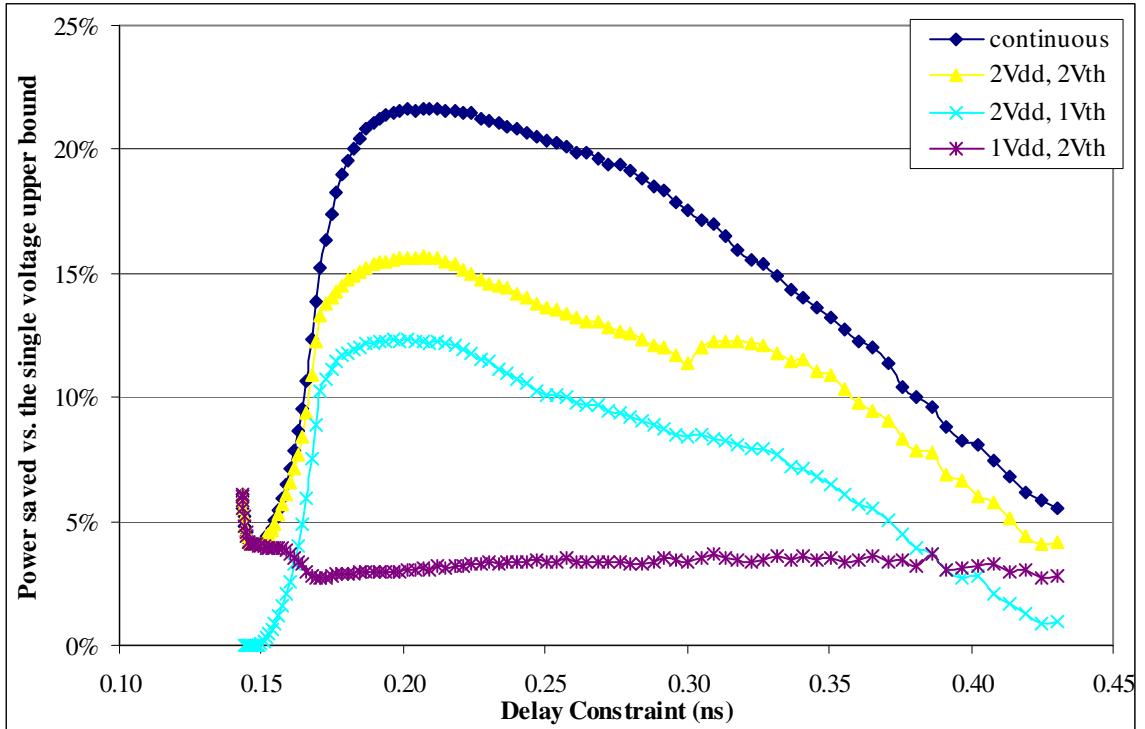


Figure 6.16 This graph shows how much power the dual voltage and continuous voltage solutions save versus the single voltage upper bound for c17 without wire loads.

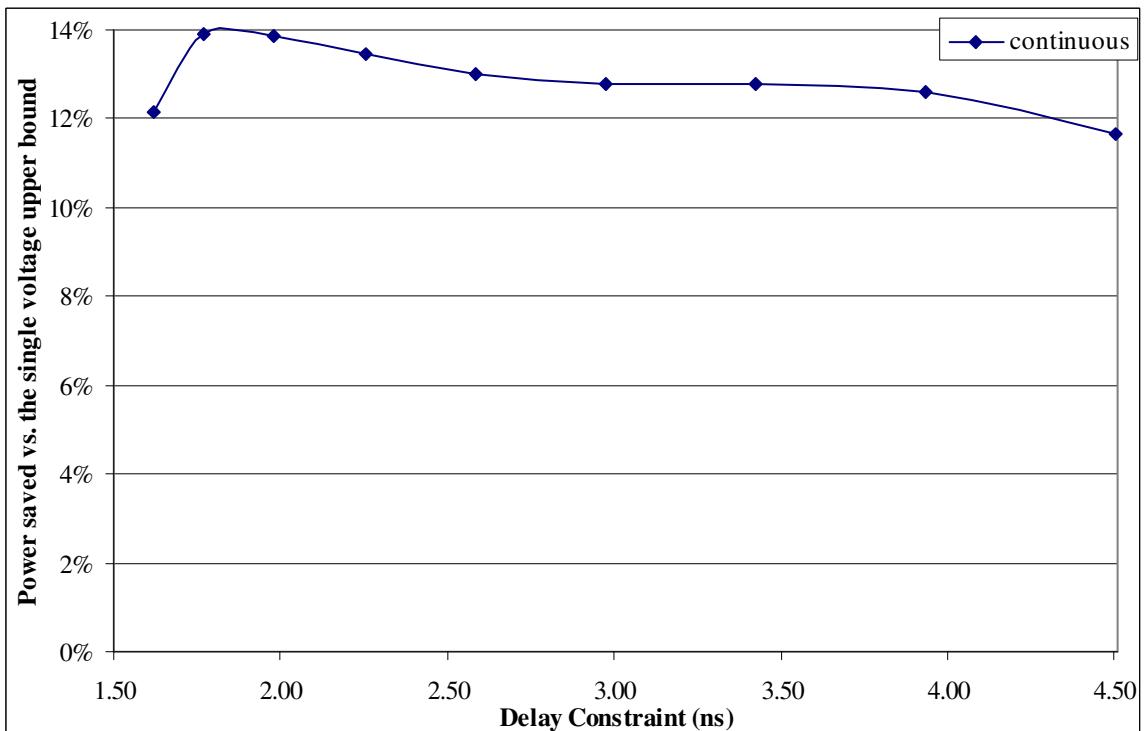


Figure 6.17 This graph shows how much power the dual voltage and continuous voltage solutions save versus the single voltage upper bound for c432nr without wire loads.

6.5.2.4 Results for c880

There was a noticeably larger gap between the single voltage upper bound and continuous voltage lower bound for c880. The continuous voltage power is up to 22.2% lower power than the single voltage solution, as shown in Figure 6.18. This leaves significant room for dual voltage solutions to improve upon the single voltage results. The gap between the single voltage and continuous voltage results decreases steadily as the delay constraint is relaxed.

There is a large gap between the single voltage and continuous voltage results because there is substantial timing slack on some paths in c880 that do not converge with timing critical paths, so input drivers and gates on those paths can be set to a low supply voltage. This enables the continuous voltage solution to substantially reduce the dynamic power and hence total power. Whereas to meet the delay constraint on the timing critical paths, the single voltage solution must use a high supply voltage¹.

With geometric programming solver runtimes of two to four hours for c880, we examined only a limited range of dual voltage solutions where the gap was largest between the single and continuous voltage solutions. As leakage was only 5% to 6% of the total power, using dual Vdd provided more power savings than using dual Vth. Dual Vdd/single Vth achieves up to 12.0% power savings versus the single voltage solution, whereas single Vdd/dual Vth provides only 8.6% power savings at this point. Dual Vdd/dual Vth provided 18.0% power savings versus the single voltage solution.

¹ For example at a delay constraint of 1.78ns, in the continuous voltage solution 12% of the supply voltages can be set to 0.600V and only 42% are at 1.300V to meet the delay constraint, whereas Vdd is 1.295V for the single voltage solution.

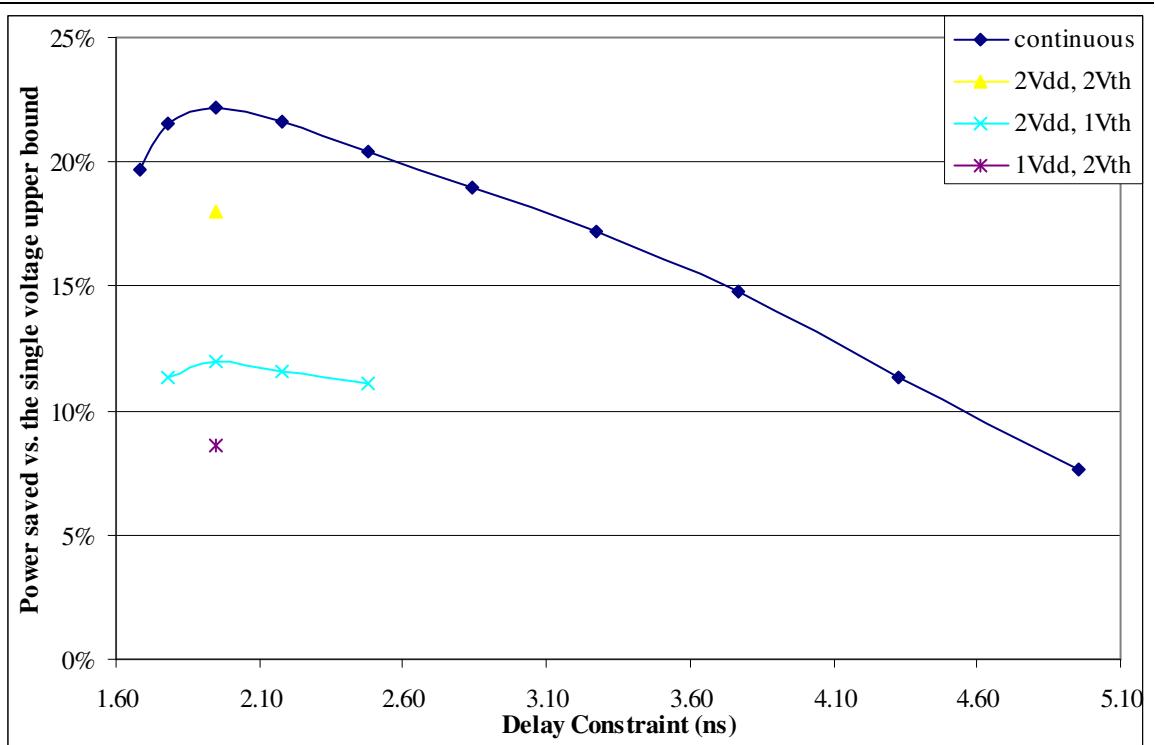


Figure 6.18 This graph shows how much power the dual voltage and continuous voltage solutions save versus the single voltage upper bound for c880.

Table 6.8 Geometric programming runtimes for delay minimization for the single voltage and continuous voltage runs. The number of (dependent and free) variables in the geometric program is determined by the number of inputs and gates in the benchmark. The number of constraints is also determined by the number of edges between vertices in the circuit graph representation. MOSEK performs elimination, reducing the number of variables and constraints, to reduce the runtime. The total runtime includes the time to set up the geometric program in MATLAB (linear in the circuit size), time for MOSEK to set up the problem and perform eliminations, and the time taken for MOSEK’s interior point solver to solve the geometric program.

Circuit	Run	# inputs	# gates	Original problem		After eliminations		Interior point solver runtime(s)	Total runtime(s)
		Variables	Constraints	Variables	Constraints	Variables	Constraints		
c17	single	5	6	108	378	95	351	0.7	1.3
	continuous			140	430	123	399	0.8	2.3
c432nr	single	36	222	3,194	12,212	2,697	11,488	1,072.6	1,208.9
	continuous			4,121	13,813	3,621	13,086	1,544.4	1,705.8
c880	single	60	555	7,904	29,182	6,381	26,784	11,883.1	12,716.0
	continuous			10,187	32,947	8,650	30,533	37,287.2	42,884.8
c499	single	41	580	8,216	31,386	7,087	29,487	15,567.0	16,505.7
	continuous			10,580	35,460	9,443	33,553	27,085.4	31,400.2

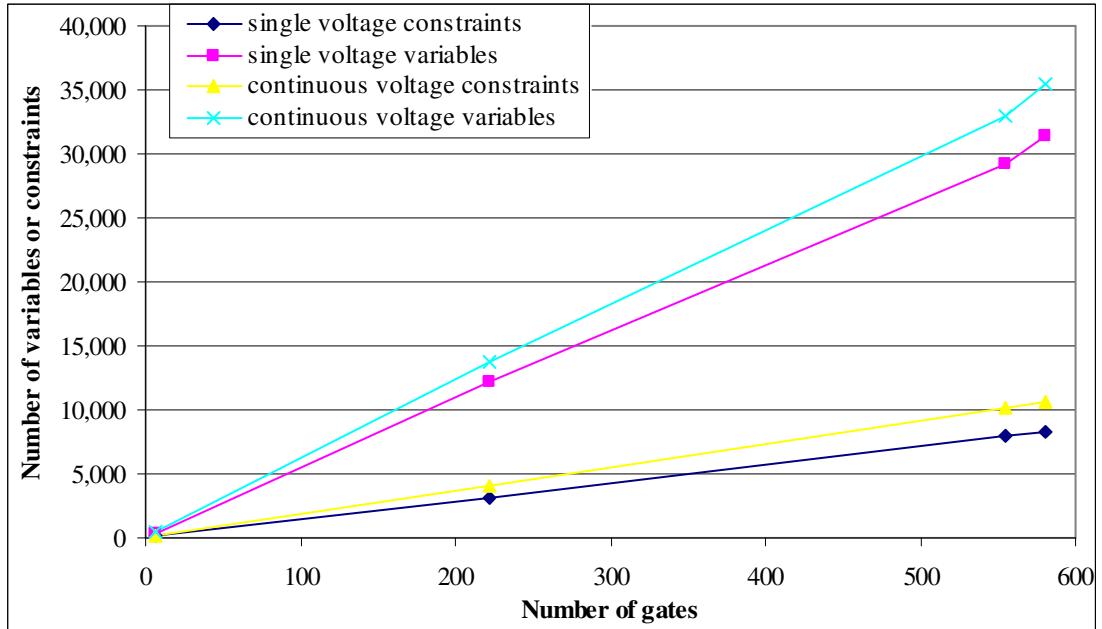


Figure 6.19 This graph shows that the number of variables and number of constraints both grow roughly linearly with circuit size.

6.6 Computational Runtime

The MOSEK software package [146] was used to the geometric optimization problems, which were posed via scripts in MATLAB. The time to pose the optimization problem and perform discrete assignments between iterations was negligible compared to the geometric program solver

runtime. The optimization runs were performed on a 2.4GHz Pentium 4 with 1GB of RAM and 512KB L2 cache.

The complexity of solving the geometric program is determined by the number of variables and number of constraints. The number of variables and constraints in the geometric program are shown in Table 6.8 for the continuous voltage and single voltage delay minimization problems for the benchmarks. MOSEK performs elimination to reduce the number of variables and constraints, in order to reduce the runtime. In the original continuous problem, there are three variables for each input: $V_{dd,gate}$, $C_{load,rise}$, and $C_{load,fall}$; and eighteen variables for each gate: $V_{dd,gate}$, $C_{load,rise}$, $C_{load,fall}$, d_{rise} , d_{fall} , $s_{out,rise}$, $s_{out,fall}$, $C_{in,rise}$, $C_{in,fall}$, $V_{dd,in}$, $t_{in,rise}$, $t_{in,fall}$, $s_{in,rise}$, $s_{in,fall}$, W_n , W_p , V_{thn} , and V_{thp} . Of these, the optimization variables are $V_{dd,gate}$, W_n , W_p , V_{thn} , and V_{thp} ; the rest are dependent variables determined by the constraints. In the single voltage optimization problem, the voltages ($V_{dd,gate} = V_{dd,in}$, V_{thn} and V_{thp}) must be the same for all inputs and gates. The number of constraints also depends on the number of graph edges. Both the number of constraints and the number of variables grow linearly with the circuit size, as shown in Figure 6.19. c499 and c880 with about 10,000 variables and 30,000 constraints are near the upper limit of geometric program sizes that are computationally feasible today.

The majority of the runtime is for MOSEK's interior point solver to solve the geometric program, as shown in Figure 6.20. There is a significant increase in setup runtime above 30,000 variables, possibly because of running out of memory and using virtual memory swapping with the hard disk – c499 and c880 take 1.6GB and 1.5GB of memory respectively when MOSEK is running, but the computer used for runtime benchmarking had only 1.0GB of memory.

The average runtimes for the interior point solver for power minimization were similar to delay minimization runtimes. Depending on the delay constraint and voltage assignments, the power

minimization runtimes can vary from $0.1\times$ to $4\times$ the runtime for delay minimization with continuous voltages.

To determine how the runtime grows with circuit size it is more useful to plot the total runtime with axes in logarithmic scale, as shown in Figure 6.21. Also graphed are equations of the form $y = mx^3$, which are linear on a log-log scale: $\log y = \log m + 3 \log x$. Ignoring the small runtimes for c17 where setup time is more significant, the total runtime grows as $O(k^3)$ where k is the number of variables after elimination. As the reduced number of variables is approximately linear with the number of vertices $|V|$, that is the number of gates and inputs, the runtime grows as $O(|V|^3)$.

The number of discretization iterations required to discretize N values is $O(\log_2 N)$ as described in Section 6.4.1. However, many of the voltages in the continuous voltage solutions are already assigned to the minimum voltage or maximum voltage, which can substantially reduce the number of values that need to be discretized. As was detailed in Appendix D.5, 33 optimization runs were required at a delay constraint of 2.12ns to discretize supply and threshold voltage values for c432nr – in comparison, if all the voltage values needed to be discretized, with three runs on each iteration (continuous, upper bisection, lower bisection), about 72 runs would be required. This discretization run for c432nr is quite typical. However, the average geometric program solver run time for power minimization for c432nr was about 14 minutes, thus about 7 hours are required to find the dual voltage solution. For c880 there were 73 optimization runs at a delay constraint of 1.95ns to discretize supply and threshold voltages, and the total runtime was 10.7 days on a 3.06GHz Xeon processor with 2GB of RAM and 512KB L2 cache. The runtime becomes prohibitive due to the $O(|V|^3)$ growth.

These large runtimes render the geometric programming approach infeasible for the general problem of optimizing circuits of an interesting size to designers today – at least tens of thousands

of gates, but often much larger. In Chapter 4 and Chapter 7, we examine formulating the problem as a linear programming problem that is much faster to solve, with run time complexity in the range from $O(|V|)$ to $O(|V|^2)$. The linear programming formulation is heuristic and can get stuck in local minima, but being able to scale to much larger circuit sizes is more important provided we show that good results can be achieved.

Another disadvantage of the geometric programming formulation is that interior point methods must be used to solve it, but it is not possible to “warm start” from a previous solution¹. For example, if we change the delay constraint or number of gates assigned to a particular voltage, knowing the solution to the original problem does not help solving the perturbed problem. In contrast, there are warm start methods for some linear programming optimization methods [146].

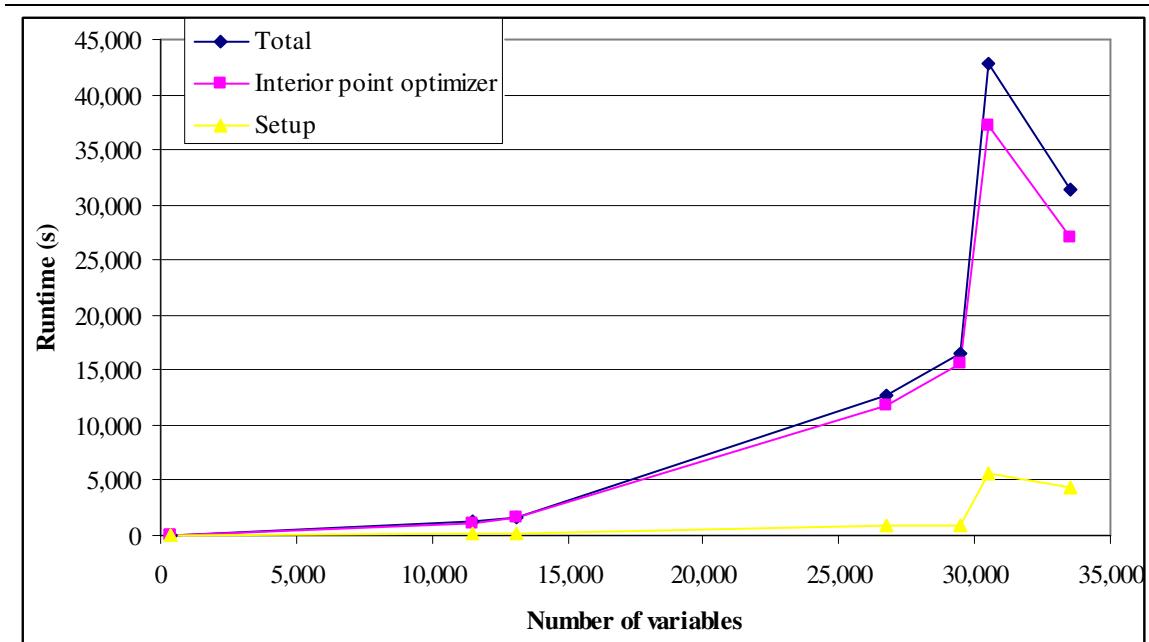


Figure 6.20 This graph shows the total runtime, runtime for the interior point optimizer, and the setup runtime (i.e. the setup time to specify the geometric program in MATLAB, the setup time for MOSEK and for MOSEK to perform variable elimination) for the single and continuous voltage optimization problems versus the reduced number of variables after MOSEK has performed variable elimination.

¹ As Erling Anderson noted over email.

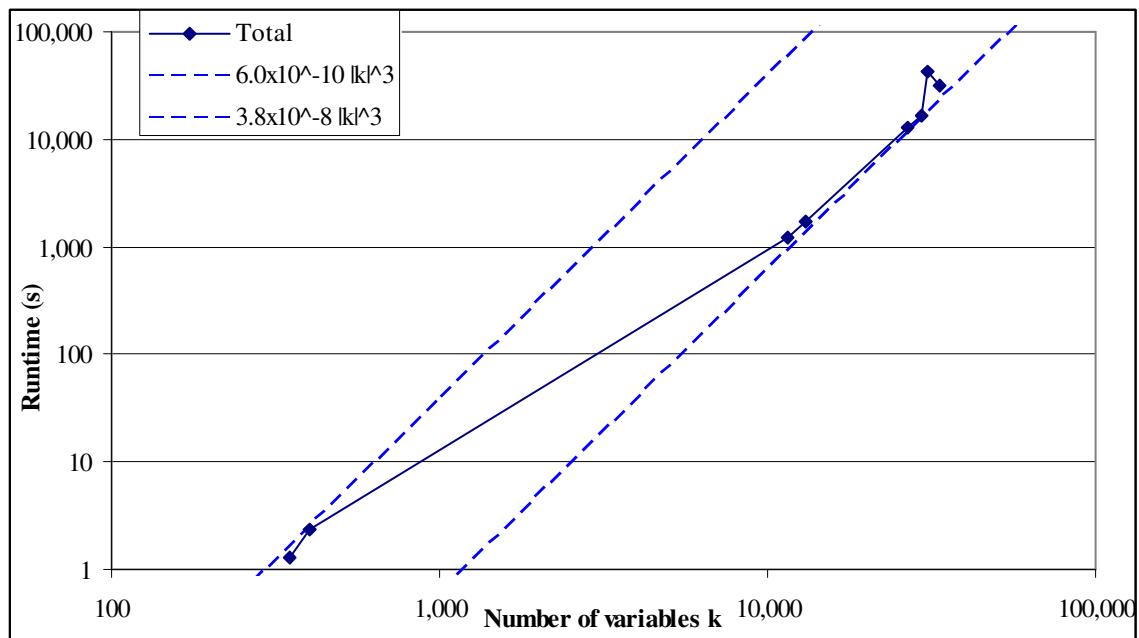


Figure 6.21 This graph shows the total runtime from Figure 6.20 on a log-log scale. Also shown are two lines representing cubic growth of runtime with the number of variables k , to illustrate that the runtime grows approximately as $O(k^3)$.

6.7 Conclusions

We have shown that power minimization of a combinational circuit with gate sizing, and supply voltage and threshold voltage assignment is an NP-complete optimization problem. Other researchers have adopted heuristic approaches to make the problem computationally tractable, but heuristic solutions can be suboptimal and there is no knowledge of how far they are from the global minimum.

If we *approximate* the delay and power data with posynomial models, we can pose the problem as a convex optimization problem in terms of continuous variables, for which the global minimum is polynomial time solvable. The posynomial fits had relative root mean square accuracy of 10% or better. This was over 60,000 to 160,000 sample points from SPICE characterization of NAND2, NAND3, NAND4, NOR2, NOR3, NOR4 and inverter gates using STMicroelectronics' 0.13um technology. The large number of sample points comes from eight characterization variables: input slew, load capacitance, input voltage, NMOS width, PMOS width, supply voltage, NMOS threshold voltage, and PMOS threshold voltage. Given the large number of sample points and

small number of fitting coefficients (27 or less), the accuracy is quite good, and comparable to what has been achieved by other authors for transistor sizing alone [117][207]. We have shown that supply and threshold voltages can be included in posynomial models with reasonable accuracy.

Unfortunately, the underlying data is non-convex when transistor folding and other issues are accounted for. Thus the accuracy of convex fits is fundamentally limited by the real data. 10% static timing and power analysis inaccuracy is not acceptable for a circuit designer, but it is sufficient to explore the optimization space. It is not possible to ensure that the posynomial fits are conservative, that is never underestimating delay or power, because the accuracy of conservative fits is substantially worse – for example, 23% relative root mean square error for inverter fall delay. The linear programming formulation in Chapter 4 and Chapter 7 gives the same static timing and power analysis results as the commercial tool Design Compiler [201] and thus does not have these accuracy issues.

We performed delay minimization, and power minimization subject to a delay constraint. Other objectives such as energy-delay product (i.e. PT^2 , where P is power and T is the critical path delay) are also possible. The solution to the geometric program gives a range of values for supply voltage, threshold voltages, and gate sizes. We can avoid picking discrete gate sizes by assuming a “liquid cell” sizing methodology or a library with a very fine granularity of gate sizes. However, supply and threshold voltages must be discretized, as only one or two supply or threshold voltages are acceptable because of the associated design and processing costs. The initial solution with values from a continuous range, the continuous voltage solution, provides a lower bound on what may be achieved with dual supply voltages and dual threshold voltages.

We proposed a discretization heuristic that allows us to find a dual voltage solution near the continuous voltage lower bound with $O(\log_2 N)$ iterations of optimization, where N is the number

of values to be discretized. This binary search method is very similar to the classic Fibonacci search method. The discretized dual voltage results were at most 5.4% higher power than the continuous voltage lower bound and on average were 2% worse, ignoring results without wire loads. Thus there is only a small penalty for discretization to dual supply and dual threshold voltages, with gate sizing.

We can also pose the geometric programming problem where all gates have the same supply voltage, same NMOS threshold voltage, and same PMOS threshold voltage. The single voltage solution provides an upper bound on the dual voltage results. In particular, the gap between the single voltage upper bound and continuous voltage lower bound is the maximum savings that can be achieved with multiple supply and threshold voltages. For c17, c432nr and c499, the continuous voltage solution gave less than 9% power savings versus the single voltage results. Thus there is little room for improvement with dual voltages, and small power savings do not justify the additional process costs. However for c880, the continuous voltage power was up to 22.2% less than the single voltage result. Thus dual voltages might actually be worthwhile for c880.

For c17, we saw up to 3.5% power savings versus single voltage results with single Vdd/dual Vthn/dual Vthp or dual Vdd/single Vthn/single Vthp, and up to 5.5% savings with dual supply and dual threshold voltages. Almost all the power savings for c432nr were with dual threshold voltages, the single Vdd/dual Vthn/dual Vthp results being up to 7.2% lower power than the single voltage solution; dual supply voltages provided 0.5% or less power savings for c432nr. We did not perform discretization to dual voltages for c499 due to the long runtimes and small gap between the continuous voltage lower bound and single voltage upper bound. For c880, we have seen up to 12.0% power savings with dual Vdd/single Vthn/single Vthp, 8.6% power saving with single Vdd/dual Vthn/dual Vthp, and 18.0% power saving with dual supply and dual threshold voltages.

Both the single voltage and continuous voltage solutions are globally optimal, whereas the discretized dual voltage solution is heuristic. For c17, we have seen cases where the dual voltage solution found is up to 4% worse than the minimum dual voltage solution. The single voltage and dual voltage solutions are technologically feasible, but the continuous voltage solution is not.

If the supply and threshold voltage values are limited to a couple of values each as provided in standard cell libraries from a foundry, then significant power savings may be missed. Even with dual Vdd and dual Vth, the power was up to about 50% worse than the continuous lower bound for c17 and c432nr with fixed values of 1.3V and 0.6V for Vdd, and nominal and nominal – 0.13V for Vth. This shows how important it is to scale voltages optimally for a design and a given performance constraint. Substantial power savings may be achieved if foundries provide a larger number of supply and threshold voltages to choose from to find something near the optimal.

The geometric programming formulation cannot handle power and delay overheads for insertion of voltage level converters, because of the stepwise discontinuity introduced. Thus this approach was limited to a clustered voltage scaling (CVS) multi-Vdd methodology where level restoration occurs only at the combinational outputs with level converter flip-flops, for which zero delay and zero power overhead is assumed. Our linear programming approach can handle level converter overheads as discussed in Chapter 7.

The geometric programming computational runtime with MOSEK grows rapidly as $O(|V|^3)$, where $|V|$ is the number of inputs and gates in a circuit. For example, the delay minimization runtime for c880 with 60 inputs and 555 gates was nearly 12 hours. This is too slow. The geometric programming formulation will be computationally infeasible for circuits of an interesting size to circuit designers, tens and hundreds of thousands of gates typically. While the discretization heuristic takes only $O(\log_2 N)$ steps to discretize N values, the large runtimes for a single optimization run are prohibitive to performing multiple optimization runs for discretization.

Lagrangian relaxation has been used recently to solve geometric programs for gate sizing and threshold voltage optimization with quadratic runtime growth on benchmarks of up to 3,512 gates [42], see Appendix H.1.9 for details, but it remains to be seen how well this approach scales to larger circuit sizes.

The maximum feasible circuit size could be increased slightly by reducing the number of variables. Reducing the number of dependent variables modeling the delay and power would increase the relative RMS error beyond 10%. Reducing the number of optimization variables for a gate, for example forcing NMOS and PMOS threshold voltage to be the same, would make the solution suboptimal. Partitioning the circuit into groups of at most N gates would give runtime complexity of $O(N^3|V|)$, that is linear growth with circuit size, but the solutions would be suboptimal.

Recent work by Zlatanovici and Nikolić has shown that in custom carry-look-ahead Kogge-Stone 64-bit adders, where the layout and gate sizes can be carefully managed, the repetitive structure allows groups of gates to be sized in an identical manner substantially reducing the number of variables and constraints for a posynomial formulation [247]. This made the geometric programming formulation computationally feasible for a 64-bit domino carry-look-ahead adder comprising 1,344 gates. With the geometric program solution, they used heuristic optimization with more accurate models to produce a very high speed 4GHz adder in 90nm technology at 1V supply voltage [114]. However, tight control of wire length is not possible with automated place and route, so identical conditions for individual gates cannot be assured in ASICs. Consequently, convex optimization formulations will not be practical for use on ASIC digital circuits of significant size (tens of thousands of gates or more). For optimization of general circuits, we must turn to faster heuristic methods, such as the linear programming approach in Chapter 4 and Chapter 7 which has run time complexity in the range of $O(|V|)$ to $O(|V|^2)$.

The results in this chapter suggest that use of multiple supply and multiple threshold voltages seldom provide significant savings versus using a single supply and single threshold voltage values that have been optimized for the design. This is borne out by many of the results in Chapter 7, where power savings of 20% to 30% were only seen with a single supply voltage scaled down from 1.2V to 0.8V and dual threshold voltages of 0.08V and 0.14V. The results are not directly comparable as SPICE characterizations for this chapter had 0.18um channel length, whereas the 0.13um PowerArc characterized libraries used in Chapter 7 had 0.13um channel length, transistor folding and other differences.

Chapter 7. Linear Programming for Threshold Voltage and Supply Voltage Assignment with Gate Sizing

Having provided a strong gate sizing benchmark using only a single transistor threshold voltage (V_{th}) and single supply voltage (V_{dd}) in Chapter 4, we now examine the impact of additionally using multiple- V_{th} and dual V_{dd} to minimize power. Comparing cells with different V_{th} values is no different to comparing cells with different sizes, providing that the leakage is included in the total circuit power. Multiple supply voltages can also be handled similarly, with level converter overheads for restoring to high V_{dd} .

Our dual- V_{dd} /dual- V_{th} /sizing results achieve on average 5% to 13% power savings versus the two alternate dual- V_{dd} /dual- V_{th} /sizing optimization approaches suggested in [124] and [188]. Importantly, the linear programming approach has runtimes that scale between linearly and quadratically with circuit size, whereas other algorithms that have been proposed for multi- V_{dd} , multi- V_{th} and gate size assignment have cubic runtime growth. This chapter examines in detail optimization with multiple supply voltages and multiple threshold voltages.

7.1 Introduction

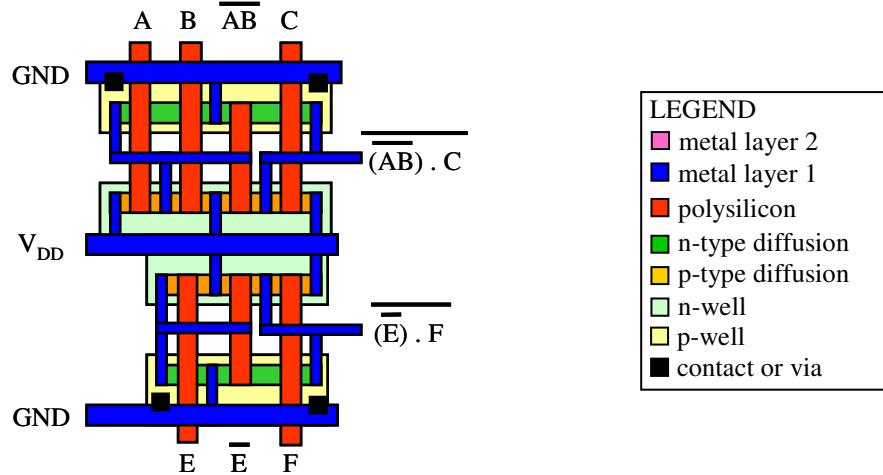
A high supply voltage and a low threshold voltage may be necessary to meet circuit delay constraints. However, using a lower supply voltage can quadratically reduce the dynamic power, and using a higher threshold voltage can exponentially reduce the leakage power. Thus it is possible to substantially reduce power while meeting delay constraints by using high V_{dd} with low V_{th} on delay critical paths, and low V_{dd} with high V_{th} where there is sufficient timing slack. There are significant design costs for using multiple supply voltages and multiple threshold voltages, so circuit designers are concerned about how much power saving multi- V_{dd} and multi- V_{th} can truly provide.

Each additional PMOS and NMOS threshold voltage requires another mask to implant a different density of dopants, which substantially increases processing costs. A set of masks costs on the order of a million dollars today and an additional V_{th} level increases the fabrication cost by 3% [165]. Each additional mask also increases the difficulty of tightly controlling process yield, which strongly motivates manufacturers to limit designs to a single NMOS and single PMOS threshold voltage. From a design standpoint, an advantage of multiple threshold voltages is that changing the threshold voltage allows the delay and power of a logic gate to be changed without changing the cell footprint, and thus not perturbing the layout.

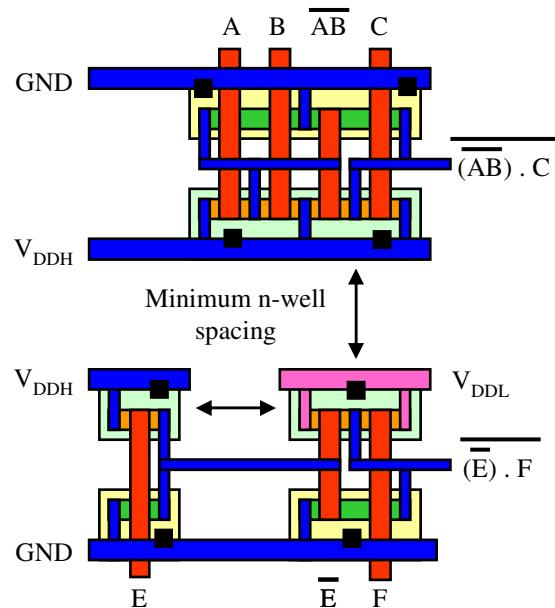
Each additional supply voltage requires an additional voltage regulator and power supply rails for that voltage. The logic needs to be partitioned in some manner into voltage regions where a single supply is used. The regions of each supply voltage are not usually fully utilized and some spacing is required between them, increasing chip area. Wire lengths also increase between cells in different V_{dd} regions. The area overhead for gate-level dual V_{dd} assignment in modules of a media processor was 15% [224].

An alternative is to route the two supply rails along every standard cell row, which increases the cell height and has a similar area overhead. In bulk CMOS there are also minimum spacing issues between the PMOS n-wells at different biases to prevent latchup¹, as shown in Figure 7.1(b). The PMOS n-wells in high V_{dd} (V_{DDH}) gates cannot be connected to low V_{dd} (V_{DDL}) as this forward biases the transistors, increasing leakage substantially, and can cause other problems. V_{DDL} gates can have the PMOS n-well connected to V_{DDH} as shown in Figure 7.1(c), but this reverse biases the transistors, making the V_{DDL} gate even slower – though this can be compensated for by using a lower PMOS V_{th} for V_{DDL} gates.

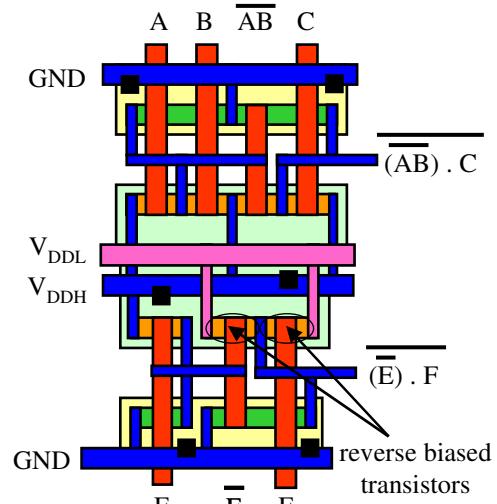
¹ Private communication with Bhushan Gupta and Farhana Sheikh.



(a) Single V_{DD} requires less area than dual V_{DD}



(b) Horizontal and vertical well isolation issues when PMOS n-wells connect to different V_{dd}



(c) Connecting the n-well of the PMOS transistors to V_{DDH} reverse biases the substrate of the V_{DDL} NAND gate (by $V_{DDH} - V_{DDL}$), increasing the PMOS V_{th}

Figure 7.1 This diagram illustrates some of the differences between single V_{dd} and dual V_{dd} layout. Single V_{dd} layout is more compact as shown in (a). If the PMOS n-wells are connected to different supply voltages, then there are minimum spacing requirements as shown in (b). An alternative is to connect the n-wells of PMOS transistors in both V_{DDH} and V_{DDL} gates to V_{DDH} , but this reverse biases the PMOS transistors in the V_{DDL} gate as shown in (c). Note that the PMOS n-wells in (c) are all connected to V_{DDH} .

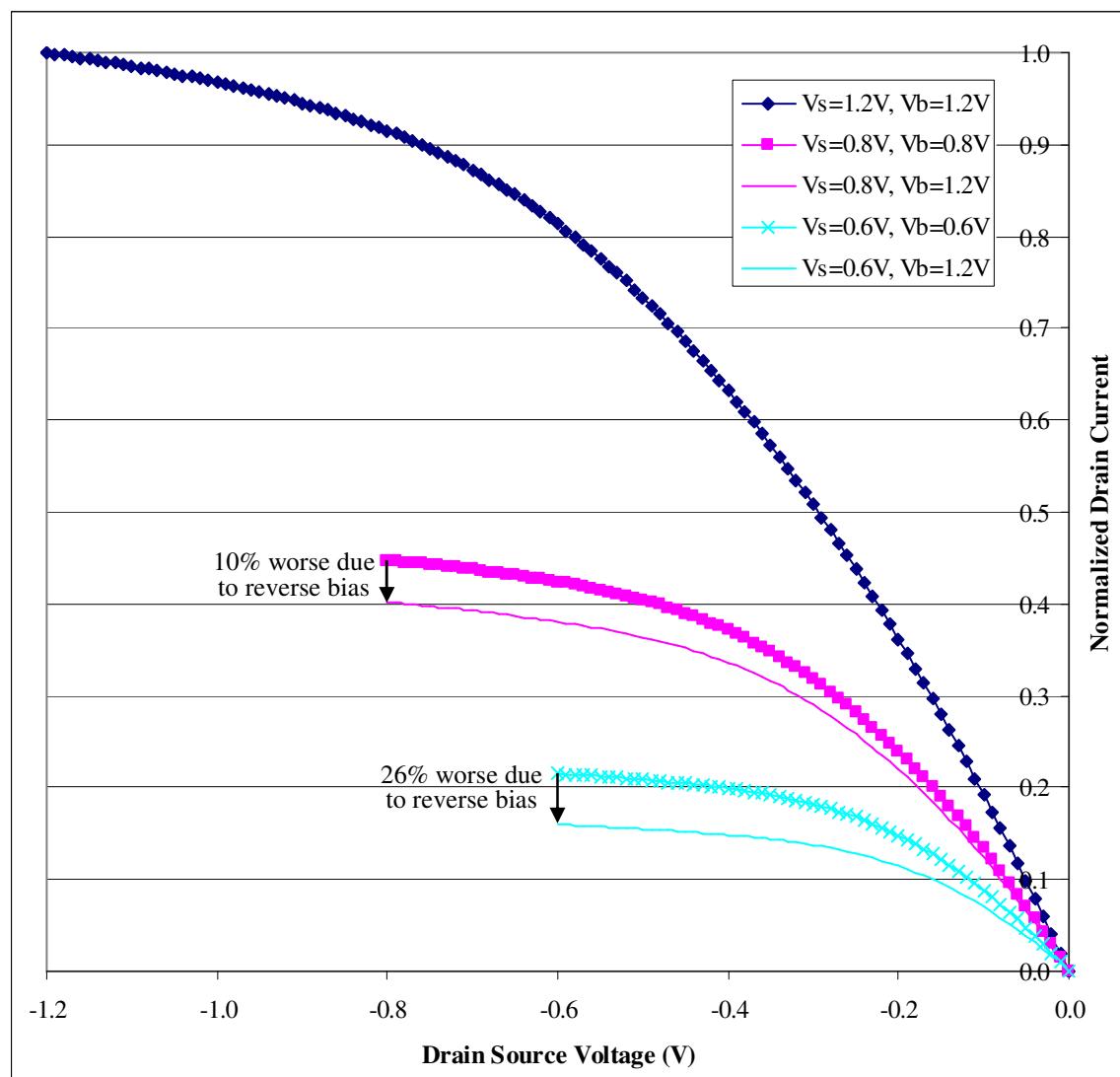


Figure 7.2 This graph shows the impact of reverse biasing the PMOS substrate. V_s is the source voltage (the supply voltage for an inverter), V_b is the body bias, and the input voltage is fixed at 0V. The body is reverse biased when $V_b > V_s$, which increases the threshold voltage and reduces the current. The thicker lines show drain current without reverse biased PMOS substrate, and the thin lines show the drain current with substrate reverse biased at 1.2V.

If the substrate is reverse biased at 1.2V for PMOS transistors with 0.8V and 0.6V drain-source voltage, the drain current is 10% lower at 0.8V and 26% lower at 0.6V, as shown in Figure 7.2. Even without being reverse biased, the PMOS transistor drain current is 55% and 78% lower respectively than a PMOS transistor connected to a high supply voltage of 1.2V. In silicon-on-insulator (SOI) technology, the transistors are isolated and spacing between wells at different biases is not an issue.

If the alternate supply voltage rails are routed on each standard cell row, it is much easier to change the supply voltage of cells without substantially perturbing the layout, simplifying post-layout optimization. It also makes it easy to connect to both power supply rails, which is needed for most voltage level restoration “level converters” [165]. When different supply voltages are routed next to each other, the second metal layer may be needed for internal wiring in logic cells, which creates blockages on metal layer two, reducing routing resources for wiring between cells.

Whether using separate voltage regions or routing the supply voltages next to each other, using multi-Vdd increases the chip area, which lowers the yield per wafer and increases fabrication costs.

Our optimization approach does not consider the increased area for multi-Vdd and the impact on yield of both multi-Vdd and multi-Vth approaches, but clearly there must be significant power savings to justify the cost. Large power savings have been suggested by a number of researchers, but in some cases the delay and the power models were inaccurate, or the initial comparison point was poor, causing power savings to be overstated. To justify the use of multi-Vdd and multi-Vth, we must show substantial power savings versus a good choice of single supply voltage and single threshold voltage with power minimization by gate sizing.

It is essential that gate sizing is considered with multi-Vdd and multi-Vth, as gate sizing can achieve greater power savings at a tight delay constraint. We found that the power savings achieved with multi-Vdd and multi-Vth are in most cases less than the power savings achieved by gate sizing with the linear programming approach versus the TILOS-like optimizers.

While multiple threshold voltages do not complicate the optimization problem, using multiple supply voltages requires insertion of voltage level converters to restore the voltage swing to higher Vdd gates as described in Section 7.2. As there is yet no standard approach for multiple supply voltage and multiple threshold voltage optimization, Section 7.3 summarizes previous

research in the area, discussing the limitations and advantages of the various optimization approaches. Few papers have considered trying to perform simultaneous optimization with assignment of multiple supply voltages, multiple threshold voltages, and gate sizes. How the voltage level converter power and delay overheads are handled with the linear programming approach is detailed in Section 7.4.

Perhaps the best of the multi-Vdd/multi-Vth/sizing optimizations thus far are two approaches proposed by Sarvesh Kulkarni, Ashish Srivastava, Dennis Sylvester, and David Blaauw [124][125][188]. Under the same conditions, the linear programming approach for multi-Vdd/multi-Vth/sizing is compared versus their approaches in Section 7.5. On average, the linear programming approach reduces power 5% to 13% versus their results across a range of delay constraints.

Having established that the linear programming approach performs well for supply voltage assignment and threshold voltage assignment as well as gate sizing, Section 7.6 examines how much power can be saved with multi-Vth and multi-Vdd versus using a single Vdd and single Vth. Section 7.7 briefly discusses the impact of multi-Vdd and multi-Vth assignment in addition to sizing on the runtimes. A summary of our results are presented in Section 7.8.

7.2 Voltage level restoration for multi-Vdd

If a low Vdd (VDDL) input drives a high Vdd (VDDH) gate, the PMOS transistors are forward biased by $V_{DDL} - V_{DDH}$ which results in static current. This is illustrated with two inverters in Figure 7.3. To avoid this, a voltage level converter is needed to restore the signal to full voltage swing, restoring the signal from $0V \leftrightarrow VDDL$ to $0V \leftrightarrow VDDH$. A VDDH gate may drive a VDDL gate.

Algorithms for gate level supply voltage assignment can be broadly separated into two methodologies depending on where voltage level restoration may occur, as illustrated in Figure

7.4. Clustered voltage scaling (CVS) [222] refers to when voltage level restoration only occurs at the registers, to reduce the level converter power and delay overhead. All VDDL gates must either drive VDDL gates or drive a level converter latch. Hence there are distinct clusters of VDDL combinational gates that have only VDDL combinational gates in their transitive fanout. A gate may only be changed from VDDH to VDDL if the fanouts are all VDDL, or changed from VDDL to VDDH if the fanins are all VDDH. Extended clustered voltage scaling (ECVS) [225] additionally allows “asynchronous” level converters to be placed between combinational logic gates, removing the restrictions on when a gate’s Vdd may be changed. In ECVS, VDDL gates are still clustered in order to amortize the power and delay overheads for the level converters.

Combining a level converter with a flip-flop minimizes the power overhead for voltage level restoration. As typical level converter and flip-flop designs essentially include a couple of inverters acting as a buffer, the level converter can replace these in the flip-flop. The power consumed by an LCFF can be less than that of a VDDH flip-flop, particularly if a low-voltage clock signal is used [15]. An LCFF with high-speed pulsed flip-flop design is comparable in delay to a regular D-type flip-flop [110], thus avoiding the level converter delay overhead. The delay and power overheads for voltage level restoration are minimal in CVS.

Due to the additional power and delay overheads for asynchronous level converters, there have been reservations about whether ECVS provides any practical benefits over CVS [222]. There has also been concern about the noise immunity of asynchronous level converters [110]. The asynchronous level converters that we use for ECVS were shown to be robust and have good noise immunity [123]. In Section 7.5, we see up to 12.6% power reduction for ECVS versus CVS dual Vdd, and average power savings of 5.4% or less.

We now look at the algorithms that have been proposed previously for multi-Vth and multi-Vdd optimization.

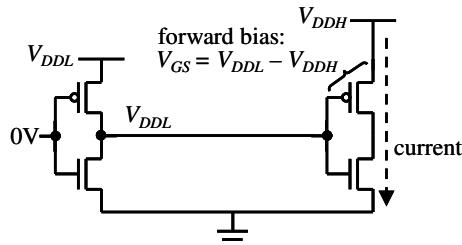
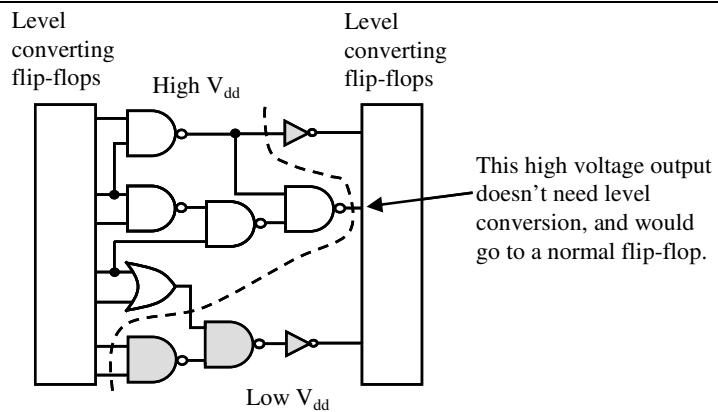
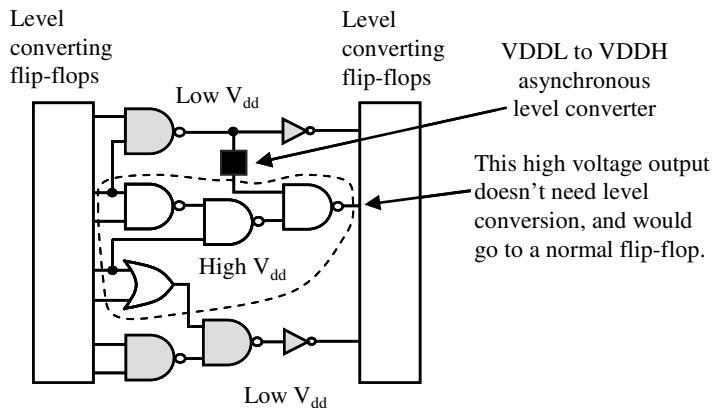


Figure 7.3 This diagram illustrates the need for voltage level restoration. The schematic shows an inverter with supply voltage of V_{DDH} , being driven by an inverter of supply voltage V_{DDL} . When the V_{DDL} inverter output is high, the driving voltage is only V_{DDL} , which results in a forward bias of $V_{DDL} - V_{DDH}$ across the PMOS transistor of the V_{DDH} inverter. The forward-biased PMOS transistor is not completely off resulting in large static currents.



(a) Clustered Voltage Scaling (CVS) – combinational logic can be partitioned into separate VDDH and VDDL portions in topological order.



(b) Extended Clustered voltage scaling (ECVS) – a VDDL signal may go to a VDDH gate if an asynchronous level converter is inserted. Here the logic cannot be partitioned into separate VDDH and VDDL portions that are topologically ordered.

Figure 7.4 This illustrates the difference between the circuit structures after the application of extended clustered voltage scaling (ECVS) and clustered voltage scaling (CVS).

7.3 Summary of previous research on optimization with multi-Vdd and multi-Vth

A number of researchers have explored use of multiple threshold voltages, and/or multiple supply voltages. Some papers report large power savings versus an initial configuration that is substantially sub-optimal. For example when starting with a circuit with high leakage power with all transistors at low threshold voltage, introducing a second higher threshold voltage will provide significant power savings, but the real question is how much power would be saved with dual Vth versus choosing a more optimal single threshold voltage and sizing gates optimally. This was one of the concerns of our industrial liaisons from Intel. To address this, multi-Vth and multi-Vdd results in Section 7.6 are compared versus the optimal sizing results with the best threshold voltage available from a choice of three Vth values – though if a finer granularity of Vth were available, that would no doubt provide some additional power savings. In particular, we generally look at dual Vth/sizing power savings versus sizing with a higher Vth library, as it is much more difficult to reduce power versus an initial configuration that is already low on leakage power that has gate sizes optimized to reduce dynamic power.

Another major shortcoming of many academic papers is using simplified delay and power models that are inaccurate, such as ignoring slew and failing to use separate rise and fall timing arcs. Few algorithmic papers state the accuracy of their models. In some cases, it is not clear how optimization approaches with such models can be extended to real circuits using full static timing analysis with standard cell libraries, as their algorithmic approach or computation speed depend on the underlying simplified models. Merging rise and fall delays halves the number of delay constraints in optimization. Simplified power and delay analysis that ignores slew and doesn't use lookup tables for analysis, for example using linear interpolation versus load capacitance, can speed up analysis runtimes by an order of magnitude. This has a major impact on the total computational runtime as analyzing trade-offs is an essential portion of the inner loop of any optimizer. For example, runtimes reported for our initial linear programming approach [154] are

about 10× faster with 0.18um simplified logical effort delay models and no internal power analysis versus interpolating 0.13um library data. The 0.13um interpolation for each gate size versus load capacitance did not include slew analysis, which would further increase computation time.

Given the range of computers used to run benchmarks in different papers, it is difficult to directly compare runtimes across tools. For the purposes of sizing large circuits in industry, it is more interesting to compare the runtime complexity. It is essential that runtime complexity be less than $O(|V|^2)$ to run on circuits of any appreciable size [195].

We summarize some of the better multi-Vth and multi-Vdd optimization approaches below. Detailed summaries for these and other papers are in Appendix H.

7.3.1 Summary of papers on optimization with multi-Vth

TILOS-like optimizers have been used for multi-Vth assignment by a number of researchers, including Wei et al. [232]; Sirichotiyakul et al. [183]; and Wei, Roy, and Koh [233]. These optimizers proceed in a greedy manner, picking the gate with the best power or area versus delay tradeoff to change, and iterating. A number of other optimization heuristics have also been tried.

Most of the threshold voltage assignment approaches detailed in Appendix H.1 concentrate on reducing leakage power. Wei, Roy, and Koh minimized total power [233]. Power dissipation is a major constraint in today's technologies, so minimizing total power is the more appropriate optimization objective. If so desired, total power could be minimized until a given constraint is reached, and then the objective could be set to reducing leakage power with constraints on both total power and delay. The only way to encode multiple objectives in an optimization is to weight the objectives according to their priority and include appropriate constraints, but it is generally best to find feasible solutions, for example satisfying the delay constraint, before focusing on a secondary objective.

Sirichotiyakul et al. began with a high V_{th} circuit and iteratively assigned transistors to low V_{th}, with transistors prioritized in a TILOS-like manner for the delay reduction versus the increase in leakage power. After a transistor was assigned to low V_{th}, transistors in neighboring gates over three levels of logic were resized. In the 0.25um process with supply voltage of 0.9V, their dual V_{th} approach reduced leakage by 3.1× to 6.2× with at most 1.3% delay penalty [183] compared to gate sizing with all gates at low V_{th}. The circuit sizer was a TILOS-like sizer that provides a good baseline, but, as noted earlier, large savings can be achieved versus an all low V_{th} configuration. They did not consider the impact on dynamic power and the total circuit power consumption. Analysis of their algorithm indicates a theoretical complexity of $O(|V|^2)$.

Wei, Roy and Koh began with all gates at minimum size and high V_{th}. The sensitivity metric for gate upsizing or reducing a gate's threshold voltage was $-\Delta d/\Delta P$. In a 0.25um process with V_{dd} of 1V and threshold voltages of 0.2V and 0.3V with 0.1 switching activity, total power was reduced by 14% using dual V_{th} and gate sizing versus gate sizing with low V_{th} [233]. The theoretical worst case runtime complexity of this approach is the same as TILOS, $O(|V|^2)$.

Wei et al. compared gate-level assignment versus stack-level, and versus assignment at the level of series connected transistors, and found that series-level assignment provided 25% better power reduction than gate-level assignment [232]. Our work does not directly consider transistor-level V_{th} assignment, but if standard cell libraries are available with characterized cells of mixed-V_{th}, it is straightforward to use them in the optimization.

These TILOS-like algorithms appear to be the best of the V_{th} assignment algorithms, as other multi-V_{th} research has not shown better results than TILOS. We do not compare the LP approach versus TILOS for gate-level threshold voltage assignment, as Chapter 4 has already shown that our linear programming approach produces better sizing results than TILOS.

7.3.2 Summary of papers on optimization with multi-Vdd

Clustered voltage scaling (CVS) with voltage level restoration by level converters combined with latches was proposed in 1995 by Usami and Horowitz [222]. Their CVS algorithm proceeded in depth-first search manner from the combinational outputs, assigning VDDH gates to VDDL if they have sufficient slack and only VDDL or level converter latch fanouts. Assigning gates to VDDL was prioritized by the gate load capacitance or the slack in descending order. For two benchmarks in 0.8um process technology, dual supply voltages of 5V and 4V gave power savings of 9% and 18% when using a library with fine grained gate sizes [222]. Their delay models did not include slew nor separate rise/fall delays, and short circuit power was not included. The complexity of CVS is $O(|V|^2)$ [125].

In 1997, Usami et al. proposed an ECVS algorithm with the delay constraint relaxed to allow all flip-flops to be set to LCFFs or VDDL [224]. The combinational gates were examined in reverse topological order. If a VDDH gate had all VDDL fanouts and there was sufficient timing slack, it was set to VDDL. If the gate had some VDDH fanouts, an asynchronous level converter must be inserted. However, one gate may be insufficient to amortize the power overhead for the level converter. Thus in addition to the reduction in power for the gate changing from VDDH to VDDL, the potential power reduction for changing the gate's fanins to VDDL was estimated. They fabricated a dual supply voltage media processor chip in 0.3um technology with VDDH of 3.3V. In the initial circuit, more than 60% of the paths had slack of half the cycle time, suggesting that the initial circuit was not sizing power minimized, thus giving larger power savings with dual Vdd. They achieved on average 28% power savings for the combinational logic with VDDL of 1.9V. The area overhead in the dual Vdd modules was 15% due to level converters, additional VDDL power lines, and reduced cell density due to constrained placement on a VDDH row or VDDL row [224]. The theoretical runtime complexity for this heuristic is also $O(|V|^2)$.

More recent optimization approaches have included gate sizing with CVS and ECVS multiple supply voltage assignment. The theoretical runtime complexity of these algorithms that we detail in Appendix H is $O(|V|^3)$ or worse, which is too slow given the typical size of circuits of interest to designers today.

Kulkarni, Srivastava and Sylvester prioritized gates for ECVS assignment from VDDH to VDDL using the metric $-\Delta P \times \text{slack} / \Delta d$, where the slack and delay are sums of the rise and fall worst slack and worst delay change respectively, and the power and delay overheads for a level converter are included. On each iteration, the gate with maximum sensitivity that won't violate the delay constraints was assigned to VDDL. Moves which increase power were allowed, in the hope that assigning additional gates to VDDL may provide an overall power reduction. The configuration that provided the minimum power was returned. They refer to this algorithm as greedy ECVS, or GECVS. They analyzed the power savings in 0.13um libraries with VDDH=1.2V, VDDL of 0.8V or 0.6V, Vth of 0.12V, wire loads of $C_{\text{wire}} = 3 + 2 \times \# \text{fanouts}$, output port loads of 3fF, and two sizes of the strength 5 asynchronous level converter described in [123]. ECVS gave average power savings of 7.3% versus CVS, and GECVS gave average power savings of 6.5% versus ECVS [124]. Updated results will appear in [125]. At a delay constraint of $1.1 \times T_{\min}$, the average power savings were 9.1% and 14.7% for CVS and GECVS versus the TILOS-like sizer, but only 1.7% and 7.6% respectively versus our better linear programming sizing results. GECVS has runtime complexity of $O(|V|^3)$ [125].

7.3.3 Summary of papers on optimization with multi-Vth and multi-Vdd

For multiple supply voltage assignment, the approach in common to many of the algorithms is starting with a VDDH netlist, prioritizing assignment to VDDL, typically in reverse topological order. The approaches by Srivastava and Kulkarni take this one step further by forcing additional VDDL assignment using slack gained from starting with all gates at low Vth and gate upsizing. Having changed as many gates as possible to VDDL in either a CVS or ECVS methodology, they

pick the best multi-Vdd/low Vth configuration, then look at assigning gates to high Vth [125][188]. The only other multi-Vdd/multi-Vth/sizing approach is a very dubious genetic algorithm [99], which performs worse than TILOS gate sizing [125].

A serious problem with the approaches by Srivastava and Kulkarni is the $O(|V|^3)$ computational complexity. This limits the use of these algorithms on larger circuits. However, they do have the best CVS and ECVS multi-Vdd/multi-Vth/sizing approaches. Their results and algorithmic approaches are detailed below, and we compare our results to their work in Section 7.5.

Srivastava, Sylvester and Blaauw extended the CVS algorithm to include gate sizing and dual Vth assignment [188]. After CVS is performed, gates on critical paths are upsized with a TILOS-like sizer to allow CVS to continue. This process is iterated until no further gates can be assigned to VDDL, then the best configuration found after each iteration is chosen. Gates are then upsized or assigned back to VDDH to provide slack to assign gates to high Vth using TILOS-like sensitivity metrics. Updated results with VDDL of 0.8V appear in [125], with VDDL=0.6V results excluded as those were worse. They used the same setup as for the GECVS approach described earlier, with an additional high Vth of 0.23V. In comparison to the TILOS sizing results for a delay constraint of $1.1 \times T_{\min}$, the average power savings were 18.9%. In comparison to the better gate sizing results from the linear programming approach, the average power savings were only 12.3%. Their CVS approach with gate sizing and dual Vth achieved 10.6% power savings versus CVS.

Kulkarni, Srivastava and Sylvester extended their GECVS approach with dual Vth assignment with gate sizing [125]. After GECVS has been performed, they grow the VDDL clusters by considering only VDDH gates with VDDL fanouts, avoiding the need for additional level converters. The gate with maximum sensitivity is assigned to VDDL, violating a delay constraint. Gates are then upsized to reduce the delay with a TILOS-like metric. This process is iterated and the lowest power configuration found is returned. Remaining slack in the circuit is used to

iteratively assign gates from low V_{th} to high V_{th} to reduce leakage power, again with a TILOS-like metric. Then gates are upsized or assigned to VDDH, if all fanins are VDDH to avoid inserting level converters, to get slack to assign more gates to high V_{th}. More gates are then assigned to high V_{th}. The minimum power configuration found in the whole procedure is picked as the solution [125]. They used the same conditions as described above for their earlier CVS work with dual V_{th} and sizing. At $1.1 \times T_{\min}$, the GECVS approach with gate sizing and dual V_{th} achieved average power savings of 21.6% versus the TILOS-like sizer, or 15.0% versus our LP sizing results. The GECVS approach with gate sizing and dual V_{th} achieved 7.9% power savings versus using GECVS alone.

Previous multi-Vdd algorithms have handled level converter power and delay overheads by iteratively changing gates to VDDL and choosing the best configuration found along the way, or by estimating the power savings of changing multiple gates to VDDL. Both of these approaches can be computationally expensive. We now examine how to account for the voltage level converter delay and power overheads with the linear programming approach that was detailed in Chapter 4.

7.4 Optimizing with multiple supply voltages and multiple threshold voltages

Using multiple threshold voltages does not complicate our optimization approach. The different delay and power – particularly leakage power – must be accounted for, but this just changes the values in the corresponding lookup tables for the standard cell library. Static timing and power analysis do not otherwise change. The power and delay trade-offs for different cell sizes and different threshold voltages for a cell are considered when determining the best alternate cell for a gate to encode in the linear program. Then optimization proceeds normally.

In contrast, using multiple supply voltages not only complicates optimization, but can hinder getting out of local minima. A voltage level converter must be inserted between low supply

voltage and high supply voltage gates. The additional power and delay for level converters must be encoded in the linear program. The level converter overheads create a very “bumpy” optimization surface with many local minima, hindering gates changing from low Vdd to high Vdd or vice versa.

We assume that level converters are placed by the driven input of the VDDH gate to avoid any additional wiring overheads. The same assumption is made in [124] and [125], which we shall compare our results to.

7.4.1 The linear program formulation with multi-Vth and multi-Vdd

With multi-Vth and multi-Vdd, the linear program is the same as used for gate sizing, detailed in Section 4.4.5. The optimization parameters are the same as those used with gate sizing. We restate the linear program here to aid the reader. The linear program for power minimization is

$$\begin{aligned}
 & \text{minimize} \quad \sum_{v \in V} \gamma_v \Delta P_v \\
 & \text{subject to} \quad
 \begin{aligned}
 & t_{\text{Source } u, \text{fall}} = 0, \text{ for all } u \in D \\
 & t_{\text{Source } u, \text{rise}} = 0, \text{ for all } u \in D \\
 & t_{w \text{Sink}, \text{rise}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
 & t_{w \text{Sink}, \text{fall}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
 & 0 \leq \gamma_v \leq 1, \text{ for all } v \in V \\
 & \gamma_v = 0, \text{ for all } v \in D
 \end{aligned}
 \tag{7.1}
 \end{aligned}$$

For all $v \in V \cup D$, $w \in \text{fanout}(v)$, $u \in \text{fanin}(v)$, timing arc constraints:

$$\begin{aligned}
 t_{vw, \text{rise}} & \geq t_{uv, \text{fall}} + d_{uv, \text{rise}} + \gamma_v (\Delta d_{uv, \text{rise}, v} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, v}) \\
 & + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{rise}, x} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, x}) \\
 t_{vw, \text{fall}} & \geq t_{uv, \text{rise}} + d_{uv, \text{fall}} + \gamma_v (\Delta d_{uv, \text{fall}, v} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, v}) \\
 & + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{fall}, x} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, x})
 \end{aligned}$$

And the linear program for delay reduction with a weighting on power is

$$\begin{aligned}
\text{minimize} \quad & \max\{\tau T_{\max}, T\} + k \sum_{v \in V} \gamma_v \Delta P_v \\
\text{subject to} \quad & t_{\text{Source } u, \text{fall}} = 0, \text{ for all } u \in D \\
& t_{\text{Source } u, \text{rise}} = 0, \text{ for all } u \in D \\
& t_{w \text{Sink}, \text{rise}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
& t_{w \text{Sink}, \text{fall}} \leq T_{\max}, \text{ for all } w \in \text{fanin}(\text{Sink}) \\
& 0 \leq \gamma_v \leq 1, \text{ for all } v \in V \\
& \gamma_v = 0, \text{ for all } v \in D
\end{aligned} \tag{7.2}$$

For all $v \in V \cup D$, $w \in \text{fanout}(v)$, $u \in \text{fanin}(v)$, timing arc constraints:

$$\begin{aligned}
t_{vw, \text{rise}} \geq & t_{uv, \text{fall}} + d_{uv, \text{rise}} + \gamma_v (\Delta d_{uv, \text{rise}, v} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, v}) \\
& + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{rise}, x} + \beta_{vw, \text{rise}} \Delta s_{uv, \text{rise}, x}) \\
t_{vw, \text{fall}} \geq & t_{uv, \text{rise}} + d_{uv, \text{fall}} + \gamma_v (\Delta d_{uv, \text{fall}, v} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, v}) \\
& + \sum_{x \in \text{fanout}(v), x \neq w} \gamma_x (\Delta d_{uv, \text{fall}, x} + \beta_{vw, \text{fall}} \Delta s_{uv, \text{fall}, x})
\end{aligned}$$

The different cell choices that are encoded in the linear program affect the ΔP , Δd and Δs values. In particular, the change for adding or removing a level converter as necessary for multi-Vdd is included in these values. The optimization parameters are the same as those used with gate sizing.

7.4.2 Voltage level converter power and delay overheads

If a gate is changed from VDDH to VDDL, it needs level converters to VDDH fanouts, and it no longer needs any fanin level converters. If a gate is changed from VDDL to VDDH, it needs level converters on any VDDL fanins, and no longer needs any fanout level converters. The level converters overheads are not represented directly in the linear program, to avoid adding unnecessary variables and unnecessary constraints. Instead the change in power for adding or removing level converters is added to the change in power for changing the gate's cell, and the delay change is added to the gate's delay change on the appropriate delay constraint edge.

The overheads for level converter flip-flops and “asynchronous” level converters are shown in Figure 7.5 and Figure 7.6. This data is from examination of the cell alternatives for a particular gate before encoding the best alternative for each gate in the LP.

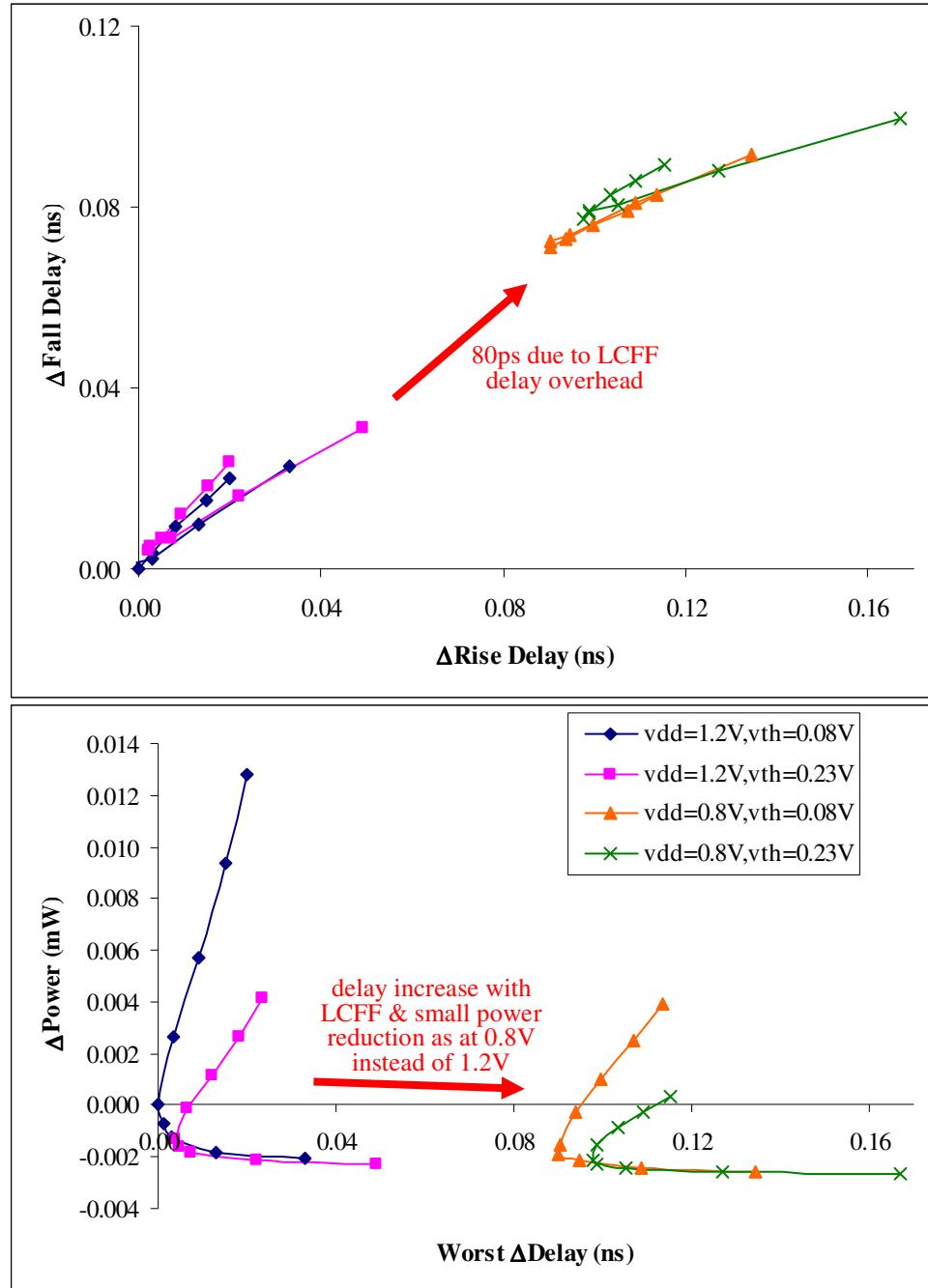


Figure 7.5 These graphs show delay and power trade-offs for alternate cells for a gate, including the level converter flip-flop overhead. We assumed that level converter flip-flops do not consume more power than a normal flip-flop, but that they do impose an 80ps delay penalty. The effect of the 80ps delay penalty is shown here on alternate cell choices for a size X4 Vdd=1.2V/Vth=0.08V inverter that drives an output port in ISCAS'85 benchmark c17. Some power is saved by changing to VDDL and inserting the level converter flip-flop. Each point on a curve represents a different gate size – the largest gates consume the most power and have higher delay than the current cell with size X4 that is at the origin.

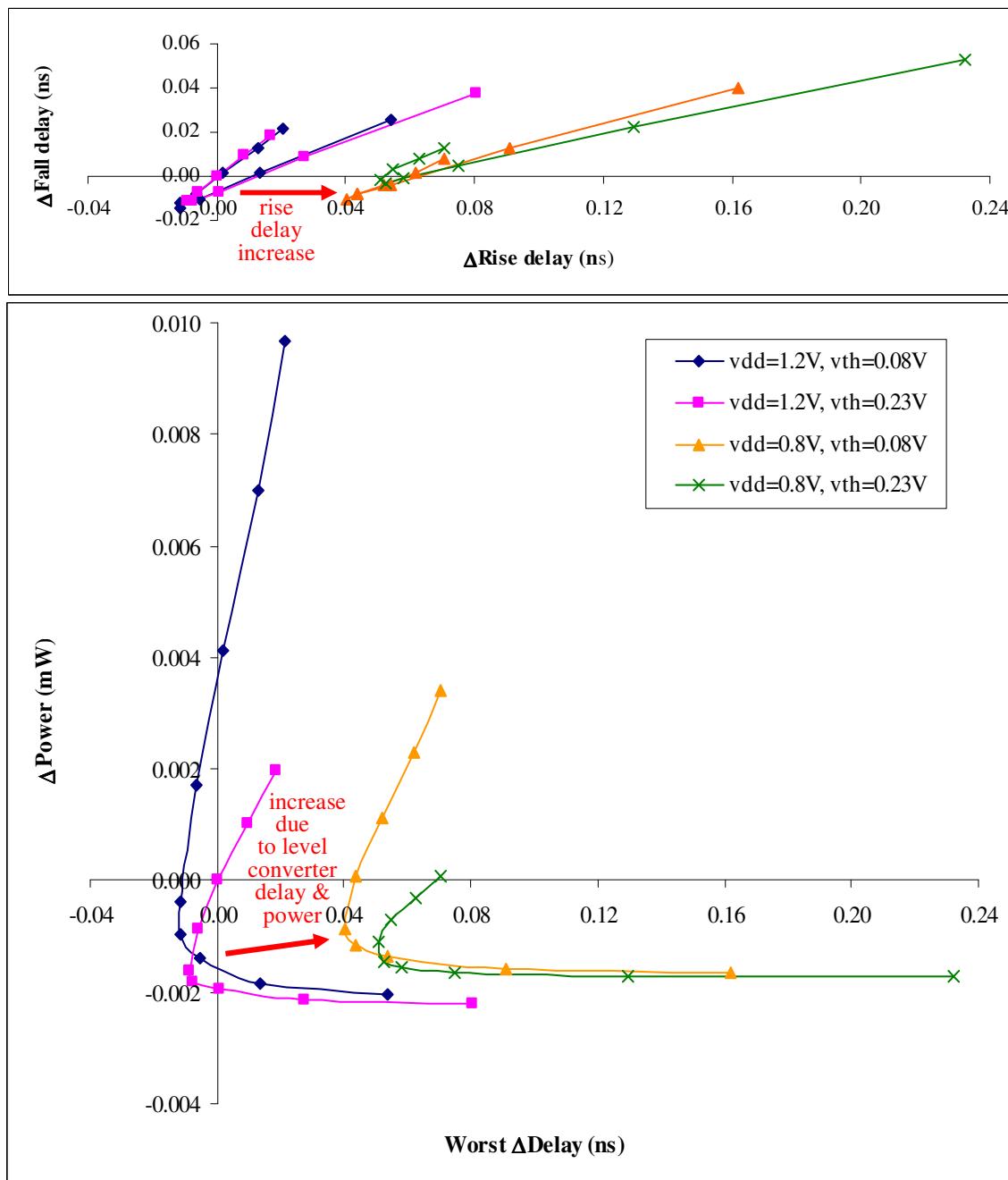


Figure 7.6 These graphs show delay and power trade-offs for alternate cells for a gate, including the asynchronous level converter overhead. These graphs show the alternate cell choices for a Vdd=1.2V/Vth=0.23V drive strength X12 inverter in ISCAS'85 benchmark c5315. The rise delay increases, because of the delay for a falling VDDH input to reach VDDL. In contrast, there is only a small increase in the fall delay. The power overhead for an asynchronous level converter is large, shown by the shift in the curves on the power versus delay graph, and cannot be amortized across a single gate. Each point on a curve represents a different gate size – the largest gates consume the most power.

For level converter flip-flops at the VDDL outputs of combinational logic, we assumed that there is no power overhead. There is some delay overhead – typically about 2 FO4 delays [15], which

corresponds to 80ps in our 0.13um process. The impact of the additional 80ps LCFF fall and rise delay on VDDH and VDDL cell alternatives is shown in Figure 7.5. The 80ps LCFF delay penalty is quite considerable compared to the delay of the inverter. The power savings by changing to VDDL are often less than by downsizing a gate, which argues against prioritizing assignment to low Vdd over gate downsizing.

To analyze the ECVS approach, we used the strength 5 asynchronous level converter shown in Figure 7.7 [123]. This level converter design is a higher speed and more energy efficient modification of a pass gate level converter. Transistor M1 is an NMOS pass gate that isolates the input from the level converter's VDDH supply. Feedback from the inverter, composed of M2 and M3 transistors, to transistor M4 pulls up a VDDL input to VDDH. The logic connected to the transistor gate of M1 serves to raise the transistor gate voltage to VDDL + 0.11V, ensuring that transistor M1 is still off when the input voltage is VDDL to isolate the input from VDDH, but improving the performance of M1 and reducing contention with the inverter's feedback [123]. The two characterized drive strengths of the level converter have similar input capacitance and delay to a $V_{dd}=1.2V/V_{th}=0.23V$ X2 drive strength inverter, but their leakage is about 30x more due to use of low V_{th} transistors and more leakage paths from V_{dd} to ground.

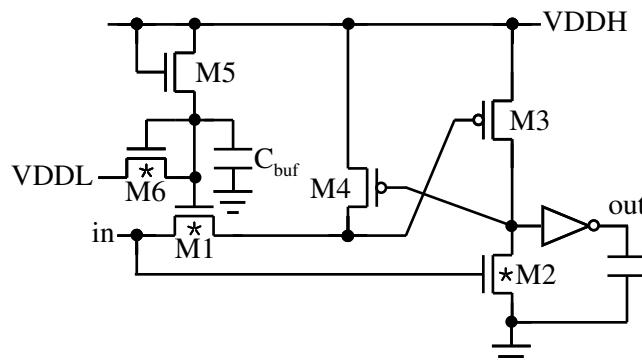


Figure 7.7 The strength 5 asynchronous voltage level converter [123]. The NMOS transistors labeled with ★ have V_{th} of 0.11V. The other NMOS and PMOS transistors have V_{th} of 0.23V and -0.21V respectively. VDDH is 1.2V and VDDL is 0.8V or 0.6V, with transistors and capacitance C_{buf} sized appropriately.

A VDDH input to a VDDL gate reduces the fall delay, but increases the rise delay. For example, an inverter's input falling from $V_{dd}=1.2V$, with slew of 0.15ns, only reaches 0.8V after 0.05ns, which delays when the inverter's output starts to rise by at least 0.05ns. Compared to 0.8V input drivers, using input drivers of 1.2V to 0.8V gates generally reduces the fall delay more than the increase in rise delay for our 0.13um libraries, giving about 1% to 3% net reduction in circuit delay for most of the ISCAS'85 benchmarks. The increased rise delay is apparent for V_{dd} of 0.8V in Figure 7.6, which shows the VDDH and VDDL cell alternatives for an inverter with the overhead for inserting an asynchronous level converter at the output.

Changing more than one gate to VDDL is required to amortize the asynchronous level converter power overhead, as can be seen from the power-delay trade-offs in Figure 7.6. This poses a significant barrier to our optimization formulation, as we pick the best cell alternative to encode in the linear program by considering only a single gate.

7.4.3 Climbing the optimization barrier posed by level converter power overheads

Linear programming CVS results showed some power savings versus only using gate sizing. LP results for ECVS, where asynchronous level converters were allowed, provided minimal (1%) or no additional power savings versus CVS [39]. The optimized circuits did not have asynchronous level converters, because the power overhead for a level converter was too high to amortize across a single gate. The linear programming CVS results were still lower power than the ECVS algorithm proposed by Kulkarni et al. [125] in most cases.

The linear programming optimization had no method of climbing the “hills” posed on the optimization surface by the power overhead for a level converter. In contrast, iteratively forcing gates to VDDL enables hill climbing in an ECVS multi- V_{dd} methodology [125].

We examined the power savings possible when the level converter power and delay overheads were reduced. With reduced level converter overheads, asynchronous level converters were used in the LP optimized circuits. From this came the idea of setting the level converter power overheads to zero, enabling use of the level converters, without violating the delay constraints.

It was essential to not change the level converter delays for two reasons. Firstly, the linear programming optimization approach is not as good at delay reduction, which suggests that trading delay for power reduction, then trying to correct it later would be a mistake. Secondly, increased path delay with the delay overhead of a level converter has a major impact on optimization choices and the power, for example causing gates to be upsized.

Setting level converter power consumption to zero, running the LP approach, correcting the level converter power, then running the LP approach again on the earlier result provided good results with VDDH=1.2V and VDDL=0.6V, using the $\alpha = 1.3$ delay with Equation (5.9) scaling to correct the 0.6V delays. The resulting netlists made use of level converters. The run with level converter power set to zero power results in using a larger number of level converters. The LP run with the correct level converter power then substantially reduces the number of level converters, but more gates remain at VDDL than in a CVS approach – that is VDDL regions are clustered. As detailed in Appendix I.2, ECVS power savings were up to 30% versus CVS with this optimistic delay scaling for Vdd of 0.6V. Changing two gates to VDDL can sometimes be sufficient to justify the insertion of a level converter. The ECVS multi-Vdd optimization flow is shown in Figure 7.8. A simplified example to illustrate what happens is shown in Figure 7.9.

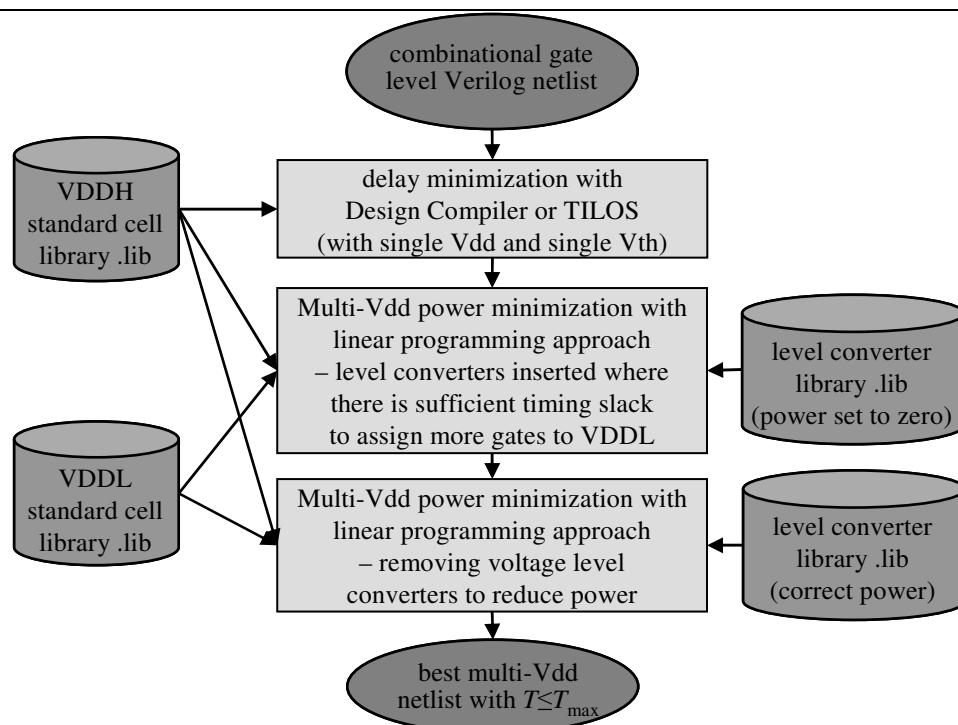


Figure 7.8 An overview of the optimization flow for multi-Vdd with asynchronous level converters. The linear programming flow for each multi-Vdd power minimization run is the same as for sizing in Figure 4.9, with removal and insertion of level converters included when considering alternate cells for a gate.

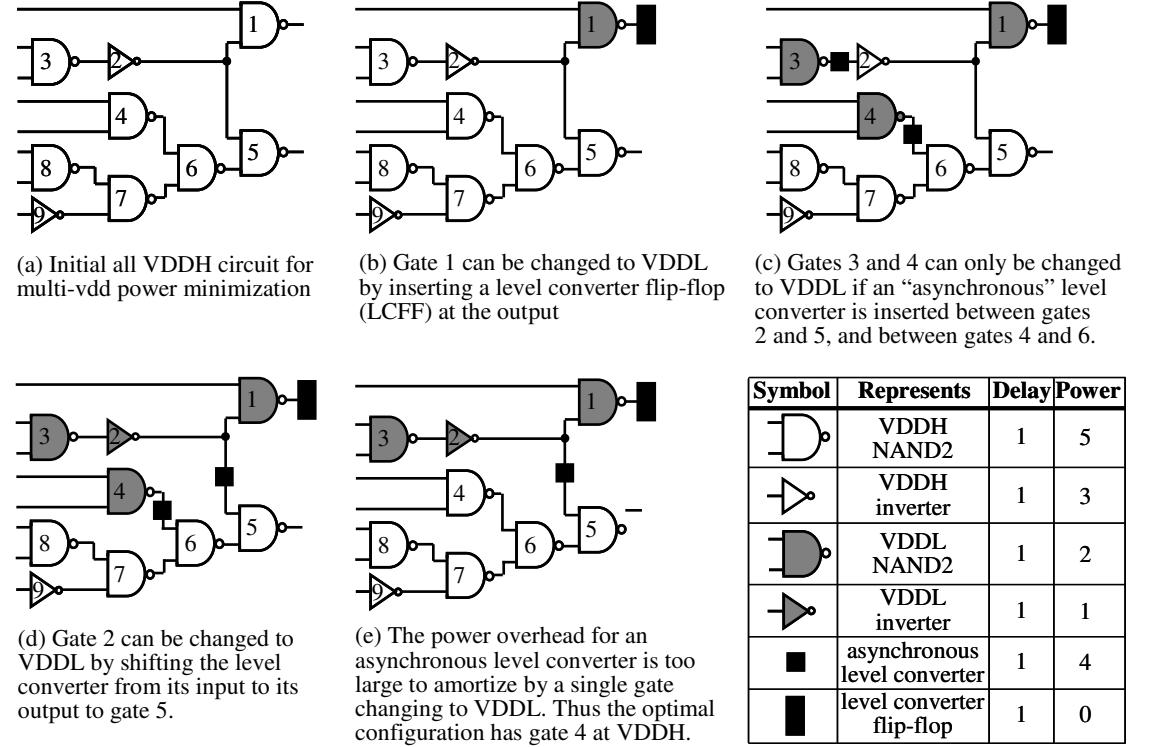


Figure 7.9 This illustrates where level converters may be needed on a simple circuit. The legend at the bottom right lists the delay and power of gates for this example. Suppose we start in (a) with a circuit where all gates are at VDDH, and that the circuit delay constraint is 4 units. We require that the outputs are driven at VDDH. Gate 1 may be changed to VDDL, if we use a level converter flip-flop at its output, shown in (b). The power overhead for an asynchronous level converter is too large to amortize over only a single gate. Thus we temporarily set the level converter power to zero to try and change more gates to VDDL. Changing gates 3 and 4 to VDDL give power savings of 3 each, and we insert level converters at their outputs in (c). Then in (d), we can propagate the level converter from the input of gate 2 to its output, to get additional savings. Now we restore the power overhead for the level converters, and find that it is best to change gate 4 back to VDDH, as shown in (e).

7.4.4 Climbing the optimization barrier posed by level converter delay overheads

The delay overhead for level converters and larger rise delays when $V_{in} > V_{dd}$ can still pose a significant barrier to achieving better results with multi-Vdd, as detailed in Section 7.6.5.2. To illustrate the severity of this problem, consider the circuit in Figure 7.9(e), but now suppose that the delay overhead for an asynchronous level converter is 2 units of delay. The larger level converter delay causes the path from gate 3→2→5 to be 5 units, which violates the delay constraint. We cannot change gate 5 to VDDL to remove the level converter, as this would introduce a level converter from gate 6 to gate 5 which is a critical path. Changing gate 2 to VDDH shifts the level converter to its input, which does not reduce the delay on the path from

gate 3→2→5. The solution is to change both gate 2 and gate 3 to VDDH, but as we determine the best alternatives to encode in the LP for a single gate at a time, we cannot find this solution.

The problem with the multi-Vdd delay overheads can occur in situations where the best solution would be to change multiple gates to VDDH, or in situations where the best solution would be to change multiple gates to VDDL. For the example in Figure 7.9, if the combinational outputs can be driven at VDDL, setting all gates to VDDL would be the lowest power solution. Indeed, setting all gates to VDDL is a better solution in the case described in Section 7.6.5.2, but it is not a solution that we can find without forcing all gates to VDDL in the first place.

It might be possible to surmount the optimization barrier posed by the level converter delay overheads by temporarily setting the delay overhead to zero. Actually setting level converter delays to zero may be unacceptable, because the level converters then act as very effective buffers that can reduce delay by reducing the load on a gate. A better approach would be to set the net delay impact to zero, that is no change in delay of the fanin gate or on the timing arc where the level converter would be inserted. However, this will usually result in too many gates being at VDDL and thus slower, causing the delay constraint to be violated once correct delay overheads are used. Performing delay reduction to meet the delay constraints may then give a suboptimal power result where too many gates have to be upsized to reduce the delay. In particular, it would be helpful to have a better delay minimizer than the linear programming approach currently provides.

Compared to $V_{in}=V_{dd}=VDDL$, for $V_{in}=VDDH > V_{dd}=VDDL$ the rise delay is larger, but the fall delay is reduced more. The optimizer can take advantage of the slight net reduction in circuit delay, if it uses the average change in delay on timing arcs rather than taking the worst case delay on any timing arc when determining the best alternate cell. In practice, using the average delay change on timing arcs to choose the best alternate cell was more likely to cause a delay constraint

violation. To get the best results, the worst case delay change was used by default at the start of optimization, but the average across the timing arcs is used in later iterations when optimization progress is slow to see if any further power savings can be achieved.

Given a good delay reduction approach to fix violated delay constraints, relaxing the delay constraint would allow more gates to be assigned to VDDL, and then delay reduction could be performed to satisfy the tightened delay constraint. Thus the level converter delay penalty is not an optimization barrier to reducing power by assigning more gates to VDDL at a relaxed delay constraint.

Another approach would be to find groups of gates to assign to VDDL or VDDH amortizing the delay penalty across them. However, this will be very expensive computationally.

More gates could be forced to VDDL in the manner of [125] and [188], but these approaches did not find solutions where all gates are at VDDL, so it seems that this may not work. Prioritizing assignment to lower supply voltage over gate downsizing or increasing threshold voltage will be suboptimal, except in situations where there is greater power sensitivity to Vdd, which is often not the case. These approaches are also too computationally expensive as they are order $O(|V|^3)$.

It is not clear how to resolve this problem. Further optimization experiments with the multi-Vdd delay overhead barrier require a better delay minimizer than the linear programming approach. For now, we will examine solutions where all gates are at the lower supply voltage, noting that there may be other intermediate multi-Vdd solutions that would be better that we cannot find due to the delay barrier.

7.5 Comparison of multi-Vdd and multi-Vth results versus University of Michigan

We shall now compare the linear programming results versus the multi-Vdd/multi-Vth/sizing CVS and ECVS results provided by Sarvesh Kulkarni and Ashish Srivastava for Vth values of

0.23V and 0.12V, and Vdd values of 1.2V, 0.8V and 0.6V. The results with VDDL of 0.6V in this section use the delay scaling from 0.8V to 0.6V with Equation (5.10) and α of 1.101.

We used the same libraries and conditions as the University of Michigan work [124][125][188]. The port loads were $3fF$, excluding the additional wire load. The wire loads were $3+2\times\text{num_fanout} fF$, and slews were 0.1ns for the 1.2V input drive ramps. Switching activities were multiplied by a fraction such that leakage was about 20% of total power at $Vdd=1.2V/Vth=0.12V$. We used an 80ps delay overhead for level converter flip-flops, and used the two characterized sizes of the strength 5 asynchronous level converter described in [123]. The ISCAS'85 and Huffman benchmarks for comparison were discussed in Section 4.6.1. Switching activity was multiplied by a fraction such that leakage was about 20% of total power at $Vdd=1.2V/Vth=0.12V$.

For results in this section, the PowerArc characterized 0.13um libraries at 25°C were used. The channel length was 0.13um. There were twelve inverter cell sizes, and seven sizes for NAND2, NAND3, NOR2 and NOR3 logic gates¹. For multi-Vdd with a low supply voltage of 0.8V or 0.6V, we encountered slews that were outside the cell input slew characterization range. To avoid input slews exceeding 1.02ns, the maximum cell capacitance was set to prevent a gate having an output slew of more than 1.02ns.

Their multi-Vth/multi-Vdd/sizing approaches start from a circuit where gates have been upsized from minimum size by their TILOS-like sizer to meet the delay constraint with high Vdd and low Vth. They do not perform any gate downsizing, but may perform further gate upsizing to allow more gates to be changed to low Vdd and high Vth. As the linear programming approach outperforms their TILOS-like sizer, some power savings will be simply due to better gate sizing. Their multi-Vdd approaches climb the voltage level converter power and delay hills in the

¹ The inverter gate sizes were X1, X2, X3, X4, X6, X8, X10, X12, X14, X16, X18 and X20. The other gate sizes were X1, X2, X4, X5, X6, X7 and X8.

optimization surface (see description in sections 7.4.3 and 7.4.4) by forcing as many gates as possible to VDDL. Despite lacking a global circuit view when assigning gates to VDDL, they may achieve lower power by assigning more gates to VDDL, as we do not have any method for climbing the level converter delay hills in the multi-Vdd optimization surface.

The linear programming approach performs better if optimization starts with a delay minimized netlist. Thus, the netlists sized by the TILOS-like sizer for minimum delay ($1.0 \times T_{\min}$) with high Vdd and low Vth were the starting point for the LP runs. The logic gates in the netlists are the same, but the sizes differ from the starting point used for the University of Michigan optimization, which starts with the circuit sized to meet the particular delay constraint.

The TILOS sizing power results were on average 7.6% worse than the LP sizing results, thus we use the LP sizing results to provide the sizing baseline with single Vdd and single Vth, as shown in Table 7.1. Dual Vth with sizing in the LP approach is on average 13.7% lower power than the LP sizing baseline, which is not surprising given that leakage is 20% of the total power with single Vth of 0.12V at the delay constraint of $1.1 \times T_{\min}$. The LP CVS and ECVS Vdd=1.2V&0.8V/Vth=0.23V&0.12V results are respectively on average 17.4% and 21.7% lower power than the LP sizing results. The largest power saving is 37.6% for benchmark c7552. For the lower power ECVS results, using VDDL of 0.6V provides 2% or less power savings versus VDDL of 0.8V. As a lower supply voltage is less robust to noise and will have slower LCFFs, though we assume 80ps LCFF for both VDDL values in these results, VDDL of 0.6V is probably not worthwhile. The LP ECVS results are on average about 5% lower power than the LP CVS results, as shown on the right in Table 7.2.

The University of Michigan sizing/multi-Vth/multi-Vdd results for netlists c432 and c1355 are worse than the LP sizing results, because multi-Vdd is not particularly helpful for these netlists and their TILOS-like sizing results are more than 20% worse than the LP sizing results for these

two benchmarks. The linear programming results are on average about 6% lower power than the University of Michigan CVS and ECVS results, as shown in Table 7.2.

Table 7.1 This table shows the percentage power savings versus the sizing baseline provided with the linear programming (LP) approach with $V_{dd}=1.2V$ / $V_{th}=0.12V$ at $1.1 \times T_{min}$. University of Michigan (UM) results from [125] are reported in the last two columns for their CVS and ECVS approaches that include dual V_{th} and gate sizing – these are suboptimal versus the sizing baseline for c432 and c1355 as highlighted in red. The results from their TILOS-like sizer are reported in the second column.

TILOS	LP						UM	
	CVS			ECVS			CVS	ECVS
V _{th} (V)	0.12	0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12
V _{dd} (V)	1.2	1.2	1.2	1.2, 0.8	1.2, 0.6	1.2, 0.8	1.2, 0.6	1.2, 0.8
Netlist	Power savings versus LP sizing							
c432	-23.0%	0.0%	6.6%	4.5%	9.2%	6.6%	8.7%	-7.4% -11.2%
c880	-1.4%	0.0%	15.9%	22.4%	21.9%	23.7%	22.6%	22.9% 26.3%
c1355	-21.4%	0.0%	5.7%	6.0%	6.7%	8.2%	8.0%	-10.8% -9.9%
c1908	-8.9%	0.0%	9.8%	11.1%	11.2%	14.4%	13.8%	11.1% 7.4%
c2670	-1.8%	0.0%	17.7%	29.0%	31.8%	32.3%	33.1%	26.5% 27.9%
c3540	-3.9%	0.0%	15.1%	16.6%	17.0%	21.8%	20.9%	13.1% 11.7%
c5315	-1.7%	0.0%	15.2%	18.0%	19.5%	26.7%	28.1%	16.9% 25.3%
c7552	-2.1%	0.0%	21.8%	30.3%	28.0%	37.6%	37.1%	22.0% 33.1%
Huffman	-4.4%	0.0%	15.8%	18.5%	19.3%	23.7%	23.9%	16.2% 24.1%
Average	-7.6%	0.0%	13.7%	17.4%	18.3%	21.7%	21.8%	12.3% 15.0%

Table 7.2 This table compares the linear programming CVS and ECVS results versus the University of Michigan CVS and ECVS results respectively, and compares our LP results with gate sizing and dual V_{th} for ECVS versus CVS. Results are highlighted in red where they are worse.

Netlist	LP savings versus UM				LP ECVS savings versus LP CVS	
	LP CVS		LP ECVS		0.23, 0.12	0.23, 0.12
	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12
c432	11.1%	15.4%	16.0%	17.8%	2.1%	-0.5%
c880	-0.6%	-1.2%	-3.5%	-5.1%	1.7%	0.9%
c1355	15.2%	15.8%	16.5%	16.3%	2.3%	1.5%
c1908	-0.1%	0.1%	7.6%	7.0%	3.8%	2.9%
c2670	3.3%	7.1%	6.1%	7.1%	4.7%	1.9%
c3540	4.1%	4.5%	11.4%	10.3%	6.2%	4.6%
c5315	1.4%	3.1%	1.9%	3.8%	10.6%	10.7%
c7552	10.6%	7.7%	6.8%	6.0%	10.5%	12.6%
Huffman	2.8%	3.7%	-0.4%	-0.2%	6.4%	5.7%
Average	5.3%	6.3%	6.9%	7.0%	5.4%	4.5%

If we allow the LP approach to also use the minimum size XL cells, then the LP sizing results are 8.5% lower power. Using XL cells provides another 4.8% power savings with multi-Vth/single Vth, and 2.6% to 0.7% additional savings on average with multi-Vth/multi-Vdd. The impact is less when there are other methods to reduce the power. Compared to the LP sizing results with XL cells, LP multi-Vth/sizing results are 12.3% lower power on average and the LP multi-Vdd/multi-Vth/sizing ECVS results are 15.7% lower power. The maximum power savings with multi-Vdd and multi-Vth are reduced to 28.4% for c7552. The LP CVS and ECVS results with XL cells average about 8% than the corresponding University of Michigan CVS and ECVS results that did not use XL cells. The results with and without XL cells are in Appendix I.1.

Our initial comparison versus the University of Michigan results used the delay scaling from 0.8V to 0.6V with Equation (5.9) and α of 1.3, which is optimistic for the delays at 0.6V. Those results at delay constraints of $1.1 \times T_{\min}$, $1.2 \times T_{\min}$ and $1.5 \times T_{\min}$ are in Appendix I.2. With XL cells allowed for the LP approach but not used by them, our power savings versus their results ranged from 8.2% on average for CVS at $1.1 \times T_{\min}$ to 12.7% on average for ECVS at $1.2 \times T_{\min}$.

In most cases the LP results are lower power, but they are up to 5.1% higher power for c880. In the few cases where the University of Michigan results are better, they were able to assign more gates to VDDL to achieve lower total power. This emphasizes the importance of level converter “delay hill” climbing in the optimization space, which our linear programming approach lacks. Several possible approaches to climbing these delay barriers between local minima with the linear programming approach were discussed in 7.4.4. It is a difficult non-convex optimization problem. We would expect that an optimization approach which uses only a subset of the optimization space (e.g. CVS with level converter flip-flops only) would provide suboptimal or at best equivalent results to approaches with additional options (e.g. ECVS which also has asynchronous level converters). This is not always the case as the optimization approaches are only heuristic, with no guarantee of finding the global minimum, and they can get stuck in local minima in the

optimization space. The benefits of multi-Vdd may be underestimated due to the difficulty of assigning more gates to the low supply voltage VDDL with the level converter delay penalty creating a nonconvex “bumpy” optimization space where it is difficult to get out of local minima.

Our multi-Vdd/multi-Vth power savings of 28% or more versus sizing are sufficient to justify use of multiple supply voltages and multiple threshold voltages. We compare multi-Vdd and multi-Vth results versus single Vdd and single Vth in more detail in the next section. In particular, we start with high Vth netlists for which leakage is 1% of the total power, from which it is more difficult to achieve substantial power savings with multi-Vth.

7.6 Analysis of power savings with multiple Vth and multiple Vdd

Comparisons versus Design Compiler in Section 4.6.2 showed that the linear programming approach provided very good gate sizing results. In Section 7.5, we saw that the LP approach also provides good multi-Vdd/multi-Vth/sizing results, averaging about 10% lower power for both CVS and ECVS multi-Vdd methodologies. By comparison to the good gate sizing baseline with single Vdd and single Vth, we can now carefully analyze the power savings that may be possible with multi-Vdd and multi-Vth in addition to gate sizing.

With the large amount of data presented in this section, we italicize the more significant results.

7.6.1 General experimental conditions

The starting gate sizes for multi-Vth and multi-Vdd optimization are the Design Compiler netlists that were sized to minimize delay with the $V_{dd}=1.2V/V_{th}=0.23V$ 0.13um PowerArc characterized library. As discussed at the start of Section 5.2, using high Vdd and low Vth cells provides the minimum delay starting point for optimization. We used fewer gate sizes for results in this section. There were nine inverter sizes, and four sizes for NAND2, NAND3, NOR2 and

NOR3 logic gates¹. Using more gate sizes did not significantly change the results from Design Compiler or our LP approach, as only the larger gate sizes were not included. Inclusion of the larger gate sizes does not change the results significantly as the larger gates with substantial power consumption are seldom used in the power minimized netlists. For multi-Vdd with a low supply voltage of 0.8V or 0.6V, we encountered slews that were outside the cell input slew characterization range. These cell sizes were characterized with input slew of up to 1.8ns. To avoid input slews exceeding 1.8ns, the maximum cell capacitance was set to prevent a gate having an output slew of more than 1.8ns.

As for the sizing results in Chapter 4 and the multi-Vdd comparison in Section 7.5, the port loads were 3fF, not including the wire load, and wire loads were $3+2\times\text{num_fanout}$ fF. Switching activities were multiplied by a fraction such that leakage was about 1% of total power at $V_{dd}=1.2V/V_{th}=0.23V$. Input slew was set to 0.1ns. The input drive ramps have voltage swing from 0V to 1.2V, except in Section 7.6.5 for the single 0.8V Vdd results where we look at the impact of using 0.8V drivers. With multi-Vdd, the optimization is not allowed to set the input drivers to VDDL to further reduce power.

7.6.2 Experimental conditions for multi-Vth comparison

Power savings are compared at a tight delay constraint for high Vth to avoid exaggerating savings versus low Vth, where leakage and thus total power are substantially higher. Starting with all gates at low Vth, it is easy to reduce leakage by going to high Vth, as many gates are not on timing critical paths. The power savings are less when starting with all gates at high Vth, because using low Vth causes a substantial increase in leakage power that can only be justified by gate downsizing on timing critical paths, or using the resulting timing slack to reduce Vdd.

¹ The inverter gate sizes were XL, X1, X2, X3, X4, X8, X12, X16 and X20. The other gate sizes were XL, X1, X2, X4.

We shall examine multiple threshold voltages with three possible thresholds voltages: 0.23V, 0.14V, and 0.08V. The leakage at Vth of 0.14V and 0.08V is respectively about 10 \times and 50 \times than at 0.23V, but they also provide a substantial delay reduction versus Vth=0.23V, ranging from 12% to 43% less depending on Vdd. There is minimal benefit for choosing Vth higher than a value that results in leakage being 1% of total power – the power savings are at best 1%, and in practice less due to the reduced slack for gate downsizing. Thus we consider a scenario with 1% of total power being leakage at high Vth.

The library characterization was at the typical process corner for both NMOS and PMOS and nominal operating conditions, 25°C and no reduction in Vdd. Higher temperature and a fast process corner would give much more leakage at Vth of 0.08V, as a small reduction in threshold voltage increases leakage considerably. Thus Vth of 0.08V may be too low, but similar analysis can be done at worst case process corners, using the fast corner for worst case leakage, and will provide similar results with higher Vth values to limit the leakage.

The delay constraints were $1.0 \times T_{\min}$ and $1.2 \times T_{\min}$ for multi-Vth results, where T_{\min} is the minimum delay for the Design Compiler delay minimized netlists at Vdd=1.2V/Vth=0.23V. Using a lower threshold voltage provides sufficient timing slack to get good multi-Vth results with the linear programming approach at a delay constraint of $1.0 \times T_{\min}$ for Vdd=1.2V/Vth=0.23V. We expect that using a lower Vth may provide most benefit at the tight $1.0 \times T_{\min}$ delay constraint, as the additional slack allows gates to be downsized, backing away from the sharp rise in dynamic power on the gate sizing power versus delay “banana” curves. Analysis is also performed at $1.2 \times T_{\min}$, as this is where we found the greatest power savings with geometric programming optimization of multi-Vdd and multi-Vth for benchmarks c499 and c880, and where other researchers have performed multi-Vdd analysis [124][188].

7.6.3 Experimental conditions for multi-Vdd comparison

We shall examine three possible supply voltages: 1.2V, 0.8V and 0.6V. Earlier multi-Vdd research suggested as a rule of thumb to use a low supply voltage of about 70% of VDDH, while some more recent research has suggested that VDDL should be 50% of VDDH [124]. Thus if VDDH is 1.2V, then VDDL should be 0.8V or 0.6V. While VDDL of 50% of VDDH has been strongly supported by the multi-Vdd research at the University of Michigan, this conclusion may be erroneous due to the incorrect characterization at $V_{dd}=0.6V$, causing the delay penalty for low V_{dd} to be substantially underestimated. To estimate the delay at $V_{dd}=0.6V$, we perform delay scaling with Equation (5.9) and $\alpha = 1.66$.

There is a 50% delay increase when using $V_{dd}=0.6V$ even with $V_{th}=0.08V$, though $V_{dd}=0.6V/V_{th}=0.08V$ does reduce power by about 65% versus $V_{dd}=1.2V/V_{th}=0.23V$ from the analysis without gate sizing in Table B.7. In comparison, $V_{dd}=0.8V/V_{th}=0.08V$ has only a 10% delay increase and reduces power by 28%. For multi-Vdd, the smaller delay penalty at $VDDL=0.8V$ will allow it to be used for more gates in the circuit than $VDDL=0.6V$. A smaller VDDL delay penalty leaves more slack for power minimization by gate down sizing or increasing V_{th} .

We used a delay constraint of $1.2 \times T_{min}$ to look at the benefits of multi-Vdd, where T_{min} is the minimum delay for the Design Compiler delay minimized netlists at $V_{dd}=1.2V/V_{th}=0.23V$. There is sufficient slack for the linear programming approach to work well at $V_{th}=0.23V$. 10% to 20% relaxed delay constraints from the TILOS sized netlists were used in [124] and [188] to allow sufficient slack for good power savings with multi-Vdd. The Design Compiler delay minimized netlists average 22% faster than the TILOS delay minimized netlists. Thus, there may be significantly less timing slack at $1.2 \times T_{min}$ for the Design Compiler netlists than for the TILOS netlists. However, it is difficult to compare the netlists as they differ because delay minimization

in Design Compiler used both technology mapping and gate sizing, whereas TILOS was limited to gate sizing. Reducing V_{th} below 0.23V provides multi-Vdd scenarios with more timing slack.

We must account for the delay and power overheads for restoring a low voltage swing signal. Compared to high speed flip-flops, a level converter flip-flop has a delay overhead of about 2 FO4 delays [15], which corresponds to 80ps in our 0.13um process. Making the same assumption as in [125] and [188] for results in Section 7.6.4, we assume an 80ps delay overhead for voltage level restoration with an LCFF from 0.6V or 0.8V to 1.2V and no power overhead.

An LCFF delay overhead of 0ps is appropriate if comparing to the typical D-type flip-flops in an ASIC standard cell library [110], rather than high speed pulsed flip-flops. Results with 0ps LCFF delay overhead are discussed in Section 7.6.5. The additional timing slack permits lower power results with Vdd of 0.8V. With 80ps LCFFs, VDDH of 1.2V is required to meet the delay constraints with sufficient timing slack to reduce power.

For ECVS, we used two characterized drive strengths of the strength 5 asynchronous level converters [123]. This higher speed and more energy efficient modification of a pass gate level converter was described in Section 7.4.2.

We make the same assumption as made by Kulkarni and Srivastava et al. in [124], [125] and [188] that level converters are placed next to the input pin of the gate that they drive, and that there are no additional wiring overheads. This assumption is optimistic, unless there is a standard cell library with level converters incorporated into the logic cells. However, there are level converter designs incorporating additional logic, for example the single supply level converters in [165], so this assumption may be reasonable.

The optimization approach is only a heuristic, so in some cases the results found given a larger possible state space, for example dual Vdd versus single Vdd, can be worse. Section 7.6.5 will dwell on multi-Vdd results with 0ps LCFF delay overhead that are substantially worse than using

a single Vdd of 0.8V. As we are interested in looking at the benefits of multi-Vdd and multi-Vth, a suboptimal result obscures the benefits. Instead, if there was a better solution found with a subset of the Vdd or Vth values, that solution has been tabulated, except for the multi-Vdd results in Section 7.6.5 where their suboptimality is discussed.

7.6.4 Results with 80ps level converter flip-flop delay overhead

We begin analysis assuming an 80ps LCFF delay overhead for voltage level restoration at the outputs. This applies to the multi-Vdd results and to the single Vdd results where the supply voltage has been scaled to 0.8V. We have made the same assumptions as in [125][188] to compare our results to those papers. In Section 7.6.5, we analyze results with a 0ps LCFF delay overhead, which substantially improves the single Vdd=0.8V results.

7.6.4.1 Impact of multi-Vth with single Vdd at $1.0 \times T_{\min}$

To examine the benefits of using multiple supply and threshold voltages, we must first provide a sizing only baseline with single Vdd and single Vth. At a delay constraint of $1.0 \times T_{\min}$, the only possible choice of supply voltage if only a single Vdd is used is 1.2V, as the delay is too large otherwise. For Vth of 0.23V, there is no timing slack, and the linear programming approach cannot minimize power without violating the delay constraint. Thus the Design Compiler power minimization results are reported for Vth=0.23V at the $1.0 \times T_{\min}$ delay constraint.

The best single Vth gate sizing results at $1.0 \times T_{\min}$ are with Vth of 0.14V, reducing the power on average by 12.0% from the Vth=0.23V sizing results. The lower threshold voltage gives sufficient slack for gate downsizing to reduce the dynamic power without resulting in excessive leakage, unlike using Vth of 0.08V which is 19.7% higher power on average as listed in Table 7.3.

The largest power saving with dual Vth versus the single Vth=0.14V baseline is 7.0% with Vth of 0.23V and 0.14V, and on average they provide 5.2% power savings. The additional leakage with

low Vth of 0.08V is too great to justify using it with dual Vth, though sparing use of it on the critical path for *triple Vth* provides up to 5.1% power savings versus dual Vth.

In Appendix J.2, we analyze the benefits of multi-Vth when leakage is a larger portion of the total power as will be the case in deeper submicron technologies where leakage is an increasing problem. In that scenario with leakage being 8% of total power at $Vdd=1.2V/Vth=0.23V$ at $1.0 \times T_{min}$, the optimal single Vth was 0.23V due to substantial leakage for lower Vth values. Average power savings with dual Vth were 4.6%, but the maximum power savings of 11.4% was higher – gate sizing with the additional slack from low Vth of 0.14V can still provide significant reduction in dynamic power. Triple Vth provided 1% or less additional power savings because of the excessive leakage with Vth of 0.08V.

Table 7.3 This table compares dual Vth and triple Vth sizing power minimization results versus the best sizing only results with single Vth of 0.14V. The delay constraint was $1.0 \times T_{min}$ and Vdd was 1.2V for all these results. Results for single Vth of 0.23V are from Design Compiler (DC); the other results are from the linear programming approach (LP).

Vth (V)	Single			Dual			Triple	
	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14
Power savings vs. $Vdd=1.2V/Vth=0.14V$								
Netlist	DC	LP						
c17	-13.7%	0.0%	-20.1%	7.0%	-8.0%	0.0%	7.0%	
c432	-19.6%	0.0%	-17.9%	2.5%	-5.6%	0.8%	3.3%	
c499	-20.9%	0.0%	-17.1%	7.0%	-0.2%	2.8%	8.1%	
c880	-4.2%	0.0%	-20.8%	5.1%	4.6%	2.8%	10.0%	
c1355	-26.1%	0.0%	-17.2%	4.6%	-1.8%	2.6%	5.9%	
c1908	-11.1%	0.0%	-19.9%	6.1%	-1.6%	1.0%	6.1%	
c2670	-13.5%	0.0%	-23.5%	5.2%	0.5%	1.4%	7.1%	
c3540	-17.4%	0.0%	-20.9%	4.5%	-1.7%	0.8%	5.9%	
c5315	-6.4%	0.0%	-24.1%	6.2%	0.0%	1.4%	6.8%	
c6288	-12.7%	0.0%	-16.9%	2.4%	-3.5%	1.6%	3.8%	
c7552	-8.0%	0.0%	-18.3%	6.2%	0.8%	3.2%	7.6%	
Average	-14.0%	0.0%	-19.7%	5.2%	-1.5%	1.7%	6.5%	

Table 7.4 This table compares dual Vth and triple Vth sizing power minimization results versus the best sizing only results with single Vth. The delay constraint was $1.2 \times T_{\min}$ and input drivers were ramps with voltage swing from 0V to 1.2V. There was an 80ps LCFF delay overhead at the outputs for the Vdd=0.8V results. All these results are for the linear programming approach.

	Single		Dual		Triple		Single		Dual		Triple	
Vth (V)	0.23	0.14	0.23	0.23	0.23	0.23	0.14	0.14	0.23	0.14	0.23	0.14
Vdd (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	0.8	0.8	0.8	0.8
Netlist	Power savings vs. single Vdd/single Vth baseline											
Netlist												
c17	-8.8%	0.0%	-20.4%	5.0%	-2.3%	0.0%	5.0%	5.0%	failed delay constraint			
c432	-3.5%	0.0%	-24.3%	6.7%	1.0%	0.0%	6.7%	6.7%	failed delay constraint			
c499	-7.2%	0.0%	-30.0%	1.8%	-5.7%	-3.7%	1.8%	1.8%	failed delay constraint			
c880	0.0%	-3.1%	-31.5%	5.0%	1.9%	-2.3%	5.0%	5.0%	failed delay constraint			
c1355	-1.8%	0.0%	-26.2%	4.5%	-0.8%	0.0%	4.7%	4.7%	failed delay constraint			
c1908	0.0%	-0.2%	-29.3%	7.1%	3.3%	-0.8%	7.7%	7.7%	-78.0%	-45.7%	-18.1%	-8.3%
c2670	0.0%	-2.8%	-31.1%	4.3%	2.5%	-2.8%	4.6%	4.6%	failed delay constraint			
c3540	-0.4%	0.0%	-27.9%	6.9%	2.3%	0.8%	7.5%	7.5%	-68.3%	-68.3%	-8.8%	-8.8%
c5315	0.0%	-4.9%	-36.0%	3.7%	1.4%	-4.7%	3.9%	3.9%	-53.3%	-53.3%	-53.3%	-53.3%
c6288	-1.0%	0.0%	-24.9%	3.9%	1.5%	0.7%	5.2%	5.2%	-12.5%	-2.2%	7.8%	7.8%
c7552	0.0%	-1.3%	-29.0%	5.6%	2.3%	-0.9%	5.6%	5.6%	failed delay constraint			
Average	-2.1%	-1.1%	-28.2%	5.0%	0.7%	-1.2%	5.2%	5.2%	-53.0%	-42.4%	-18.1%	-15.7%

7.6.4.2 Impact of multi-Vth with single Vdd at $1.2 \times T_{\min}$

With a relaxed delay constraint of $1.2 \times T_{\min}$, in some cases single Vth of 0.23V produced the best results for gate sizing and in other cases Vth of 0.14V was the best choice, as shown in Table 7.4. The best single Vth/single Vdd/gate sizing results were with Vdd of 1.2V, though the relaxed delay constraint does allow Vdd of 0.8V with Vth reduced to 0.08V in some cases. The best of these gate sizing only results are used for a baseline to compare multi-Vth against.

Dual threshold voltages of 0.23V and 0.14V with Vdd of 1.2V provide the best dual Vth power savings, except for benchmark c6288, averaging 5.0% lower power than the single Vth baseline.

The leakage with Vth of 0.08V is too high to justify its use with Vdd of 1.2V. Triple Vth provides at most 1.3% power savings versus dual Vth.

Interestingly, optimization of c6288 with single Vdd of 0.8V and Vth values of 0.14V and 0.08V achieves the largest dual Vth power savings of 7.8% versus the single Vth gate sizing baseline,

and this result is 4.1% lower power than the multi-Vth results with Vdd of 1.2V. This is the only case where the multi-Vth results with Vdd of 0.8V achieve lower power. With Vdd of 0.8V, low Vth of 0.08V is essential to try and meet the delay constraint. For the single Vth results, using only Vth of 0.08V results in too much leakage and worse total power. For those gates with sufficient slack to change to high Vth, the higher Vth of 0.14V provides about a 5x reduction in leakage power. The 80ps LCFF delay overhead is only 2% of the delay constraint for c6288, but for the other netlists it is 6% or more of the delay constraint, leaving less slack for gates to be downsized or changed to high Vth. This is why the results for c6288 with Vdd of 0.8V are different.

For the Vdd=0.8V/multi-Vth results where the delay constraints were not violated, the delay reduction phase of the LP approach did manage to meet the delay constraint after iterations of the power reduction phase, which was not the case for the Vdd=0.8V/single Vth results. To reduce delay with Vth, a gate can be changed back to low Vth, which only slightly increases the capacitive load on fanin gates and doesn't reduce their speed substantially. Whereas to reduce delay with sizing, a gate must be upsized, which substantially increases the capacitive load on the fanins and increases their delay. Thus the delay reduction phase can reduce delay better with multi-Vth than with gate sizing alone.

7.6.4.3 Summary of multi-Vth results with 80ps LCFF delay overhead

Using dual threshold voltages provides only a power saving of up to 11.4%, which does not justify the additional processing costs and yield impact. Using three different threshold voltages provides no significant additional benefits. However, we will revisit this with Vdd of 0.8V and 0ps LCFF delay overheads in Section 7.6.5, where dual Vth is found to provide larger power savings.

In the next section we examine the power savings with multiple supply voltages at $1.2 \times T_{\min}$. Then we look at using multiple supply voltages in conjunction with multiple threshold voltages in Section 7.6.4.5. We might anticipate that multi-Vth is more beneficial with multi-Vdd, as the lower Vth can help provide sufficient slack to change more gates to low Vdd.

7.6.4.4 Impact of multi-Vdd with single Vth at $1.2 \times T_{\min}$

The largest power saving with ECVS dual Vdd versus the single Vdd baseline was 13.9% for c2670 with Vdd values of 1.2V and 0.6V and Vth of 0.14V, as shown in Table 7.5. The largest power saving with CVS dual Vdd was 12.2% was for the same benchmark and Vdd values, but Vth was 0.23V. On average ECVS provides only 1.2% power savings versus CVS, though the maximum power saving is 7.3%, excluding the suboptimal results with Vth of 0.08V that are detailed in Appendix J.4. Dual Vdd with single Vth provided no power savings for c17 and c499.

Comparing the best CVS dual Vdd/single Vth/sizing results against the baseline, *CVS dual Vdd gives on average 2.5% power savings*. The results with VDDL of 0.8V and 0.6V were quite similar, indicating a somewhat flat optimization space in terms of the choice for VDDL. In most cases, the best choice for a single Vth with dual Vdd was 0.14V, as it provides more slack for gates to change to VDDL than a Vth of 0.23V. *The best ECVS dual Vdd with single Vth results versus the baseline give an average 4.1% power saving.*

The dual Vdd/single Vth results range from 6.6% better to 11.2% worse than the single Vdd/dual Vth results at $1.2 \times T_{\min}$, and are 1% better on average. The power savings depend on the particular design and the optimizer, for example the LP approach works well with multi-Vth but has problems in some cases with multi-Vdd. Depending on the power savings and the design cost, it may be preferable to use only dual Vth, only dual Vdd, or both. However, the maximum power savings of 7.8% with single Vdd/dual Vth and 13.9% with dual Vdd/single Vth are probably insufficient to justify the additional design cost.

We now look at the benefits of multi-Vdd with multiple threshold voltages. Using a low Vth provides slack for greater use of low Vdd, and voltage level converter designs may utilize more than one threshold voltage. For example, the strength 5 level converters from [123] use Vth of 0.23V and 0.11V.

Table 7.5 This table compares CVS and ECVS dual Vdd/single Vth/sizing power minimization results versus the sizing only baseline results in Table 7.4. At the bottom are shown the ECVS power savings versus CVS.

Vth (V)	Single							
	0.23	0.14	0.23	0.14	0.23	0.14	0.23	0.14
Vdd (V)	CVS dual Vdd				ECVS dual Vdd			
	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6
Netlist	Power savings vs. single Vdd/single Vth baseline							
c17	-8.8%	0.0%	-8.8%	0.0%	-8.8%	0.0%	-8.8%	0.0%
c432	-2.4%	0.6%	-2.9%	0.0%	-2.4%	0.6%	-2.9%	0.0%
c499	-7.1%	0.0%	-7.1%	0.0%	-7.1%	0.0%	-7.1%	0.0%
c880	2.3%	0.0%	3.0%	1.4%	2.3%	6.2%	3.0%	4.2%
c1355	-1.8%	0.0%	-1.8%	0.0%	-1.8%	0.8%	-1.5%	0.0%
c1908	1.4%	2.2%	0.4%	2.6%	1.6%	3.4%	1.9%	3.6%
c2670	9.8%	7.8%	12.2%	11.3%	10.0%	10.4%	12.2%	13.9%
c3540	1.8%	2.1%	2.2%	1.9%	3.0%	7.6%	2.2%	5.5%
c5315	3.3%	-0.1%	3.2%	0.4%	6.6%	7.1%	3.9%	7.7%
c6288	0.4%	1.3%	-0.1%	1.2%	0.4%	1.3%	-0.1%	1.2%
c7552	2.3%	0.6%	1.7%	0.7%	2.3%	3.7%	1.7%	3.2%
Average	0.1%	1.3%	0.2%	1.8%	0.5%	3.7%	0.4%	3.6%

Netlist	ECVS power saved vs. CVS			
c17	0.0%	0.0%	0.0%	0.0%
c432	0.0%	0.0%	0.0%	0.0%
c499	0.0%	0.0%	0.0%	0.0%
c880	0.0%	6.2%	0.0%	2.8%
c1355	0.0%	0.8%	0.3%	0.0%
c1908	0.1%	1.2%	1.4%	1.0%
c2670	0.2%	2.8%	0.0%	3.0%
c3540	1.1%	5.6%	0.0%	3.7%
c5315	3.4%	7.2%	0.7%	7.3%
c6288	0.0%	0.0%	0.0%	0.0%
c7552	0.0%	3.1%	0.0%	2.5%
Average	0.4%	2.5%	0.2%	1.8%

Table 7.6 This table compares CVS and ECVS dual Vdd/multi-Vth/sizing power minimization results versus the sizing only baseline results in Table 7.4. At the bottom are shown the ECVS power savings versus CVS. Suboptimal dual Vdd/dual Vth results with low Vth of 0.08V are omitted here – see Appendix J.5 for those results.

	Dual	Triple	Dual	Triple	Dual	Triple	Dual	Triple
Vth (V)	0.23 0.14 0.08							
	CVS dual Vdd				ECVS dual Vdd			
Vdd (V)	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6

Netlist	Power savings vs. single Vdd/single Vth baseline							
	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%
c17	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%
c432	6.9%	7.7%	7.5%	7.5%	7.0%	7.7%	7.5%	7.5%
c499	1.8%	1.9%	1.8%	1.8%	1.8%	1.9%	1.8%	1.8%
c880	6.9%	6.9%	7.0%	7.0%	10.4%	10.4%	7.3%	9.3%
c1355	4.7%	4.7%	4.8%	4.9%	4.7%	4.9%	4.8%	4.9%
c1908	8.4%	8.6%	10.2%	10.2%	9.4%	9.4%	10.2%	10.2%
c2670	13.8%	14.0%	16.7%	16.8%	15.1%	16.4%	18.6%	18.6%
c3540	8.1%	8.8%	8.5%	9.0%	10.6%	10.8%	10.4%	10.9%
c5315	6.6%	6.8%	6.7%	6.7%	11.0%	11.5%	11.9%	13.9%
c6288	4.7%	5.2%	5.1%	5.2%	4.7%	5.2%	5.1%	5.1%
c7552	6.2%	6.6%	6.8%	7.0%	6.9%	7.7%	8.1%	8.1%
Average	6.6%	6.9%	7.3%	7.4%	7.9%	8.3%	8.2%	8.7%

Netlist	ECVS power savings vs. CVS			
	0.0%	0.0%	0.0%	0.0%
c17	0.0%	0.0%	0.0%	0.0%
c432	0.1%	0.0%	0.0%	0.0%
c499	0.0%	0.0%	0.0%	0.0%
c880	3.8%	3.8%	0.3%	2.5%
c1355	0.0%	0.2%	0.0%	0.0%
c1908	1.0%	0.8%	0.0%	0.0%
c2670	1.5%	2.8%	2.3%	2.2%
c3540	2.7%	2.1%	2.1%	2.1%
c5315	4.8%	5.0%	5.5%	7.7%
c6288	0.0%	0.0%	0.0%	0.0%
c7552	0.8%	1.2%	1.4%	1.2%
Averag	1.3%	1.5%	1.1%	1.4%

7.6.4.5 Impact of multi-Vdd with multi-Vth at $1.2 \times T_{min}$

In most cases, the best Vdd values were 1.2V and 0.6V for dual Vdd/dual Vth, and the best Vth values were 0.23V and 0.14V. The largest power saving seen with ECVS dual Vdd/dual Vth versus the single Vdd/single Vth baseline was 18.6%, as shown in Table 7.6, and the average power saving with the best dual Vdd/dual Vth results for each benchmark is 8.5%. *The best dual*

Vdd/dual Vth power saving versus single Vdd/dual Vth is 6.9%, and the results are 4.6% better on average. The best dual Vdd/dual Vth power saving versus dual Vdd/single Vth is 15.0%, and the results are 3.4% better on average. The largest power saving for dual Vdd/triple Vth versus dual Vdd/dual Vth was 2.3% for c5315.

The dual Vdd/dual Vth results for c6288 are suboptimal, with 2.9% worse power than the $V_{dd}=0.8V/V_{th}=0.14V\&0.08V$ result of 2.72mW. The optimizer fails to find the better solution with all gates at Vdd of 0.8V. We will see more suboptimal cases and discuss this further in Section 7.6.5, where we reduce the LCFF delay overhead of 0ps.

Results from these small benchmarks suggest that dual Vth with dual Vdd doesn't provide substantial benefits over using single Vth with dual Vdd, and that the processing costs for additional mask layers would not be justified. An additional Vth value doesn't provide much power saving: either by using a second higher Vth to reduce leakage, or a second lower Vth to provide more timing slack to reduce Vdd.

Comparing ECVS versus CVS for the best dual Vth values of 0.23V and 0.14V, ECVS provided up to 5.5% power savings. This may be insufficient improvement to justify use of asynchronous level converters, given that they are not as robust to noise as level converter flip-flops, and thus require tighter design constraints on voltage IR drop and more careful noise analysis.

The gate sizing results for the linear programming approach in Chapter 4 had average power savings of 16.6% versus Design Compiler for the delay constraint of $1.2T_{min}$. Thus LP gate sizing provides about twice the average improvement of 8.5% seen with dual Vdd/dual Vth/sizing, without any additional processing costs for multi-Vth or area overhead for multi-Vdd. Given the additional design costs, use of dual Vdd and dual Vth appear dubious.

In the next section with 0ps LCFF delay overheads, we will examine how optimal these multi-Vdd results are, and see situations where multi-Vth can provide larger power savings.

7.6.5 Results with 0ps level converter flip-flop delay overhead

Thus far, an 80ps LCFF delay penalty has been assumed for voltage level restoration to 1.2V at the primary outputs if the driving gate has a 0.8V or 0.6V supply. However, the delay of a level converter flip-flop is comparable to that of a typical D-type flip-flop in an ASIC standard cell library [110], though slower than fast pulsed flip-flops. Thus a 0ps delay penalty is appropriate if comparing to D-type flip-flops.

The output signals may also not require voltage level restoration. For example, the whole circuit may use a 0.8V supply voltage. In this case, there will be some additional delay due to using registers with 0.8V supply. A typical D-type flip-flop used for a register in an ASIC has delay of 2 to 4 FO4 delays, corresponding to 80ps to 160ps in this 0.13um process. From Appendix B.1, voltage scaling from $V_{dd}=1.2V/V_{th}=0.23V$ to $V_{dd}=0.8V/V_{th}=0.14V$ increases the delay by about 15%. So the flip-flops may be 12ps to 24ps slower, which is substantially less than an 80ps delay penalty.

In this section, we look at LP results with single $V_{dd}=0.8V$ and 0ps LCFF delay overhead. Then we will examine why optimization with multi- V_{dd} has problems finding result as good as these.

Table 7.7 This table compares single Vdd=0.8V results versus the baseline of the best sizing only results with 1.2V input drivers from Table 7.4 and the second column here. At the bottom left, the gate sizing results with Vdd of 0.8V are compared against the best gate sizing results from Table 7.4. The input drivers had voltage swing of 0.8V or 1.2V. At the bottom right are shown the power savings with 0.8V drivers versus 1.2V drivers.

Vth (V)	with 1.2V drivers			with 0.8V drivers		
	Single	Dual	Triple	Single	Dual	Triple
	0.08	0.08	0.08	0.08	0.08	0.08
Netlist	Power savings vs. single Vdd/single Vth baseline with 1.2V drivers					
c17	failed delay constraint			3.3%	3.3%	24.4%
c432	-8.6%	-8.7%	2.3%	3.4%	5.5%	6.0%
c499	-2.7%	0.7%	9.2%	10.1%	7.8%	10.0%
c880	-5.7%	8.9%	16.7%	16.7%	8.8%	19.5%
c1355	-15.5%	-1.1%	6.7%	6.7%	7.5%	9.4%
c1908	-11.8%	7.8%	15.5%	16.5%	9.0%	15.1%
c2670	0.0%	9.4%	16.3%	16.6%	3.2%	22.2%
c3540	0.0%	9.2%	16.9%	17.3%	8.7%	17.8%
c5315	0.0%	14.0%	18.1%	18.6%	13.2%	21.4%
c6288	0.0%	2.6%	9.5%	9.5%	5.9%	5.9%
c7552	0.0%	10.5%	18.0%	18.5%	11.9%	18.1%
Average	-4.4%	5.3%	12.9%	13.4%	7.7%	22.5%
						23.3%

Netlist	Saved vs. Vdd=1.2V single Vth baseline			Saved vs. 1.2V drivers		
	c432	c499	c880	c1355	c1908	c2670
c432	-8.6%			13.0%	13.4%	12.0%
c499	-2.7%			10.3%	9.4%	11.6%
c880	-5.7%			13.8%	11.6%	13.2%
c1355	-15.5%			19.9%	10.4%	12.2%
c1908	-11.8%			18.6%	8.0%	10.1%
c2670	4.1%			3.2%	14.1%	14.7%
c3540	0.3%			8.7%	9.5%	8.9%
c5315	3.9%			13.2%	8.7%	11.7%
c6288	3.3%			5.9%	3.4%	5.3%
c7552	3.5%			11.9%	8.4%	8.6%
Average	-2.9%			Average	11.8%	9.7%
						10.8%
						11.3%

7.6.5.1 Impact of multi-Vth with Vdd of 0.8V at $1.2 \times T_{min}$

Vth of 0.08V is necessary with Vdd=0.8V to meet the delay constraint. With 1.2V drivers, the single Vdd=0.8V/Vth=0.08V results are up to 4.1% better than the Vdd=1.2V gate sizing baseline, but on average are 2.9% worse as shown in Table 7.7. The increased rise delay with 1.2V input

swing to the 0.8V gates prevents the delay constraint being satisfied for c17, where the 0.033ns for the input to fall from 1.2V to 0.8V is 29% of the 0.112ns delay constraint¹.

When voltage is scaled from $V_{dd}=1.2V/V_{th}=0.23V$ to $V_{dd}=0.8V/V_{th}=0.08V$, there is about a $26\times$ increase in leakage, but the dynamic power is reduced substantially by about a factor of $2.2\times$ from the analysis that excluded gate sizing in Table B.7. As the dynamic power was 99% of the total power at $V_{dd}=1.2V/V_{th}=0.23V$, this trade-off can be worthwhile. In the absence of gate sizing, $V_{dd}=0.8V/V_{th}=0.08V$ reduced power on average by 28% versus $V_{dd}=1.2V/V_{th}=0.23V$, but with gate sizing included the average power saving is only 10.4% with 0.8V input drivers. The extra timing slack at $V_{dd}=1.2V/V_{th}=0.23V$ allows more gate downsizing, thus reducing V_{dd} provides less power savings.

With 1.2V drivers, the $V_{dd}=0.8V/dual\ V_{th}$ results with V_{th} values of 0.14V and 0.08V are up to 18.1% lower power than the baseline, and average 12.9% less. In comparison for single V_{dd} of 1.2V at $1.2\times T_{min}$, dual V_{th} was only 5.0% better than the baseline on average, and at best 7.1% better. Thus the dual V_{th} results at V_{dd} of 0.8V are much better. The low V_{th} of 0.08V provides sufficient slack for V_{dd} to be reduced to 0.8V, and the higher V_{th} of 0.14V is used where possible to reduce leakage. Comparing the $V_{dd}=0.8V$ results, adding the high V_{th} of 0.14V allows power to be reduced by 16.5% on average versus the single $V_{th}=0.08V$ results. This may be sufficient power savings to justify the additional process costs for dual V_{th} . At $V_{dd}=0.8V$, gates with V_{th} of 0.14V were only about 15% slower than at V_{th} of 0.08V, compared to V_{th} of 0.23V which had 50% larger delay. Consequently, V_{th} of 0.14V is a better choice for high V_{th} than 0.23V. Comparing triple V_{th} and dual V_{th} results at $V_{dd}=0.8V$ in Table 7.7, there is at most 2.3% power savings by using the third threshold voltage.

¹ Delay for the input to fall from 1.2V to 0.8V was calculated from the 0.1ns input slew of the drivers.

Until now, we have assumed that the input drivers had voltage swing of 1.2V. Using 0.8V input drivers reduces the dynamic power to switch capacitances driven by the primary inputs. The results with 0.8V input drivers average 10.9% lower power than the results with 1.2V drivers in Table 7.7. With 0.8V gates and 0.8V input drivers, the single V_{th} of 0.08V results average 7.7% lower power than the other gate sizing results. *In comparison to these better gate sizing results, results for 0.8V gates and 0.8V drivers with dual V_{th} of 0.14V and 0.08V average 16.0% lower power, and the maximum power saving is 26.2%*. This shows the full extent of power savings that might be achieved by scaling V_{dd} from 1.2V to 0.8V. These savings are comparable to what we achieved with the gate sizing approach versus Design Compiler.

The multi-V_{dd} results were restricted to having 1.2V input drivers, so it is not fair to compare them to V_{dd}=0.8V results with 0.8V drivers. With 0ps LCFF delay overhead and 1.2V drivers, we will compare the multi-V_{dd} results with single V_{dd}=0.8V results in the next subsection.

7.6.5.2 Multi-V_{dd} results are suboptimal versus using single V_{dd} of 0.8V

On average there is only a 0.9% reduction in power for the multi-V_{dd}/single V_{th} results with 0ps LCFF delay overhead versus assuming 80ps delay overhead, and the maximum reduction in power is 8.8% as detailed in Appendix J.7. In comparison, including the V_{dd}=0.8V/V_{th}=0.08V results in the sizing baseline reduced the power on average by 1.5%. Thus the multi-V_{dd}/single V_{th} results with 0ps LCFF delay overhead are somewhat worse versus the sizing baseline, as listed in Table 7.8. The maximum power saving with multi-V_{dd}/single V_{th} versus the baseline is 10.2%, and the average saving of the best multi-V_{dd}/single V_{th} results for each benchmark is only 3.5%. Indeed, all the multi-V_{dd}/single V_{th} results are 1.7% higher power or worse than the V_{dd}=0.8V/V_{th}=0.08V result for c6288.

For a single V_{dd}/dual V_{th} baseline to compare multi-V_{dd}/multi-V_{th} against, V_{dd}=0.8V/V_{th}=0.14V&0.08V with 1.2V input drivers achieves lower power than using single

$V_{dd}=1.2V$ /dual V_{th} for all benchmarks, except for c432 where $V_{dd}=1.2V/V_{th}=0.23V\&0.14V$ achieves 5.0% lower power and c17 where the delay constraint could not be met with $V_{dd}=0.8V$ and 1.2V input drivers. Excluding c17 where all gates are assigned to $V_{DDH}=1.2V$ anyway, the ECVS multi- V_{dd} /multi- V_{th} results with 0ps LCFF delay overhead average 8.9% worse than the $V_{dd}=0.8V/V_{th}=0.14V\&0.08V$ results with 1.2V drivers and 0ps LCFF delay overhead. As we saw earlier, the ECVS results average 0.9% better than the corresponding CVS results, and are up to 6.6% better. Multi- V_{dd} /multi- V_{th} does save up to 5.7% power for benchmark c432 for which single V_{dd} of 0.8V was suboptimal, but in almost all other cases the results are worse than the single 0.8V V_{dd} /multi- V_{th} results, as shown in Table 7.9. The solutions with V_{DDL} of 0.8V are clearly suboptimal as the single $V_{dd}=0.8V$ solution is in a subspace of the multi- V_{dd} solution space. On average, the ECVS $V_{dd}=1.2V\&0.8V/V_{th}=0.14V\&0.08V$ results are suboptimal by 9.8% compared to the $V_{dd}=0.8V/V_{th}=0.14V\&0.08V$ results. The $V_{dd}=1.2V\&0.8V/\text{triple } V_{th}$ results are suboptimal by 6.0% on average versus the $V_{dd}=0.8V/\text{triple } V_{th}$ results.

The multi- V_{dd} /multi- V_{th} results with 0ps LCFF delay overhead are only 1.1% better on average than the corresponding results with 80ps, though up to 7.0% power is saved in some cases, as detailed in Appendix J.8. This is somewhat surprising, as 80ps amounts to 6% to 10% of the delay constraint for all benchmarks except c17 which has a much smaller delay constraint and c6288 which has a much larger delay constraint. From previous analysis of power/delay trade-offs, at a tight delay constraint we would expect power savings in the range of up to twice the additional slack, namely something in the range of 6% to 20%. However, most of the gates are still at V_{DDH} and many paths have all gates at 1.2V, thus not terminating in an LCFF and not benefiting from any additional slack. With dual V_{dd} of 1.2V and 0.6V, the largest number of gates that have 0.6V supply is 33%, and on average only 8% of the gates have 0.6V supply. For dual V_{dd} of 1.2V and 0.8V, the largest number of gates with 0.8V V_{dd} is 27%. The majority of

the timing slack from relaxing the delay constraint from $1.0 \times T_{\min}$ to $1.2 \times T_{\min}$ has been used for gate downsizing, rather than reducing Vdd.

Several different approaches were tried to improve the suboptimal multi-Vdd results, as detailed in Appendix J.8. The closest results started optimization with all gates at Vdd=0.8V and Vth=0.08V, for which the dual Vdd=1.2V&0.8V results were only 2.8% worse¹ on average than the single Vdd=0.8V results. In general with the 0ps LCFF delay overheads, dual Vdd with VDDH of 1.2V gives suboptimal results versus voltage scaling the single Vdd to 0.8V.

This concludes the analysis of using multi-Vdd and multi-Vth in comparison to using only a single Vdd and single Vth. We saw some power savings, but they were not comparable to the power savings from sizing alone with the linear programming approach versus TILOS, except for Vdd=0.08V/Vth=0.14V&0.08V with 0ps LCFF delay overheads. We now look at what additional computation time is required for multi-Vdd and multi-Vth optimization.

Table 7.8 This table compares CVS and ECVS dual Vdd/single Vth/sizing power minimization results versus the best single Vdd/single Vth results with 1.2V drivers from Table 7.4 and Table 7.7.

Single												
Vth (V)	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08
CVS dual Vdd with 1.2V drivers						ECVS dual Vdd with 1.2V drivers						
Vdd (V)	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6
Netlist	Power savings vs. single Vdd/single Vth baseline with 1.2V drivers											
c17	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%
c432	-2.3%	0.7%	-24.3%	-3.0%	0.0%	-23.6%	-2.3%	0.7%	-21.1%	-3.0%	0.0%	-22.4%
c499	-7.1%	0.0%	-24.9%	-7.1%	0.0%	-30.0%	-7.1%	0.0%	-24.9%	-7.1%	0.0%	-29.7%
c880	4.2%	7.6%	-14.6%	3.0%	5.4%	-15.0%	4.2%	7.6%	-9.4%	3.0%	5.5%	-10.1%
c1355	-1.8%	0.2%	-25.1%	-1.8%	0.0%	-24.1%	-1.8%	0.9%	-17.9%	-1.5%	0.0%	-21.7%
c1908	2.2%	3.0%	-23.3%	2.4%	3.5%	-23.3%	2.7%	4.9%	-21.1%	3.5%	5.7%	-17.3%
c2670	6.0%	4.7%	-20.6%	8.7%	7.5%	-16.3%	6.1%	6.6%	-15.4%	8.7%	10.2%	-9.7%
c3540	1.6%	2.0%	-24.6%	1.9%	2.6%	-25.0%	2.7%	7.4%	-15.0%	2.0%	5.2%	-14.0%
c5315	2.4%	1.3%	-24.6%	0.5%	-2.1%	-26.9%	2.7%	3.3%	-19.0%	0.8%	5.9%	-14.0%
c6288	-2.8%	-1.7%	-27.9%	-3.3%	-2.1%	-26.8%	-2.8%	-1.7%	-27.5%	-3.3%	-2.1%	-24.1%
c7552	-0.9%	-1.1%	-29.6%	-1.9%	-2.5%	-29.3%	-0.9%	2.2%	-20.9%	-1.9%	0.1%	-19.0%
Average	-0.7%	1.5%	-23.6%	-0.9%	1.1%	-23.7%	-0.5%	2.9%	-19.3%	-0.7%	2.8%	-18.4%

¹ Ignoring the very suboptimal outlier result for c1908 in Table J.12.

Table 7.9 This table compares CVS and ECVS dual Vdd/multi-Vth/sizing power minimization results versus the single Vdd=0.8V/dual Vth=0.14V&0.08V results with 1.2V drivers from Table 7.7. The few cases where there are power savings are highlighted in blue.

	Dual		Triple		Dual		Triple	
Vth (V)	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
	0.14		0.14	0.14	0.14		0.14	0.14
	0.08		0.08	0.08	0.08		0.08	0.08
CVS dual Vdd with 1.2V drivers								
Vdd (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
	0.8		0.8	0.8	0.6		0.6	0.6
Netlist	Power savings vs. Vdd=0.8V/Vth=0.14V&0.08V with 1.2V drivers							
c432	4.9%	-1.2%	-1.6%	5.5%	5.7%	-1.3%	-1.4%	5.7%
c499	-5.8%	-13.5%	-10.1%	-5.5%	-8.1%	-14.5%	-10.1%	-6.2%
c880	-5.4%	-8.5%	-10.7%	-5.4%	-8.9%	-10.3%	-11.0%	-5.4%
c1355	-1.5%	-7.4%	-5.3%	-0.1%	-2.0%	-8.1%	-6.4%	-1.7%
c1908	-7.5%	-13.0%	-14.2%	-6.9%	-6.3%	-13.1%	-14.8%	-6.3%
c2670	-7.0%	-10.0%	-13.9%	-6.8%	-1.4%	-5.7%	-10.1%	-1.4%
c3540	-10.3%	-16.2%	-17.9%	-9.8%	-8.1%	-15.7%	-17.8%	-7.5%
c5315	-11.5%	-17.4%	-18.4%	-11.5%	-11.9%	-17.5%	-21.3%	-9.4%
c6288	-8.2%	-12.3%	-12.5%	-8.3%	-8.3%	-12.3%	-13.4%	-8.1%
c7552	-16.5%	-22.0%	-23.6%	-15.3%	-16.2%	-21.0%	-23.8%	-15.5%
Average	-6.9%	-12.1%	-12.8%	-6.4%	-6.6%	-11.9%	-13.0%	-5.6%
	Dual		Triple		Dual		Triple	
Vth (V)	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
	0.14		0.14	0.14	0.14		0.14	0.14
	0.08		0.08	0.08	0.08		0.08	0.08
ECVS dual Vdd with 1.2V drivers								
Vdd (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
	0.8		0.8	0.8	0.6		0.6	0.6
Netlist	Power savings vs. Vdd=0.8V/Vth=0.14V&0.08V with 1.2V drivers							
c432	4.9%	-1.2%	-1.6%	5.5%	5.7%	-1.3%	-1.4%	5.7%
c499	-5.8%	-13.5%	-10.1%	-5.5%	-8.1%	-14.5%	-10.1%	-6.2%
c880	-2.5%	-8.5%	-3.3%	-2.7%	-8.9%	-10.3%	-10.1%	-5.4%
c1355	-0.2%	-7.4%	-3.8%	0.4%	-2.0%	-7.8%	-6.4%	-1.7%
c1908	-7.3%	-13.0%	-11.4%	-6.9%	-6.3%	-13.1%	-10.7%	-5.8%
c2670	-5.9%	-8.9%	-10.2%	-4.2%	-1.4%	-5.7%	-6.1%	-0.8%
c3540	-7.9%	-13.5%	-11.1%	-6.4%	-7.5%	-14.7%	-12.3%	-7.5%
c5315	-11.5%	-14.0%	-14.7%	-10.8%	-11.8%	-14.7%	-13.7%	-9.4%
c6288	-8.2%	-12.3%	-12.4%	-8.3%	-8.3%	-12.3%	-11.5%	-8.1%
c7552	-16.1%	-22.0%	-19.8%	-15.0%	-16.2%	-21.0%	-20.0%	-15.1%
Average	-6.0%	-11.4%	-9.8%	-5.4%	-6.5%	-11.5%	-10.2%	-5.4%

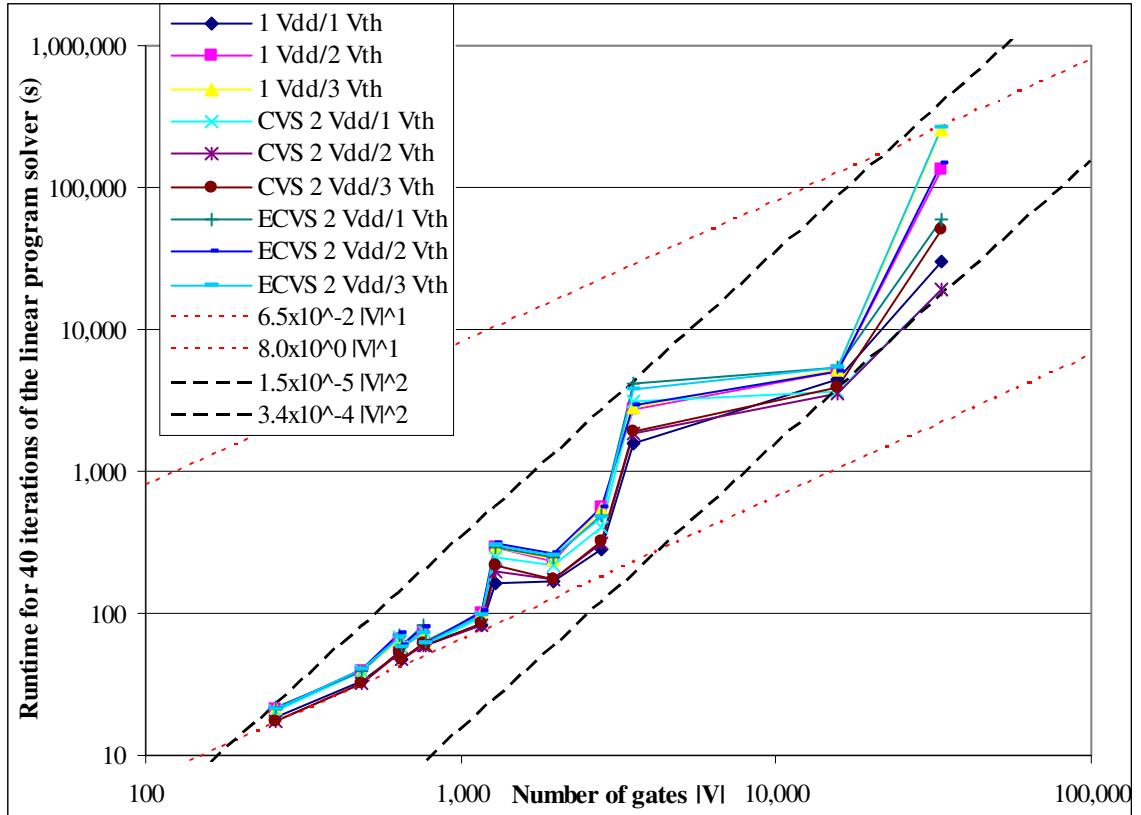


Figure 7.10 This log-log graph shows the runtime for 40 iterations of the linear program solver. As illustrated with the lines showing linear and quadratic runtime growth, the linear program solver runtimes grow between linearly and quadratically with circuit size.

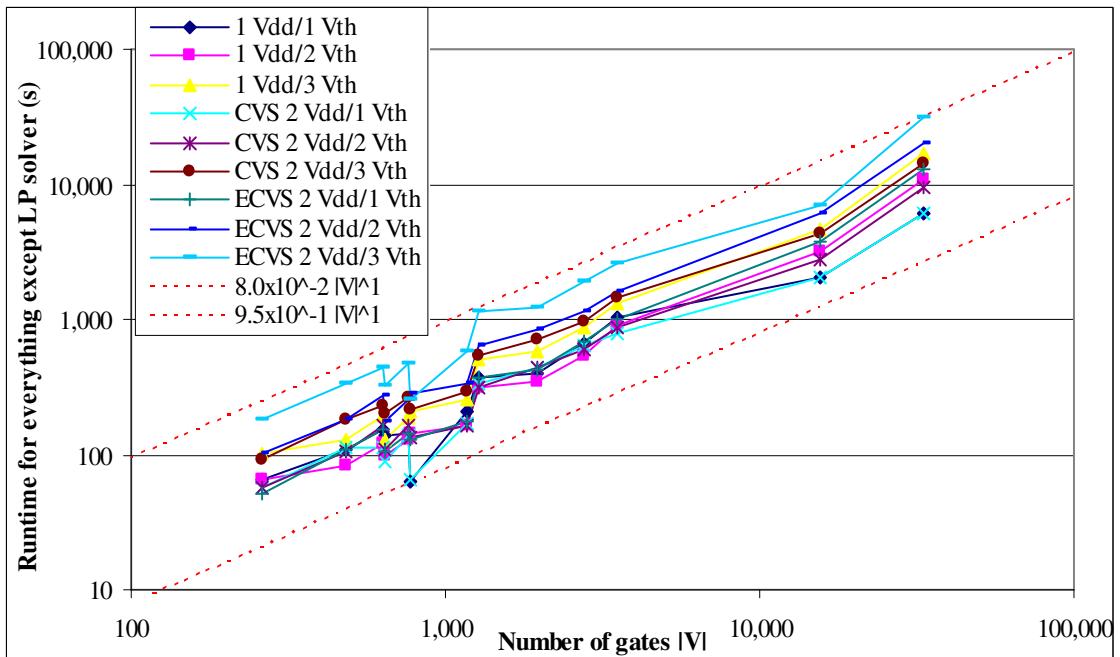


Figure 7.11 This graph shows the runtime for 40 iterations, excluding the runtime for the linear program solver, on a log-log scale. The runtimes excluding the linear program solver grow linearly with circuit size.

7.7 Computational runtimes with multi-Vdd and multi-Vth

The LP solver runtimes in Figure 7.10 are not substantially affected by the number of Vdd and Vth values, as that does not affect the number of variables or constraints in the linear program¹. Rather, the net runtime for multiple LP solver iterations depends on the timing slack and Vdd and Vth values available, as this determines the number of iterations for power minimization versus delay reduction. As observed in Section 4.7, power minimization with the LP solver typically takes twice as long as delay reduction.

Excluding the LP solver, the CVS runtimes are not much worse than the single Vdd runtimes, as Vdd changes are considered only for gates on the VDDH to VDDL wavefront, that is gates with all fanouts at VDDL or primary outputs and all fanins at VDDH. The ECVS runtimes are up to 2.2 times the CVS runtimes, as changing Vdd is considered for all gates, doubling the setup runtime to consider the different Vdd alternatives, and there are additional computation overheads for insertion and removal of level converters. Considering alternate cells for a gate for multi-Vth can double setup runtimes for dual Vth and triple setup runtimes for triple Vth. The computation runtimes excluding the linear program solver are shown in Figure 7.11.

With a CVS multi-Vdd methodology, each iteration allows at most one additional level of logic to change from high Vdd to low Vdd, or vice versa. Thus for a very deep circuit, for example c6288 with 113 logic levels, more iterations can be required for CVS multi-Vdd. The number of iterations required to get within 1% of the best solution varies substantially depending on the Vdd and Vth values available and the corresponding timing slack. Up to about 40 iterations is necessary in a few cases to get within 1% of the best solution when there is substantial timing

¹ If cells are ordered by their sensitivity to reduce power and no cell is strictly better than an alternative, then in the worst case the number of LP or TILOS iterations depends on the number of alternate cells. This worst case situation does not occur in practice, as many alternate cell choices are strictly worse than other choices. For example, starting from the largest inverter cell size at $Vdd=1.2V/Vth=0.08V$ with data for dual Vdd and dual Vth that was shown in Figure 7.5 and Figure 7.6, there are 36 alternate inverter cells, but two or three cells dominate as ordered by the LP sensitivity metric and their power. More LP iterations are required in practice for iterations of power minimization then delay reduction to meet the delay constraint.

slack, for example with V_{th} of 0.08V. For many cases the number of iterations required is still 20 or less, as for gate sizing.

As the delay overhead for level converters is substantial, an ECVS multi-Vdd methodology can also take more iterations for Vdd changes to propagate. The number of iterations required to get within 1% of the best solution are in the same range for ECVS and CVS. However, for ECVS we start with zero power for the level converters, and then run further optimization iterations with the correct level converter power. Usually less than 20 additional iterations are required to get within 1% of the best solution, as primarily gates on the boundary of the VDDH and VDDL regions change Vdd to reduce the number of level converters.

Runtimes for the Huffman, and larger SOVA_EPR4 and R4_SOVA benchmarks are included in these figures, though leakage was not normalized to 1% of total power at $V_{dd}=1.2V/V_{th}=0.23V$ for these runs. The larger benchmarks have longer runtimes, so a less exhaustive range of results were collated, and the limited multi-Vdd and multi-Vth results for these benchmarks have not been included in this chapter. The results were similar to the ISCAS'85 benchmarks. For example with 0ps LCFF delay overheads, using a single Vdd of 0.8V gave better results than dual Vdd with appropriate threshold voltages. Benchmark c17 was omitted due to its small size and small runtimes. The runtimes are listed in Appendix J.9.

In summary, the runtimes for setting up the linear program grow linearly with the number of alternate cells available. For sizing only runs in Chapter 4, we saw that only about 20 iterations were necessary to find a good solution. However, with multi-Vth and CVS multi-Vdd there are cases where up to 40 iterations were necessary to converge within 1% of the best solution, and ECVS may take up to 60 iterations in total.

On the smaller benchmarks, the University of Michigan CVS and ECVS approaches are substantially faster. However, as their runtime growth is $O(|V|^3)$ and our worst case runtime

growth is $O(|V|^2)$, the LP approach runtimes were faster for c6288 and larger benchmarks [125][Sarvesh Kulkarni private communication].

7.8 Summary

This chapter examined the power savings that the linear programming approach can achieve with single and multiple supply and threshold voltages. In comparison to the best optimization approaches without major simplifying assumptions that we know of for multi-Vdd/multi-Vth/sizing [125][188], our LP approach reduces power by 5% to 13% and has runtime growth of $O(|V|)$ to $O(|V|^2)$ rather than $O(|V|^3)$, so the LP approach is also more applicable to larger benchmarks.

As shown by the geometric programming results on small benchmarks in Chapter 6, scaling a single Vdd and single Vth optimally can provide significant power savings, reducing the power savings that may be found with multi-Vdd and multi-Vth. Versus the nominal $Vdd=1.2V/Vth=0.23V$, using $Vdd=1.2V/Vth=0.14V$ reduces power on average by 12.0% at $1.0 \times T_{min}$, and $Vdd=0.8V/Vth=0.08V$ reduces power by 10.8% on average at $1.2 \times T_{min}$ assuming 0ps level converter delay flip-flop overhead and that input drivers are also scaled down to 0.8V.

In our detailed analysis of multi-Vdd and multi-Vth, the largest power savings were with $Vdd=0.8V/Vth=0.14V \& 0.08V$ versus the optimal choice of $Vdd=0.8V/Vth=0.08V$ at $1.2 \times T_{min}$ assuming 0ps LCFF delay overhead and 0.8V input drivers. In this scenario, dual Vth reduced power on average by 16.0% and the maximum power saving was 26.2%. Triple Vth provided at most 5.1% power savings versus using dual Vth in the scenarios that we have considered, which is not sufficient to justify use of a third threshold voltage.

In spite of achieving better multi-Vdd/multi-Vth results than other known approaches, the LP multi-Vdd results with VDDL of 0.8V were suboptimal by up to 23.6% versus using a single Vdd of 0.8V with Vth values of 0.14V and 0.08V assuming 0ps LCFF delay overhead at $1.2 \times T_{min}$. To

improve on this, a stronger delay reduction approach along the lines of TILOS could be integrated with the LP power minimization approach.

The largest power saving with ECVS multi-Vdd versus single Vdd/single Vth at $1.2 \times T_{\min}$ assuming 80ps LCFF delay overhead and 1.2V input drivers was 13.9%, but the average power saving was only 4.1%. In that scenario, ECVS with asynchronous voltage level converters averages only 1.2% power saving versus CVS, where only level converter flip-flops are allowed, though the maximum power saving with ECVS versus CVS is 7.3%.

We saw larger power savings with ECVS versus CVS and multi-Vdd/multi-Vth versus single Vdd/single Vth in the comparisons to the University of Michigan results, but $Vdd=1.2V/Vth=0.12V$ was not the optimal choice for single Vdd/single Vth in that scenario.

The multi-Vth power savings may be enough to justify the additional process costs for a second threshold voltage. The weak multi-Vdd results do not justify use of multi-Vdd on these small benchmarks. More power savings may be available with multi-Vdd for an optimization approach that can get around the level converter delay overheads that pose a significant barrier to finding the global minimum.

For a larger sequential circuit with different delay constraints on portions of the circuit, different supply voltages may be justified. For example, Vdd of 1.2V at $1.0 \times T_{\min}$ and Vdd of 0.8V at $1.2 \times T_{\min}$. In the event that multiple supply voltages are justified by use at a module level in this manner, our results suggest that another 5% to 10% power saving may be available via a gate-level CVS or ECVS multi-Vdd assignment methodology.

7.8.1 Detailed commentary

Performing gate-level optimization with multiple threshold voltages is no different to performing gate sizing, providing leakage is accounted for. As discussed in Section 7.3.1, the better multi-Vth results from previous research have used TILOS-like optimizers. These optimizers proceed in

a greedy manner, picking the gate with the best power or area versus delay tradeoff to change, and iterating. In Chapter 4, we showed that the linear programming optimization approach achieved on average 12% to 22% lower power than two TILOS-like optimizers. Thus we did not dwell on further comparisons versus TILOS optimizers for multi-Vth.

Multi-Vdd complicates optimization because voltage level converters must be inserted to restore the voltage swing of low Vdd gates that drive high Vdd gates, to avoid substantial static current. The power and delay overheads for level converters create a very “hilly” non-convex optimization space where it is easy to get stuck in a local minimum.

In the comparison of the linear programming results versus the University of Michigan, the maximum power saving with multi-Vdd and multi-Vth was 28.4% versus gate sizing with the minimum XL cell size included, Vdd of 1.2V and Vth of 0.12V. These power savings may be an overestimate as the comparison was versus a starting point with leakage of 20% of total power at low Vth of 0.12V. To address this, a variety of scenarios were then examined in comparison to a starting point with leakage 1% of total power at high Vth of 0.23V.

The multi-Vdd/multi-Vth results were compared against the best single Vdd/single Vth results at a given delay constraint. Our gate sizing results provided a strong baseline to compare results against. At a delay constraint of $1.0 \times T_{\min}$, Vdd of 1.2V with Vth of 0.14V provided the best single Vdd/single Vth results. The average power savings with sizing and Vdd=1.2V/dual Vth were at best 5.2%. We did not examine dual Vdd at $1.0 \times T_{\min}$ as there was insufficient slack.

At a delay constraint of $1.2 \times T_{\min}$, the optimal single Vdd/single Vth solution varied between Vdd=1.2V/Vth=0.23V, Vdd=1.2V/Vth=0.14V, and Vdd=0.8V/Vth=0.08V assuming 0ps LCFF delay overhead. With single Vdd of 1.2V, the best dual Vth power saving was only 11.4%. However with no LCFF delay penalty at the outputs, dual Vth of 0.14V and 0.08V achieved

average power savings of 12.9% with Vdd of 0.8V, or 16.0% if we assume that the input drivers are also scaled down to 0.8V.

The optimal value of Vth depends greatly on Vdd. From the delay data in Appendix B.1, Vth of 0.08V provides a 22% speed increase at Vdd=1.2V versus Vth of 0.23V, but provides a 50% speed increase at Vdd=0.8V. In addition at Vdd=0.8V, the leakage is only about half the leakage at Vdd=1.2V. So the absolute increase in leakage power is less as Vth is reduced at Vdd=0.8V, reducing the penalty for using low Vth to reduce delay. Thus using a single Vth of 0.08V is acceptable at Vdd of 0.8V in the $1.2 \times T_{min}$ scenario, but a poor choice with Vdd of 1.2V.

The gate sizing/dual Vth assignment research by Wei, Roy and Koh in [233] is similar to our sizing/dual Vth research. They used a TILOS-like optimizer and did gate-level Vth assignment. We also performed Vth assignment at the gate-level, as the different Vth library characterizations had the same NMOS and PMOS Vth values for all transistors. Their case where switching activity was 0.1 is similar to our case studies where dynamic power dominates. Wei, Roy and Koh reported average power savings of 14% with dual Vth/gate sizing versus gate sizing using only low Vth [233]. Our result of 16.0% average power savings at $1.2 \times T_{min}$ with Vdd=0.8V/Vth=0.14V&0.08V versus Vdd=0.8V/Vth=0.08V with 0ps LCFF delay overhead is similar. In our other comparisons, dual Vth gave less power savings because dual Vth is less useful at higher Vdd and because gate size is the variable with greatest power/delay sensitivity at a tight delay constraint.

At $1.2 \times T_{min}$ assuming 80ps LCFF delay overhead, using dual Vth with dual Vdd the maximum power saving was 6.9% versus dual Vdd/single Vth and 15.0% versus single Vdd/dual Vth, and the average power savings were 4.6% and 3.4% respectively.

There is no significant advantage for using VDDL of 0.6V versus 0.8V. The greatest saving for VDDL of 0.6V versus 0.8V was 3.9% with single Vth and 4.2% with dual Vth, and the average

saving is only 0.3%. We have not accounted for any additional LCFF delay for conversion from 0.6V to 1.2V compared to converting from 0.8V to 1.2V, which were assumed to be 50ps and 70ps respectively in [124]. In correcting the $V_{dd}=0.6V$ delays, the $\alpha=1.66$ delay scaling with Equation (5.9) may still have been optimistic by 13% at $V_{th}=0.23V$ to 6% at $V_{th}=0.08V$ compared to the delay fit in Equation (5.10) that fit the $V_{dd}=0.5V$ data as well. Lower V_{dd} cells also have other problems such as smaller voltage noise margins. These weak 0.6V VDDL results argue against the conclusion in [124] and other papers that VDDL should be 50% of VDDH. The rule of thumb to use a value of about 70% of VDDH for VDDL, that is 0.8V, provides sufficiently good results. The incorrect conclusion in [124] to use VDDL=50% of VDDH was due to the incorrect delay characterization at $V_{dd}=0.6V$. It would be interesting to try multi- V_{dd} optimization with VDDH of 0.8V and VDDL of 0.6V, 0.6V being about 70% of 0.8V, but we don't have level converters designed and characterized to perform voltage level restoration from 0.6V to 0.8V.

The level converter delay and power overheads and larger rise delay when $V_{in} > V_{dd}$ pose a significant barrier to optimization. Running optimization with the level converter power set to zero then running further iterations with the correct power did help ECVS achieve up to 30% lower power for the $\alpha=1.3$ delay scaling of $V_{dd}=0.6V$ delays in Appendix I.2. While our multi- V_{dd} optimization approach is the best we know of, Section 7.6.5.2 showed that the results could be suboptimal by up to 23.6% in comparison to the single $V_{dd}=0.8V$ results, which are within a subspace of the multi- V_{dd} solution space. Experiments with reducing the level converter delay overheads would require a better delay reduction approach to ensure the final netlist meets the delay constraints, as the intermediate netlist with zero level converter delays will violate the delay constraints. The linear programming approach is not as good as Design Compiler at delay minimization, so some combined approach with a TILOS-like optimizer would be helpful. The delay reduction phase of the linear programming approach performs better with multi- V_{th} , as

changing a gate from high V_{th} to low V_{th} to speed it up only slightly increases the load on fanin gates compared to upsizing a gate. In theory, the University of Michigan CVS and ECVS approaches in [125][188] that aggressively try to set as many gates as possible to VDDL may be able to find the single V_{dd}=0.8V solution, but how good their results are at this point remains to be seen.

A fixed netlist topology was assumed in this chapter. If duplication and remapping of logic are used to provide additional slack greater power savings may be possible, primarily by further voltage scaling.

Chapter 8. Conclusions

To date the extent of the power gap between automated and custom design methodologies and the factors contributing to it have been unclear. While automated design flows are often blamed for poor speed and poor energy efficiency, other factors such as process technology have a significant impact on circuit speed and power consumption.

Our aim was to determine the extent of the power gap, then to identify and quantify the factors contributing to it. We then sought to identify low power custom design techniques that could be incorporated in an EDA flow to reduce the power gap between ASIC and custom designs. Finally, we examined in more detail some of the most promising of these methods for reducing power consumption.

8.1 Analysis of the power gap between ASIC and custom

We investigated the differences in power between ASICs and custom integrated circuits, with examples from 0.6um to 0.13um CMOS. We compared synthesizable and custom ARM processors, and also examined discrete cosine transform cores, as an example of dedicated low power functional units. In these cases, ASICs dissipated 3 to 7 \times more power than circuits designed with a custom methodology.

We developed a power and delay model to incorporate the major factors that contribute to the power gap between ASIC and custom. It includes pipelining, logic delay, voltage scaling and gate sizing. The logic delay is determined by factors such as the logic style, wire lengths after layout, process technology, and process variation which affects the worse case delay.

8.1.1 Power and delay pipeline model results

The power gap is largest at a tight performance constraint. We explored this with the pipeline model incorporating gate sizing and voltage scaling in Chapter 3. The estimates of the power gap

below do not include the impact of custom using a high speed logic style, process variation, or other factors.

A typical ASIC with slow D-type flip-flops and unbalanced pipeline stage delays may have $5.1\times$ the power of a custom processor at a tight performance constraint for the typical ASIC, but the power gap reduces to $4.0\times$ at only 7% lower performance.

If the ASIC pipeline delay overhead is reduced from 20 to 5 FO4 delays using latches or faster pulsed flip-flops with useful clock skew, the power gap is at most $1.9\times$ at the maximum performance for the excellent ASIC. In comparison, a custom design may have pipeline stage delay overhead of only 3 FO4 delays.

It is important to consider high level circuit techniques to provide timing slack along with low level circuit techniques such as voltage scaling and gate sizing that can reduce power if there is timing slack. These low level circuit techniques may provide 80% power savings, but the pipeline depth may have to be tripled to provide the required timing slack.

8.1.2 Summary of methods to close the power gap

From our analysis, the most significant opportunity for power reduction in ASICs is using microarchitectural techniques to maintain performance while reducing power by voltage scaling. In particular, reducing the pipeline delay overhead and using pipelining to increase timing slack can enable substantial power savings by reducing the supply voltage and downsizing gates. Multiple threshold voltages may be used to limit leakage while enabling a lower Vdd to be used. Choosing a low power process technology and limiting the impact of process variation reduces power by a large factor.

At a tight performance constraint for a typical ASIC design, we believe that the power gap can be closed to within $2.6\times$ by using these low power techniques with fine granularity standard cell

libraries, EDA tools targeting low power and careful RTL design. The remaining gap is mostly from custom designs having lower pipelining overhead and using high speed logic on critical paths. High speed logic styles are less robust and require careful layout, and thus are not amenable to use in an ASIC EDA methodology.

Having identified voltage scaling as the most significant approach for power reduction, we sought to put voltage scaling in an optimization framework.

8.2 Optimization of supply voltage, threshold voltage, and gate sizes

Gate sizing must be considered with optimization of Vdd and Vth, as the power consumption can be more sensitive to gate downsizing at a tight delay constraint. *Gate downsizing* reduces the gate internal capacitances and gate input pin capacitances, thus reducing switching power. Reducing the gate size also gives an approximately linear reduction in leakage and short circuit power, due to the higher transistor resistances. *Reducing the supply voltage* provides a quadratic reduction in switching power, and also provides substantial reductions in short circuit power and leakage power. *Increasing the threshold voltage* exponentially reduces the leakage power and also reduces the short circuit power. Both an *increase in Vdd* and a *decrease in Vth* increase the gate input pin capacitance, which in turn increases the delay and the switching power.

Except for gate downsizing in some situations, these power minimization approaches come with a delay penalty. If the difference between supply voltage and threshold voltage ($Vdd - Vth$) scales with Vdd then delay is inversely proportional to Vdd. However, Vth scaling is limited by the exponential increase in leakage power as Vth is reduced. Thus the delay may increase substantially when Vdd is reduced. To avoid the delay penalty for low Vdd and high Vth, we can use high Vdd and low Vth on critical paths, and use low Vdd and high Vth elsewhere to reduce the power.

The integrated circuit design industry has been very interested in how much power may be saved using multiple supply voltages and using multiple threshold voltages. An additional Vth level increases the fabrication cost by 3% and a set of masks costs on the order of a million dollars today [165], and multi-Vdd has additional area overheads for extra power rails and spatial isolation between wells of PMOS transistors biased at different voltages. Thus it is important to determine how much benefit these methods offer over scaling a single Vdd, single NMOS Vth and PMOS Vth.

8.2.1 Circuit power and delay model accuracy

One of the most important issues is how to pose the problem to avoid overly complicating the optimization with additional details, without degrading the final solution due to inaccurate models. Much academic circuit optimization research avoids considering separate timing arcs through gates and signal slew, as they add additional constraints and can increase the computational complexity.

Our results demonstrated the importance of having accurate delay and power analysis within the optimization formulation. In the linear program formulation, we included the first order delay impact of slew on the transitive fanout. When slew analysis was excluded from the linear program, the resulting netlists violated delay constraints or were on average 6.2% higher power. This indicates how important it is to account for slew both in static timing analysis to verify the results and in the actual optimization formulation. However, it was not necessary to include the impact of slew on the short circuit power of fanout gates, as typically 99% or more of the total power impact is accounted for at the gate that is changing, though in some cases it may be as low as 95%. Separate timing arcs must be considered as rise and fall delays differ, and they do not scale in the same manner as gate sizes, Vdd and Vth are changed.

Optimization researchers often exclude wire loads and short circuit power to simplify analysis; however, we found that wire loads can contribute 24% of the total power, and short circuit power can account for 27% of the total power at high Vdd and low Vth. The wire loads also increase the critical path delay, by 25% on average with the Vdd=1.2V/Vth=0.23V library. Typically, circuit designers say that short circuit power contributes around 10% of the total power, but we did not perform power minimization on these circuits. The short circuit power may be larger than typical, but as we use these netlists as a starting point for power minimization with the linear programming approach, it is essential that we model short circuit power as its contribution can be significant.

Another common error made by some researchers is to assume that delay scales in the same manner as the saturation drain current of a transistor using the Sakurai-Newton alpha-power law [173] with a velocity saturation index α of 1.3. A value for α of about 1.3 is typical to fit the saturation drain current for today's technologies, but can underestimate delay by as much as 50% when Vdd and Vth are scaled down, as other issues such as input slew and drain current in the linear region must be accounted for – see Section 5.2.1 for details. It is best to use delay and power data characterized for the given Vdd and Vth values, rather than extrapolating simple models.

Having determined what must be modeled in the optimization problem, we turn to methods of solving the problem.

8.2.2 Geometric program optimization of Vdd, Vth, and gate sizes

Optimization of gate sizes, Vdd and Vth to minimize power is an NP-complete problem. Other researchers have adopted heuristic approaches to make the problem computationally tractable, but heuristic solutions can be suboptimal and there is no knowledge of how far they are from the global minimum.

Posynomial models can be used to approximate a gate's delay and power consumption versus Vdd, Vth and gate size, giving a convex geometric programming problem in terms of continuous variables, for which the global minimum is polynomial time solvable. The posynomial fits for 0.13um logic gates characterized by SPICE had relative root mean square accuracy of 10% or better, which is comparable to what has been achieved by other authors for transistor sizing alone [117][207]. Unfortunately, the underlying data is non-convex when transistor folding and other issues are accounted for. Thus the accuracy of convex fits is fundamentally limited by the real data. 10% static timing and power analysis inaccuracy is not acceptable for a circuit designer, but it is sufficient to explore the optimization space.

The geometric programming formulation cannot handle power and delay overheads for insertion of voltage level converters, because of the stepwise discontinuity introduced. Thus this approach was limited to a clustered voltage scaling (CVS) multi-Vdd methodology where level restoration occurs only at the combinational outputs with level converter flip-flops, for which zero delay and zero power overhead is assumed.

The solution to the geometric program has a range of values for Vdd, Vth, and gate sizes. Vdd and Vth must be discretized, as only one or two values are acceptable because of the associated design and processing costs. We proposed a discretization heuristic that found dual Vdd, dual NMOS Vth, dual PMOS Vth solutions within 5.4% of the continuous voltage lower bound.

Significant power savings may be missed if Vdd and Vth are not optimized for a design. Only a couple of supply and threshold voltage values may be available in standard cell libraries provided by a foundry. Using dual Vdd and dual Vth with fixed voltages, the power was up to 40% higher than optimally scaling a single Vdd, single NMOS Vth, and single PMOS Vth. This shows how important it is to scale voltages optimally for a design with a given performance constraint. Substantial power savings may be achieved if foundries provide a larger number of supply and

threshold voltages to choose from to find something near the optimal. Dual supply and dual threshold voltages provided up to 18% power savings versus optimally choosing a single Vdd, single NMOS Vth and single PMOS Vth. As ASIC designers are limited to Vth values specified by the foundry, there may be more scope for power savings in ASICs when Vth is suboptimal.

The geometric programming computational runtime grows cubically with the circuit size. ASIC designers are typically interested in circuits with tens of thousands of gates or more, where there is no commonly repeated circuit structure that would allow the number of optimization variables to be substantially reduced. Thus faster heuristic methods must be used.

8.2.3 Linear program assignment for Vdd, Vth, and gate sizes

We developed a better approach for gate sizing based on linear programming and then extended it to multi-Vdd and multi-Vth assignment. The basic approach to gate sizing in commercial EDA software has changed little in the past 20 years. These gate sizers like TILOS [64] proceed in a greedy manner, picking the gate with the best power or area versus delay tradeoff to change, and iterating. The TILOS-like greedy approaches are relatively fast, being $O(n^2)$, and other approaches that perform better with similar static timing analysis (STA) accuracy have had worse computational complexity.

We proposed a linear programming based gate sizing approach that takes a global view of the circuit, instead of the commonly used greedy “peephole” optimization approaches. The heuristic approach proceeds iteratively by identifying the best alternate cell choices to reduce the power or reduce the delay of each gate, and encoding these in a linear program, which is then solved to determine which cells to change. Iterating cycles of reducing power then reducing delay to meet the delay constraint provided more power savings than stopping power minimization when the delay constraint is reached: further cycles of delay reduction then power reduction get out of this local minimum. This approach achieved up to 32.3% power savings and on average 16.3% power

savings versus gate sizing in Design Compiler, a commonly used commercial synthesis tool. Our optimization approach has between $O(n)$ and $O(n^2)$ runtime growth, making it scalable to large circuit sizes.

Performing gate-level optimization with multiple threshold voltages is no different to performing gate sizing, providing leakage is accounted for. On the other hand, multi-Vdd complicates optimization because voltage level converters must be inserted to restore the voltage swing of low Vdd gates that drive high Vdd gates, to avoid substantial static current. The power and delay overheads for level converters create a very “hilly” non-convex optimization space where it is easy to get stuck in a local minimum.

The dual-Vdd/dual-Vth/sizing results with the linear programming approach averaged 5% to 13% lower power than the best alternate approaches that we know of for this problem [124][125][188].

We compared multi-Vdd/multi-Vth/sizing results to the best single-Vdd/single-Vth/sizing results at a given delay constraint. The gate sizing results provided a strong baseline to compare results against. The largest power savings for multi-Vth were at a delay constraint of 1.2 times the minimum delay with the nominal Vdd of 1.2V and Vth of 0.23V. At this point, it was optimal to use a low Vth of 0.08V to allow Vdd to be reduced from 1.2V to 0.8V, reducing the total power on average by 11% versus using the nominal voltages. Leakage was about 40% of the total power with Vth of 0.08V, so using a second higher Vth of 0.14V in portions of the circuit where there was timing slack helped reduce the leakage. Dual Vth provided 16% power savings on average and up to 26% saving versus only using Vth of 0.08V. The net power saving was on average 25% versus using Vdd of 1.2V and Vth of 0.23V. Triple Vth provided at most 5.1% power savings versus using dual Vth, which is not sufficient to justify use of a third threshold voltage.

We found that the power savings with gate-level multi-Vdd were generally less than 10%. This is insufficient to justify the layout area and wiring overheads that were not accounted for in our

analysis. Using multi-Vdd is more appropriate at a module level, making a good choice of a single supply voltage for each module based on the delay of its critical paths.

Other researchers have reported large power savings with multi-Vdd compared to using a single Vdd – for example 30% to 40% average power savings [124][188]. Our multi-Vdd power savings were smaller because the results were compared against a stronger gate sizing baseline with a wider range of choices for a single Vdd and a single Vth. In addition, the gate delays at low Vdd of 0.6V in our analysis were not determined by optimistic scaling, but rather by more accurate interpolation of characterized data with Vdd of 1.2V, 0.8V and 0.5V.

8.2.3.1 Future research

Our linear programming approach is not as good as Design Compiler at delay minimization. A combined approach with a TILOS-like optimizer for delay reduction would enable increased power savings at a tight delay constraint.

In spite of achieving better multi-Vdd/multi-Vth results than other known approaches, the LP multi-Vdd results were suboptimal by up to 23.6% versus using a single low supply voltage. More power savings may be available with multi-Vdd for an optimization approach that can get around the level converter delay overheads that pose a significant barrier to finding the global minimum.

8.3 Summary

We have examined circuit design and synthesis as a whole, with energy efficiency as a design driver. ASICs dissipated 3 to 7 \times more power than circuits designed with a custom methodology for the same type of application. The power gap between ASIC and custom designs is largest at a tight performance constraint and depends on the application. Custom design techniques can provide larger benefits on small circuit elements, such as a Boolean function unit [236], but the overall benefit is less when considering a complete integrated circuit.

We compared ASIC and custom design approaches to identify techniques that could be automated to reduce power consumption in ASICs. High speed and low power techniques can be combined to minimize power in ASICs while meeting performance requirements. The most significant opportunity for power reduction in ASICs is using microarchitectural techniques to maintain performance while reducing power by voltage scaling. Using our pipeline power and delay model characterized for representative circuitry for the particular application, designers can make these high-level and low-level power and delay trade-offs. Our improved approach for gate sizing, multi-V_{th} and multi-V_{dd} assignment gives substantial power savings without degrading performance.

ASICs may be unable to meet the performance requirements for some high speed applications. However, as technology continues to scale down, ASICs can achieve higher speeds and lower power consumption. We hope to encourage EDA tool developers to enable this path for power reduction in ASICs and faster design time for custom chips. There remains significant room for improvement in tools for technology mapping, gate sizing, and layout. Foundries can help by providing richer standard cell libraries with finer granularity of sizes and skewed drive strengths, characterized over a range of supply voltages.

With improved EDA tool and standard cell library support for power reduction, using the low power techniques summarized in Section 8.1.2, careful ASIC designers will be able to reduce the power gap versus custom to 2.6×, or less at relaxed performance constraints.

Appendix A Pipeline model details

A.1 Characteristics of ASIC and custom processors

Section 3.2 presented a few representative custom and ASIC processors across process technologies from 0.35um to 65nm. A wider selection is presented here. For all the data presented, the operating conditions are worst case. Process conditions are mostly typical, but worst case for the synthesizable ARM processors.

To estimate the clock period in FO4 delays from the effective channel length L_{eff} , we use the equations from Section 2.2,

$$500 \times L_{eff} \text{ ps for worst case operating and typical process conditions} \quad (\text{A.1})$$

$$600 \times L_{eff} \text{ ps for worst case operating and worst case process conditions} \quad (\text{A.2})$$

The clock period and unpipelined combinational logic delay assuming 30% pipeline stage delay overhead for ASICs and super-pipelined custom processors is shown in Table A.1. For custom processors with shorter pipelines, a 20% pipeline stage delay overhead is assumed in Table A.2.

There are several caveats with this data:

- FO4 delay estimates are not that accurate without specifically characterizing the FO4 delay for a given process. For example, Gronowski et al. state that the “typical gate delay” per cycle were 16 for the 21064, 14 for the 21164 and 12 for the 21264 [80]. Harris and Horowitz estimate that a typical gate delay is 1.24 FO4 delays [87], so the corresponding number of FO4 delays would be 19.8 for the 21064, 17.4 for the 21164 and 14.9 for the 21264. Our from L_{eff} is 5% high for the 21164 and 10% low for the 21264.

- L_{eff} for Intel's 0.25um process that the Katmai Pentium III was fabricated in was estimated from the 18% speed increase from their P856 0.25um process with L_{eff} of 0.18um [20] to the P856.5 0.25um process [23].
- Intel has not published the number of pipeline stages for the Banias and Dothan Pentium M processors, nor for the Yonah Core Duo (not listed in the tables). Speculations have ranged between 11 and 13 pipeline stages. The next version, the Conroe Core 2 Duo, has 14 integer pipeline stages [97]. The X6900 2 Extreme version of the Conroe is expected to be released with a clock frequency of 3.2GHz in the fourth quarter of 2006. Based on similar clock periods to the Conroe, we have estimated 14 integer pipeline stages for Banias and Dothan here. If Dothan has only 13 stages, we predict that Conroe may scale to a clock period of 17 FO4 delays or 3.4 GHz in 65nm. If Dothan has only 12 stages; Conroe may scale to a clock period of 16 FO4 delays or 3.6 GHz in 65nm – time will tell.
- Samsung has not published the L_{eff} for their custom ARM 1020E implementation, the Halla. The saturation drain current for the process is the same as that for TSMC's 0.13um process with L_{eff} of 80nm, but Samsung's junction capacitances are about 25% less [120]. The saturation drain current is less than that of Intel's 0.13um process with L_{eff} of 70nm [219] and 60nm [212]. Thus L_{eff} of 80nm has been assumed for Samsung's process.
- Power constraints limit scaling of the 31 stage Pentium 4 processors to higher clock frequencies [238].
- Worst case process and operating conditions were assumed for ARM's Cortex-A8 data, as they did not state what conditions it was characterized for.

As there are many references for this data, we summarize the sources for process technology here:

- AMD: 250nm [73], 180nm [55], 130nm [163], 90nm [78]
- Intel: 250nm P865 [20] and P865.5 [23], 180nm [71], 130nm with L_{eff} of 70nm [219] and with L_{eff} of 60nm [212], 90nm [211], 65nm [220]
- STMicroelectronics 180nm [192]
- TSMC: 180nm [216], 130nm [215], 90nm [217], 65nm [241]

The references for the processor data were:

- DEC Alpha: 21064 [80], 21164 [80], 21264 [80]
- AMD: Athlon K7 [73], Opteron [115], various [84][85][148][206]
- ARM: 926 EJ-S [7], 1026EJ-S [8], 1136J-S [9], Cortex-A8 [10], Cortex-R4 [11], StrongARM [144], Halla [136], XScale [45]
- Lexra LX4380 [137]
- IBM Power PC [180]
- Intel: Pentium M [104], Core 2 [105], Pentium 4 (Willamette) [94], various [84][85][149][206]
- STMicroelectronics iCORE [170]
- Tensilica Xtensa: T1020 [209], T1050 [210], Xtensa 6 [208]

Table A.1 Characteristics of ASICs and super-pipelined Pentium 4 processors assuming 30% pipelining delay overhead. The FO4 delay for typical (“typ”) process conditions was calculated with Equation (A.1), and Equation (A.2) was used for worst case (“wc”) process conditions.

	Frequency (MHz)	Technology (nm)	Effective Channel Length (mm)	Voltage (V)	Integer Pipeline Stages	FO4 delays/stage	20% Pipelining Overhead (FO4 delays)	Unpipelined Clock Period (FO4 delays)	Pipelining Overhead % of Unpipelined Clock Period	Clock Frequency Increase by Pipelining	Process Conditions	Operating Conditions
Custom PCs												
Pentium 4 (Willamette)	2000	180	100	1.75	20	10.0	3.0	143	2.1%	×14.3	typ	wc
Pentium 4 (Northwood)	3200	130	60	1.50	20	10.4	3.1	149	2.1%	×14.3	typ	wc
Pentium 4 (Gallatin)	3466	130	60	1.60	31	9.6	2.9	212	1.4%	×22.0	typ	wc
Pentium 4 (Prescott)	3800	90	50	1.40	31	10.5	3.2	232	1.4%	×22.0	typ	wc
ASICs												
Xtensa T1020 (Base)	175	250	180	2.50	5	63.5	19.0	241	7.9%	×3.8	typ	wc
Xtensa T1020 (Base)	250	180	130	1.80	5	61.5	18.5	234	7.9%	×3.8	typ	wc
Xtensa T1050 (Base)	350	130	80	1.50	5	71.4	21.4	271	7.9%	×3.8	typ	wc
Xtensa 6 (Base)	600	90	50	1.20	5	66.7	20.0	253	7.9%	×3.8	typ	wc
Lexra LX4380	266	180	130	1.80	7	57.8	17.4	301	5.8%	×5.2	typ	wc
Lexra LX4380	420	130	80	1.20	7	59.5	17.9	310	5.8%	×5.2	typ	wc
iCORE	520	180	150	1.80	8	25.6	7.7	151	5.1%	×5.9	typ	wc
ARM 926EJ-S	200	180	130	1.80	5	64.1	19.2	244	7.9%	×5.9	wc	wc
ARM 926EJ-S	276	130	80	1.20	5	75.5	22.6	287	7.9%	×3.8	wc	wc
ARM 926EJ-S	500	90	50	1.00	5	66.7	20.0	253	7.9%	×3.8	wc	wc
ARM 1026EJ-S	325	130	80	1.20	6	64.1	19.2	288	6.7%	×4.5	wc	wc
ARM 1026EJ-S	540	90	50	1.00	6	61.7	18.5	278	6.7%	×4.5	wc	wc
ARM 1136J-S	400	130	80	1.20	8	52.1	15.6	307	5.1%	×5.9	wc	wc
ARM 1136J-S	620	90	50	1.00	8	53.8	16.1	317	5.1%	×5.9	wc	wc
ARM Cortex-R4	300	130	80	1.20	8	69.4	20.8	410	5.1%	×5.9	wc	wc
ARM Cortex-R4	500	90	50	1.00	8	66.7	20.0	393	5.1%	×5.9	wc	wc
ARM Cortex-A8	800	65	40	1.20	13	52.1	15.6	490	3.2%	×9.4	wc	wc

Table A.2 Custom design characteristics assuming 20% timing overhead. The FO4 delay for typical (“typ”) process conditions was calculated with Equation (A.1).

Custom Processors	Frequency (MHz)	Technology (nm)	Effective Channel Length (mm)	Voltage (V)	Integer Pipeline Stages	FO4 delays/stage	20% Pipelining Overhead (FO4 delays)	Unpipelined Clock Period (FO4 delays)	Pipelining Overhead % of Unpipelined Clock Period	Clock Frequency Increase by Pipelining	Process Conditions	Operating Conditions
Alpha 21064	200	750	500	3.30	7	20.0	4.0	116	3.4%	×5.8	typ	wc
Alpha 21164	300	500	365	3.30	7	18.3	3.7	106	3.4%	×5.8	typ	wc
Alpha 21264	600	350	250	2.20	7	13.3	2.7	77	3.4%	×5.8	typ	wc
IBM Power PC	1000	250	150	1.80	4	13.3	2.7	45	5.9%	×3.4	typ	wc
Custom PCs												
Athlon (K7)	700	250	160	1.60	10	17.9	3.6	146	2.4%	×8.2	typ	wc
Athlon XP (Palomino)	1733	180	100	1.75	10	11.5	2.3	95	2.4%	×8.2	typ	wc
Athlon 64 (Clawhammer)	2600	130	80	1.50	12	9.6	1.9	94	2.0%	×9.8	typ	wc
Athlon 64 (San Diego)	2800	90	50	1.40	12	14.3	2.9	140	2.0%	×9.8	typ	wc
Pentium III (Katmai)	600	250	150	2.05	10	22.2	4.4	182	2.4%	×8.2	typ	wc
Pentium III (Coppermine)	1130	180	100	1.75	10	17.7	3.5	145	2.4%	×8.2	typ	wc
Pentium M (Banias)	1700	130	60	1.48	14	19.6	3.9	224	1.8%	×11.4	typ	wc
Pentium M (Dothan)	2260	90	50	1.37	14	17.7	3.5	202	1.8%	×11.4	typ	wc
Core 2 Extreme (Conroe)	2930	65	35	1.34	14	19.5	3.9	222	1.8%	×11.4	typ	wc
Custom ARM												
StrongARM	215	350	250	2.00	5	37.2	7.4	156	4.8%	×4.2	typ	wc
XScale	800	180	135	1.80	7	18.5	3.7	107	3.4%	×5.8	typ	wc
Halla (ARM 1020E)	1200	130	80	1.10	6	20.8	4.2	104	4.0%	×5.0	typ	wc

A.2 Modeling register and clock tree power in different processors

We use this model for the total power consumption including combinational logic, the registers and the clock tree:

$$P_{total} = (P_{dynamic} + P_{leakage})(1 + \beta n^\eta) \quad (\text{A.3})$$

where η is the latch growth factor and β is a coefficient that helps determine the percentage of power consumed by the registers and clock tree. Assuming a given value for η , we can calculate the value of β in Equation (A.3) from the number of pipeline stages, and from the percentage of register and clock tree in a processor that was tabulated for different processors in Table 3.6.

Consider two values for η : 1.1, corresponding nominally to an integer pipeline, and 1.7, corresponding to pipelining a floating point unit. The Alpha 21264 and Itanium have both integer and floating point units, whereas the other processors do not have a floating point unit. For η of 1.1, β ranges from 0.026 to 0.154. β has values of 0.014 and 0.017 for the two processors with floating point units if we instead assume η of 1.7. These β values are shown in Table A.3.

In Chapter 3, we assume a value for β of 0.05, as is typical for most of the processors of five to eight pipeline stages.

Table A.3 Register and clock tree power consumption in various processors and corresponding β values assuming a given value for η in Equation (A.3) [6][45][75][76][139][144]. We do not include values for the floating point pipeline for those processors that did not have a floating point unit.

Processor	# of Pipeline Stages	Registers and Clock Tree Power as % of Total Power	Integer Pipeline		Floating Point Pipeline	
			η	β	η	β
16-bit CompactRISC	3	34%	1.1	0.154	n/a	n/a
MCORE	4	36%	1.1	0.122	n/a	n/a
StrongARM	5	25%	1.1	0.057	n/a	n/a
XScale (DSP FIR filter)	7	18%	1.1	0.026	n/a	n/a
XScale (Dhrystone MIPS)	7	23%	1.1	0.035	n/a	n/a
Alpha 21264	7	32%	1.1	0.055	1.7	0.017
Itanium	8	33%	1.1	0.050	1.7	0.014

A.3 Correcting Harstein and Puzak's pipeline power model

As described in Section 3.3.3, the power model used by Harstein and Puzak has the form [89]

$$P_{\text{timing}} = \left(\frac{1}{T} \alpha_{\text{clock gating}} E_{\text{dynamic}} + P_{\text{leakage}} \right) \beta n^{\eta} \quad (\text{A.4})$$

which excludes the power for combinational logic. Harstein and Puzak use $1/(1+\gamma n)$ for $\alpha_{\text{clock gating}}$, the fraction of the time that the pipeline is not clock gated to avoid dynamic power for pipeline stalls.

From Equation (3.20) with no pipeline imbalance and no slack used for voltage scaling or gate sizing, the optimization problem to maximize BIPS^m/W is to find the optimal value of n in

$$\begin{aligned}
& \text{minimize} && P_{total} (T(1+\gamma n))^m \\
& \text{subject to} && T = \frac{t_{comb\ total}}{n} + t_{timing\ overhead} \\
& && n \geq 1 \\
& && P_{total} = \frac{1}{T} \left(\frac{1}{(1+\gamma n)} (1 - k_{leakage}) + k_{leakage} \right) \beta n^\eta
\end{aligned} \tag{A.5}$$

With the power for the combinational logic the optimization problem is to find the optimal value of n in

$$\begin{aligned}
& \text{minimize} && P_{total} (T(1+\gamma n))^m \\
& \text{subject to} && T = \frac{t_{comb\ total}}{n} + t_{timing\ overhead} \\
& && n \geq 1 \\
& && P_{total} = \frac{1}{T} \left(\frac{1}{(1+\gamma n)} (1 - k_{leakage}) + k_{leakage} \right) (1 + \beta n^\eta)
\end{aligned} \tag{A.6}$$

We can graph these metrics, $\text{BIPS}^m/\text{W} = 1/(P_{total}(T(1+\gamma n))^m)$, and see the impact of excluding the combinational power.

Harstein and Puzak assumed that the leakage is 15% of the total power without clock gating, $k_{leakage}$ is 0.15, and that the latch growth factor η was 1.1. As their theoretical model ignored combinational power, it is not clear what value should be used for β to model Harstein and Puzak's results. However, their power model was based on that of Srinivasan et al. [186] which assumed that latches contribute 70% of the total power for a seven stage pipeline, corresponding to a β value of 0.27.

Harstein and Puzak found that the optimal number of pipeline stages was 22 to minimize $T/\text{instruction}$, giving a clock period of 8.9 FO4 delays [90] with $t_{comb\ total}$ of 140 FO4 delays and $t_{timing\ overhead}$ of 2.5 FO4 delays. From Equation (3.19), the corresponding CPI penalty per pipeline stage γ is about 0.12. The parameters are summarized in Table A.4.

The resulting graphs of BIPS^m/W with these parameter values look very different with and without combinational power. Without combinational power in Figure A.1, a single pipeline stage is optimal to maximize BIPS^2/W , as they showed in Figure 5 in their paper [89].

When combinational power is included in Figure A.2, 3.6 pipeline stages is optimal to maximize BIPS^2/W , giving a clock period of 41.4 FO4 delays. 6.9 pipeline stages is optimal to maximize BIPS^3/W , giving a clock period of 22.8 FO4 delays – this corresponds almost exactly with the 7 pipeline stages and 22.5 FO4 delays per stage that Harstein and Puzak report from simulation [89].

We conclude that Harstein and Puzak's omission of combinational power in Equation (A.4) is erroneous, and that their pipeline power model with combinational power included closely fits their simulation results.

Table A.4 Parameters for Harstein and Puzak's power model [89].

Parameter	Represents	Value
k_{leakage}	fraction of total power due to leakage at T_{\min}	0.15
$t_{\text{comb total}}$ (FO4 delays)	total unpipelined combinational delay	140
$t_{\text{timing overhead}}$ (FO4 delays)	timing overhead per pipeline stage	2.5
β	coefficient for power due to registers and the clock tree	0.27
γ	increase in CPI with pipeline stages due to hazards	0.12
η	latch growth factor for increase in # of registers with pipeline depth	1.1

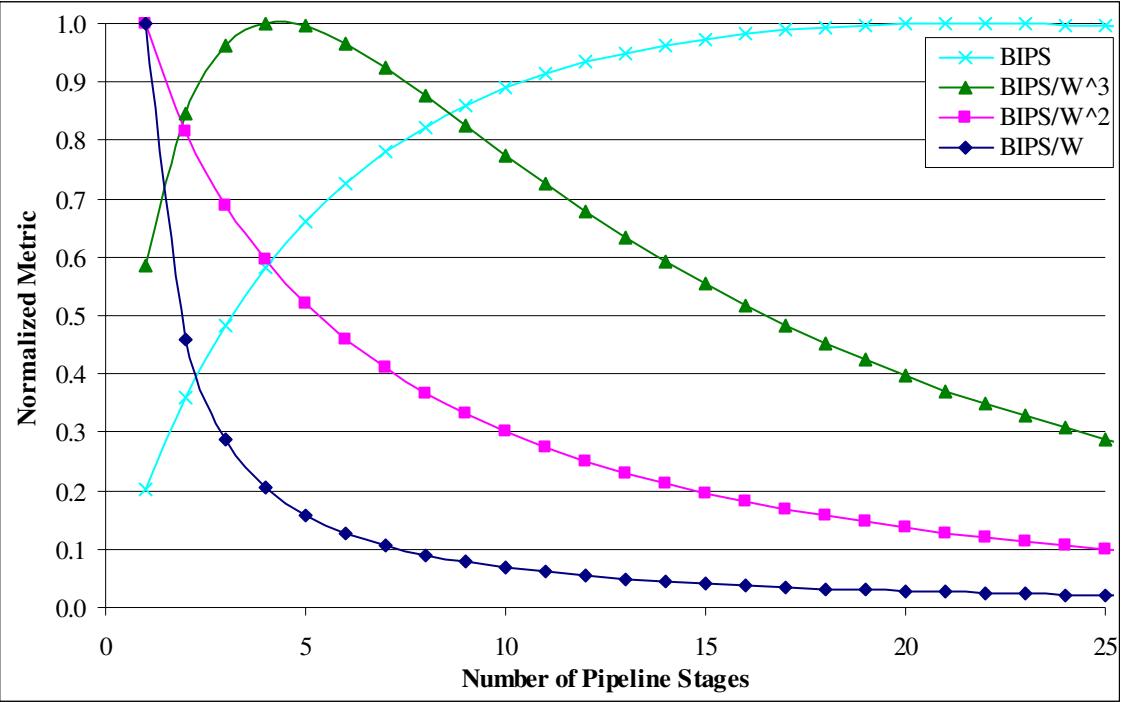


Figure A.1 Graph of metrics normalized to their maximum value versus the number of pipeline stages n from Equation (A.5) where combinational power *is not* included.

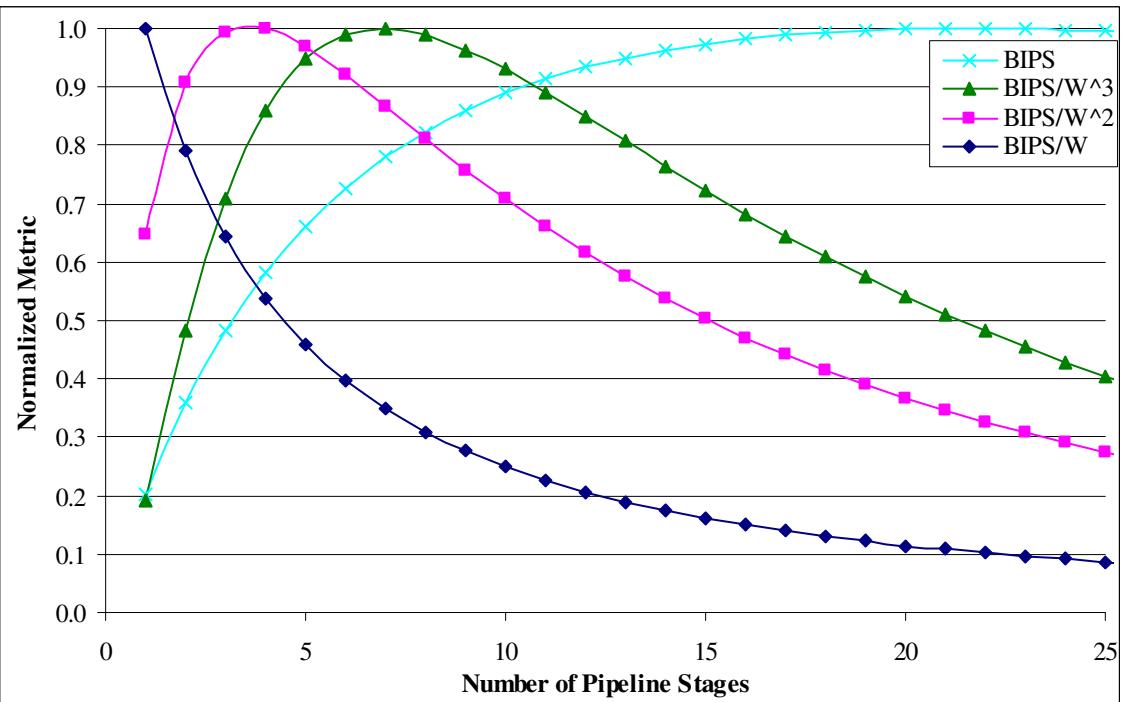


Figure A.2 Graph of metrics normalized to their maximum value versus the number of pipeline stages n from Equation (A.6) where combinational power *is* included.

A.4 Finding T/T_{\min} for minimum energy per operation

The minimum energy per operation is found when there is substantial timing slack to reduce the dynamic and leakage power, and the pipelining power overheads are minimized. Consequently, a single pipeline stage is optimal to minimize the energy per operation, as this minimizes the power overhead for registers. With only one pipeline stage and $m = 1$ to minimize the energy/operation, Equation (3.20) can be simplified to

$$\text{minimize } \tau(a'_{dynamic} + a'_{leakage}) + \frac{\tau b'_{dynamic}}{(\tau + c_{dynamic})^{d_{dynamic}}} + \frac{\tau b'_{leakage} e^{\lambda_{leakage}\tau}}{(\tau + c_{leakage})^{d_{leakage}}}$$

where

$$\begin{aligned} a'_{dynamic} &= (1 + \beta)(1 - k_{leakage})a_{dynamic} / P_{dynamic}(T_{\min}) \\ b'_{dynamic} &= (1 + \beta)(1 - k_{leakage})b_{dynamic} / P_{dynamic}(T_{\min}) \\ a'_{leakage} &= (1 + \beta)(1 + \gamma)k_{leakage}a_{leakage} / P_{leakage}(T_{\min}) \\ b'_{leakage} &= (1 + \beta)(1 + \gamma)k_{leakage}b_{leakage} / P_{leakage}(T_{\min}) \\ \tau &= T / T_{\min} \end{aligned} \tag{A.7}$$

The derivative with respect to τ , which is the ratio of T/T_{\min} , is given by

$$a'_{dynamic} + a'_{leakage} + \frac{b'_{dynamic}(\tau + c_{dynamic}) - \tau b'_{dynamic} d_{dynamic}}{(\tau + c_{dynamic})^{d_{dynamic}+1}} + \frac{b'_{leakage} e^{\lambda_{leakage}\tau} ((1 + \lambda_{leakage}\tau)(\tau + c_{leakage}) - \tau d_{leakage})}{(\tau + c_{leakage})^{d_{leakage}+1}} \tag{A.8}$$

The optimal value of τ to minimize the energy per operation makes this derivative zero. For our parameters, the derivative is given by

$$0.0160 + \frac{(-0.490\tau - 0.101)}{(\tau - 0.352)^{3.70}} + \frac{0.101e^{-0.971\tau}(-0.971\tau^2 + 1.79\tau - 0.997)}{(\tau - 0.997)^{1.182}} \tag{A.9}$$

and the solution for when the derivative is zero is $\tau = 4.46$, as shown with the graph of the derivative in Figure A.3.

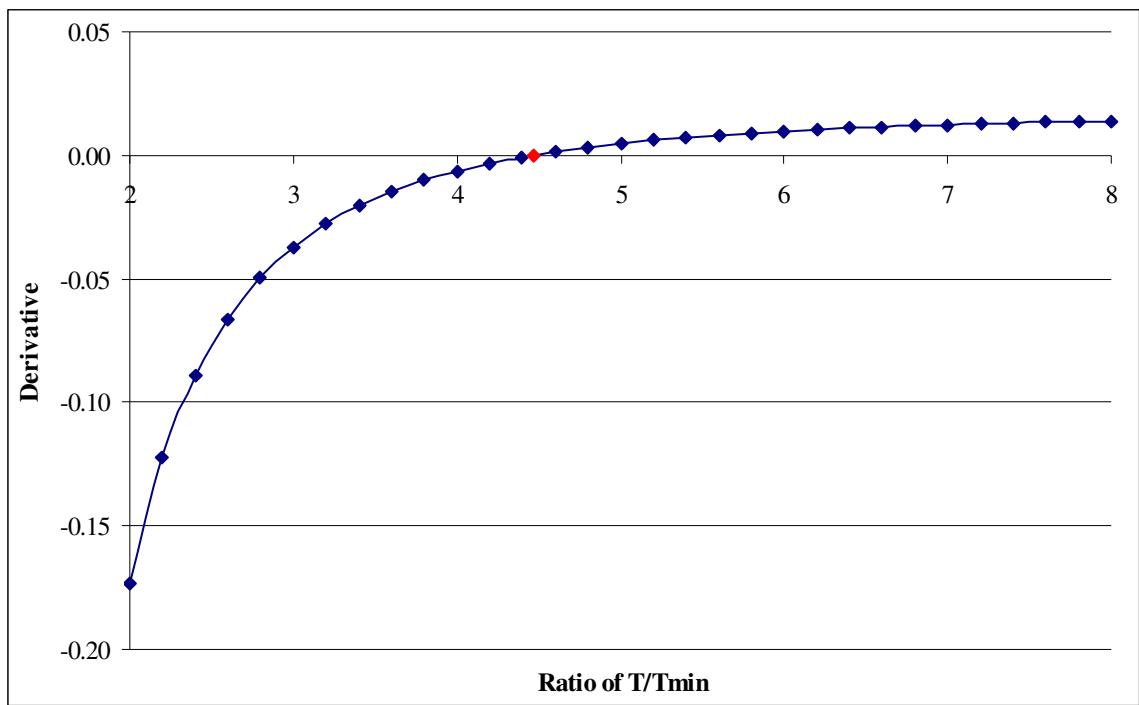


Figure A.3 Derivative of energy per operation with respect to the ratio of T/T_{\min} for a single stage pipeline.

A.5 Graphs of BIPS^m/W with voltage scaling and gate sizing

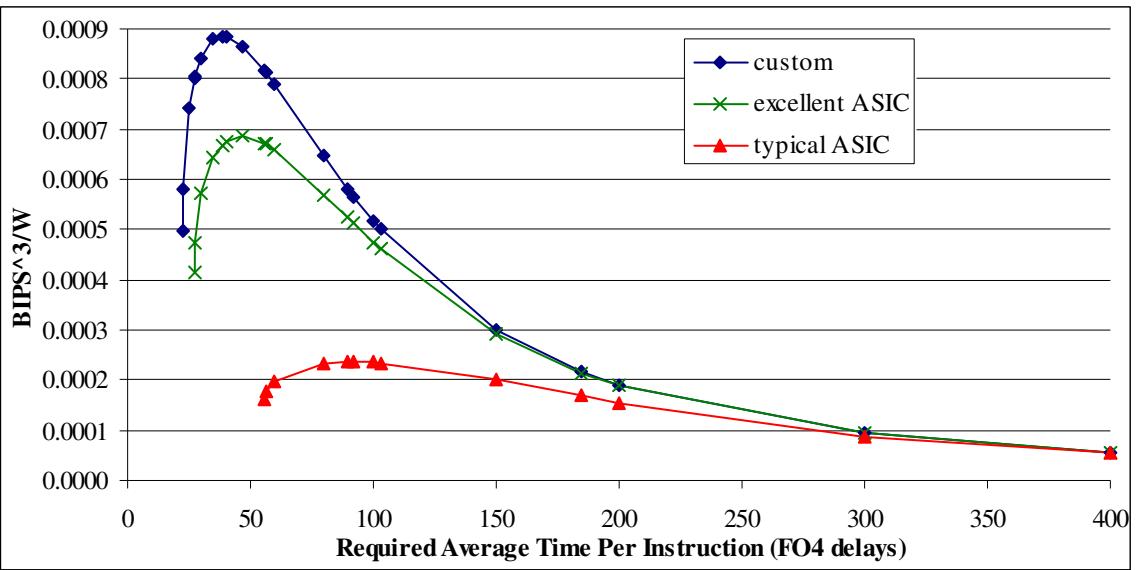


Figure A.4 This graph shows the BIPS^3/W at a given performance constraint.

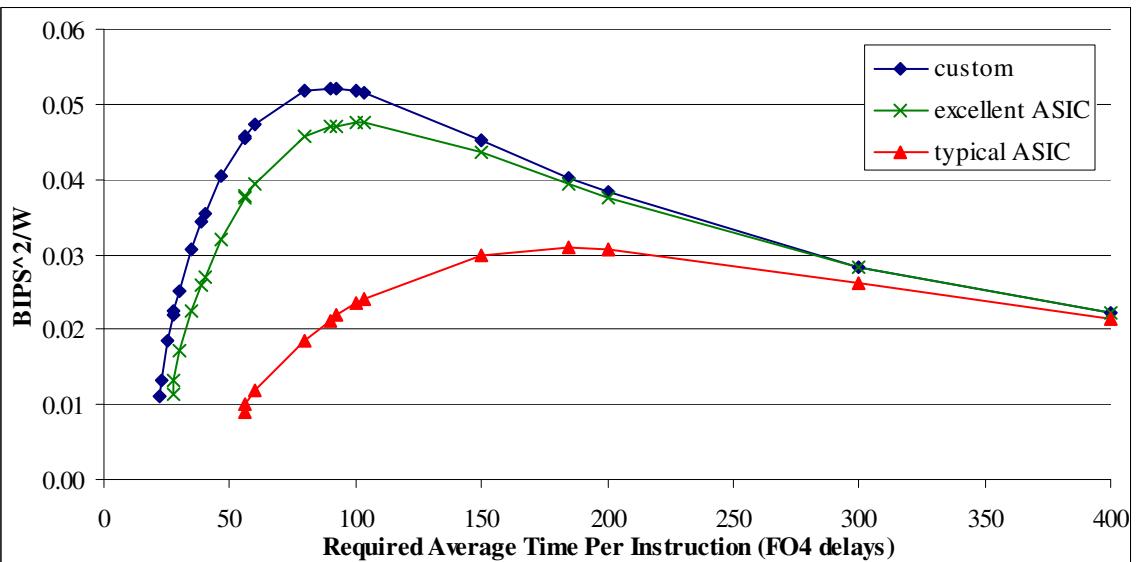


Figure A.5 This graph shows the BIPS^2/W at a given performance constraint.

A.6 Dynamic and leakage power fits for the pipeline model

These fits are used in analysis with the pipeline model for results discussed in Section 3.5.6.

A.6.1 Fits for voltage scaling with T_{\min} of 0.9

These dynamic and leakage power fits are for the power with optimal Vdd and optimal Vth to minimize total power versus delay in Section 5.2.6, with T_{\min} of 0.9 where optimal Vdd is 1.15V

and optimal V_{th} is 0.14V. The fits to normalized dynamic power and normalized leakage power were:

$$P_{dynamic}(T) = 0.00386 + \frac{0.193}{\left(\frac{T}{T_{min}} - 0.654\right)^{1.543}} \quad (1.10)$$

$$P_{leakage}(T) = 0.0258 + \frac{0.340e^{-0.355T/T_{min}}}{\left(\frac{T}{T_{min}} - 0.842\right)^{0.701}} \quad (1.11)$$

Relative root mean square error was 0.1% for the dynamic power fit and 0.2% for the leakage power fit.

A.6.2 Fits for gate sizing

Power versus delay data was provided for gate sizing from the minimum delay T_{min} to $1.5 \times T_{min}$ for a TILOS-like gate sizer by Sarvesh Kulkarni and Ashish Srivastava. This data is discussed in Appendix G.4. As the gate sizing power-delay curves were relatively shallow (see Figure G.2) compare to power-delay curves with voltage scaling, we used the fit to total power consumption for both dynamic and leakage power.

The steepest power-delay curve with gate sizing was for c1355. The fit for normalized power versus normalized delay for c1355 with relative root mean square error of 1.6% was

$$P(T) = \frac{0.251}{\left(\frac{T}{T_{min}} - 0.931\right)^{0.513}} \quad (1.12)$$

The least steep power-delay curve with gate sizing was for c7552. The fit for normalized power versus normalized delay for c7552 with relative root mean square error of 0.5% was

$$P(T) = \frac{0.563}{\left(\frac{T}{T_{\min}} - 0.808 \right)^{0.344}} \quad (1.13)$$

A.7 Power consumption due to glitching

Based on experimental data from a dynamic circuit timing simulator, Srinivasan et al. modeled glitching's contribution to the dynamic power of the pipeline's combinational logic as depending linearly on the logic depth [186]. A generated glitch was assumed to have a high probability of propagating through the combinational circuitry of a pipeline stage to the pipeline registers for that stage. The range of combinational logic delays that they consider for a pipeline stage is only 4 to 34 FO4 delays; whereas, we have up to 180 FO4 delays for an unpipelined design. While the glitching power may be fit reasonably well by a linear model over Srinivasan et al.'s $8.5\times$ range, it is not clear if that holds true for the $45\times$ range of combinational logic delays that we consider. Indeed, glitching power data for pipelined FPGA multipliers has sublinear growth with logic depth over a $32\times$ range¹ from an unpipelined 32-bit multiplier to one with 32 sets of pipeline stage registers strategically inserted in [171], and from a 64-bit multiplier with 2 to 64 pipeline stages in [237].

Registers can be intentionally positioned to reduce dynamic power by preventing glitches propagating from gates where there is a large amount of glitching to fanouts driving large capacitances [135][168]. We will not take this into account in the power model, assuming that pipeline registers which are positioned to balance pipeline stage delays provide sufficient reduction in glitching.

¹ Rollins and Wirthlin report data with 0, 1, 2, 4, 8, 16 and 32 sets of pipeline stage registers inserted [171], which corresponds respectively to 1, 2, 3, 5, 9, 17 and 33 pipeline stages in total.

A.7.1 Glitching model

We propose a simple model that attempts to estimate the average behavior of glitches propagating through combinational logic. We assume that on average each node has a probability g of generating a glitch from another fanin or noise, and a probability p of propagating a glitch. Then the average glitching activity G_m at the output of a gate at logic depth m is

$$\begin{aligned} G_m &= g + pg_{m-1} \\ &= \sum_{i=1}^m gp^{i-1} \\ &= \begin{cases} gm, & p = 1 \\ \frac{g(1-p^m)}{(1-p)}, & p \neq 1 \end{cases} \end{aligned} \quad (\text{A.14})$$

This is simply the sum of a geometric series, with the result shown. The total glitching activity $G_{total,m}$ from logic depths 1 to m is

$$\begin{aligned} G_{total,m} &= \sum_{i=1}^m G_m \\ &= \begin{cases} \frac{gm(m+1)}{2}, & p = 1 \\ \frac{gm}{(1-p)} - \frac{gp(1-p^m)}{(1-p)^2}, & p \neq 1 \end{cases} \end{aligned} \quad (\text{A.15})$$

This assumes that the average number of gates at a given logic level is the same. If the unpipelined combinational logic depth is $t_{comb\ total}$, then a balanced pipeline with n stages has combinational logic depth per pipeline stage of

$$t_{comb} = \frac{t_{comb\ total}}{n} \quad (\text{A.16})$$

The corresponding glitching activity for the entire pipeline is

$$nG_{total,t_{comb\ total}/n} = \begin{cases} \frac{gt_{comb\ total}}{2} \left(1 + \frac{t_{comb\ total}}{n} \right), & p = 1 \\ \frac{gt_{comb\ total}}{(1-p)} - \frac{ngp(1-p^{t_{comb\ total}/n})}{(1-p)^2}, & p \neq 1 \end{cases} \quad (\text{A.17})$$

The parameters g , p and $t_{comb\ total}$ can be fit to experimental data to provide a good fit. As switching activity for logical computation and from glitches are both proportional to clock frequency, the impact of pipelining on clock frequency and hence dynamic power has not been discussed here. That is, the ratio between dynamic power from glitching and non-spurious activity is not affected by the clock frequency.

A.7.2 Glitching portion of dynamic power fit for a FPGA 32-bit multiplier

Fitting Equation (A.17) to the 32-bit multiplier data in [171] by minimizing the relative least squares error gives for the glitching as a fraction of the dynamic power due to non-spurious transitions:

$$4.93 - 1.933n(1 - 0.922^{30.0/n}) \quad (\text{A.18})$$

The average error magnitude is 3.0% and the maximum error is 8.5%. This fit is reasonably accurate versus the 32-bit multiplier data, as shown in Figure A.6. From the fit, the glitch generation probability g is 0.0129, the glitch propagation probability is 0.922, and the unpipelined combinational logic depth $t_{comb\ total}$ is 30.0.

The worst case delay for a $k \times k$ parallel array multiplier is $2k+1$ times the worst case delay through a full adder cell [236]. Thus we would expect that the 32×32 multiplier would have a combinational logic depth of 65; however, our fit is empirical. If we force $t_{comb\ total}$ to be 65, the average error magnitude of the resulting fit is 6.9% and the maximum error is 11.5%.

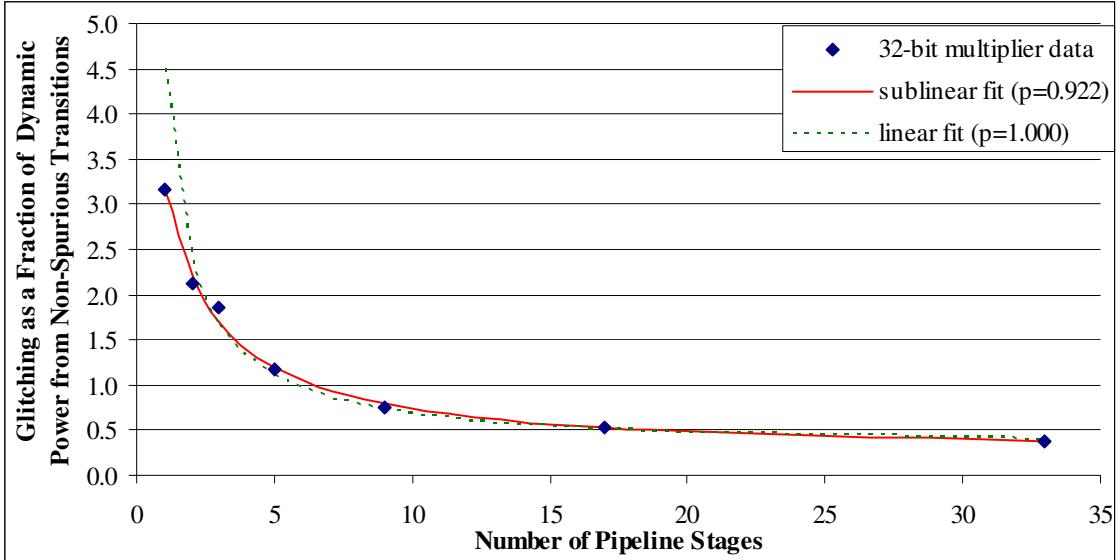


Figure A.6 This graph shows the fit of the sublinear model in Equation (A.18) and the linear model in Equation (A.19) versus the 32-bit multiplier data from [171].

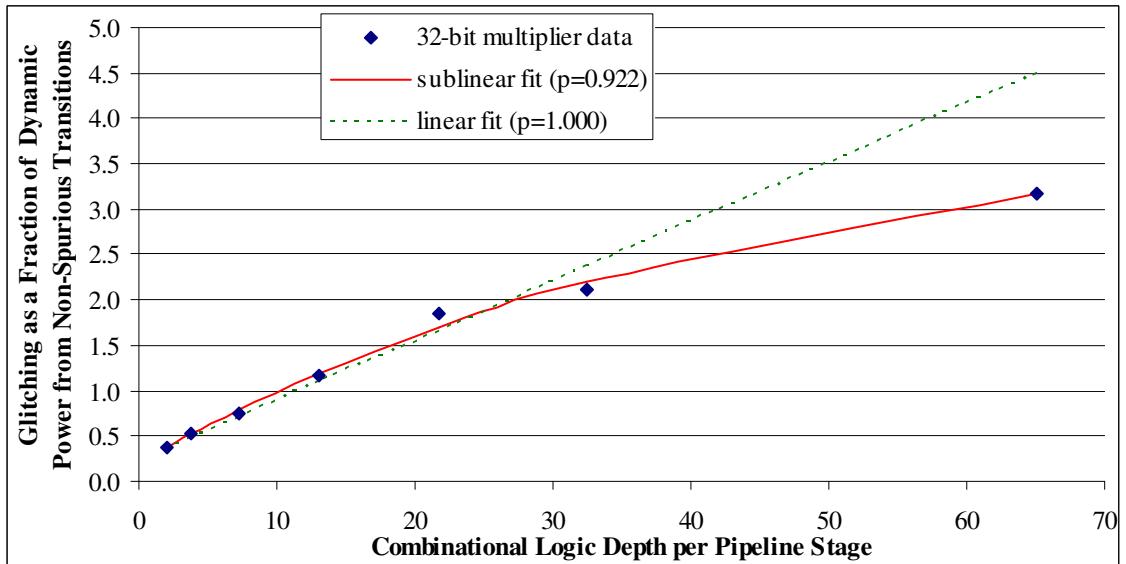


Figure A.7 This graph shows the fit of the sublinear model in Equation (A.18) and the linear model in Equation (A.19) versus the logic depth for the 32-bit multiplier data in [171]. We assumed that the unpipelined logic depth on the critical path was 65 and that pipelining divides this evenly.

If instead we assume a linear model with propagation probability p of 1, the relative least squares fit gives

$$0.262 + \frac{4.26}{n} \quad (\text{A.19})$$

The average error magnitude is 11.9% and the maximum error is 42.9%. The linear model does not provide a good fit of the data when there are few pipeline stages, as shown in Figure A.6.

The linear model in Equation (A.19) and the sublinear fit in Equation (A.18) produce very similar results when there are more pipeline stages and the logic depth is short, as shown in Figure A.7. However, the linear model over-estimates the glitching for the unpipelined 32-bit multiplier by 43%.

A.7.3 Dynamic power including glitching fit for a FPGA 64-bit multiplier

As there is a constant term, the expression in Equation (A.17) can also provide a suitable fit to the dynamic power, if we assume that the additional power for pipeline stage registers and the clock tree can be ignored. Fitting Equation (A.17) to the measured dynamic power for the 64-bit integer array multiplier in [237] by minimizing the relative least squares error gives:

$$14800 - 4710n(1 - 0.822^{14.55/n}) \quad (\text{A.20})$$

The average error magnitude is 1.8% and the maximum error is 3.1%. From the fit, the glitch propagation probability is 0.822, and the unpipelined combinational logic depth $t_{\text{comb total}}$ is 14.55. Again, the value for $t_{\text{comb total}}$ is substantially less than the value of $2k+1 = 2\times64+1 = 129$ that we might expect for a $k\times k$ array multiplier [236]. However, the maximum logic depth for the 64-bit array multiplier was reported in [237], and ranges from 105 logic elements with two pipeline stages to 43 logic elements with 64 pipeline stages. The pipeline stages were not balanced, as we would expect a 64 stage pipeline to have $1/32 = 0.03$ the logic depth of a 2 stage pipeline, but instead the ratio is $43/105 = 0.41$. Nevertheless, the empirical fit in Equation (A.20) to the dynamic power versus the number of pipeline stages for the 64-bit multiplier is good, as shown in Figure A.8.

If instead we assume a linear model with propagation probability p of 1, the relative least squares fit gives

$$1576 + \frac{13490}{n} \quad (\text{A.21})$$

The average error magnitude is 5.9% and the maximum error is 8.6%. The linear model provides a better fit for the 64-bit multiplier data than the 32-bit multiplier data, but is still not as good as the sublinear model in Equation (A.20), as shown in Figure A.8 and Figure A.9.

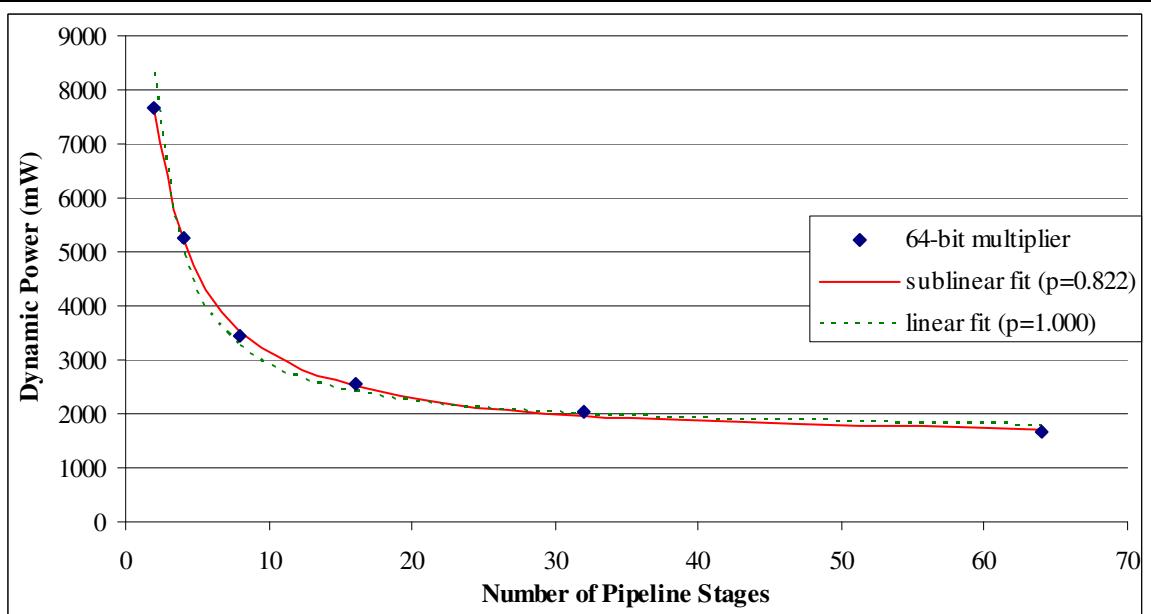


Figure A.8 This graph shows the fit of the sublinear model in Equation (A.20) and the linear model in Equation (A.21) versus the 64-bit multiplier data from [237].

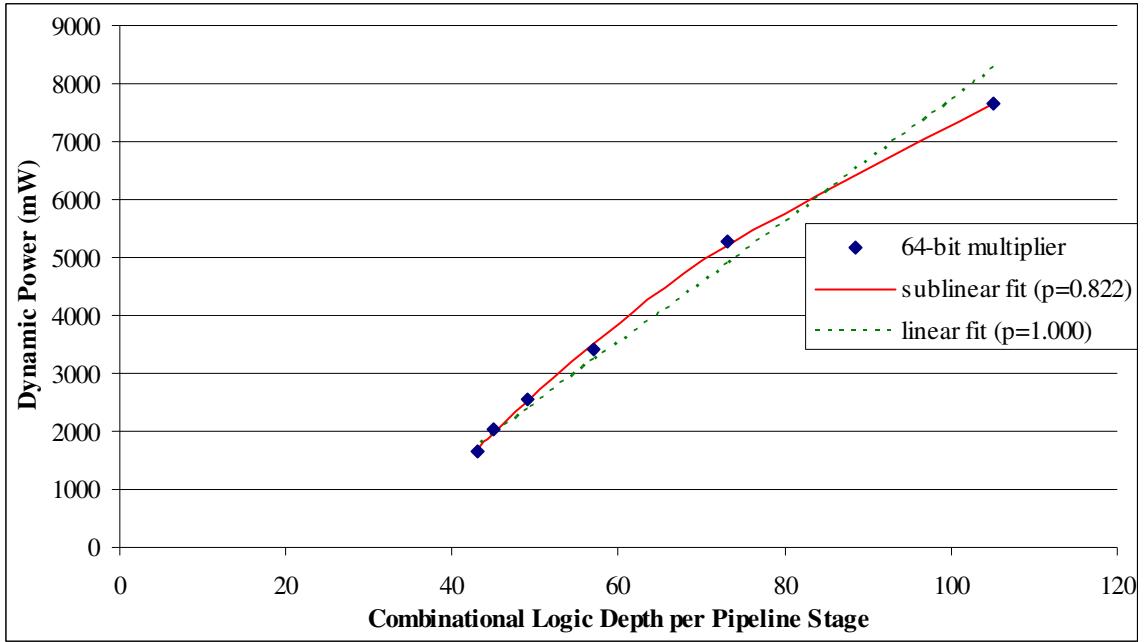


Figure A.9 This graph shows the fit of the sublinear model in Equation (A.20) and the linear model in Equation (A.21) versus the logic depth for the 64-bit multiplier data in [237].

A.7.4 Comments on glitching power versus pipeline depth in FPGAs

The relative power consumption due to glitching is higher in FPGAs than ASICs, and the relative power overhead for pipeline registers is smaller in FPGAs. Thus pipelining to reduce glitching has been explored in more detail for FPGAs [171][237] than in ASICs [135]. Our analysis of glitching versus logic depth has focused on FPGA pipelining data, as there is insufficient ASIC data to identify general trends versus the number of pipeline stages. We have focused on the glitching data for large FPGA multiplier implementations for several reasons. The data for smaller multipliers [171] and other functional blocks [237] has a smaller range in the number of pipeline stages. The data for other functional blocks is also noisier, and in some cases the dynamic power consumption actually increases with deeper pipelining due to the power overhead for more registers, which obscures any reduction in glitching power [237].

Limited FPGA routing tracks result in longer, more circuitous interconnects. Consequently, a greater portion of FPGA dynamic power consumption and path delay is due to the interconnect, and there are more unbalanced path delays, resulting in more glitching than in ASICs [237].

FPGA implementations often have unused flip-flops in the configurable logic blocks (CLBs) that are used to implement a design [171][237]. Thus the power overhead for registers is less in FPGAs than in ASICs.

As the clock tree and the register power are not detailed in [171] and [237], we have assumed that the dynamic power for registers of additional pipeline stages and the clock tree connecting to them is negligible compared to the combinational logic and interconnect in the FPGA. If this is not the case, the glitching power may decrease more with pipelining as the additional dynamic power for the registers and the clock tree mask the reduction in glitching. Thus we will use the sublinear fits to the two FPGA multipliers as a lower bound on glitching increase with logic depth, and use a linear model in the vein of Srinivasan et al. [186] as an upper bound.

A.7.5 Incorporating glitching power into the pipeline power model

From Section 3.3.3, our power model without glitching is

$$P_{total}(T) = \frac{1}{T_{min}} \left(\alpha_{clock\ gating} \frac{P_{dynamic}(T)}{P_{dynamic}(T_{min})} (1 - k_{leakage}) + k_{leakage} \frac{P_{leakage}(T)}{P_{leakage}(T_{min})} \right) (1 + \beta n^\eta) \quad (\text{A.22})$$

As glitching only affects the dynamic power for the combinational logic and not the registers, and clock gating does prevent glitching, the power model with glitching is

$$\begin{aligned} P_{total}(T) &= \frac{1}{T_{min}} \left(k_{leakage} \frac{P_{leakage}(T)}{P_{leakage}(T_{min})} \right) (1 + \beta n^\eta) \\ &\quad + \frac{1}{T_{min}} \left(\alpha_{clock\ gating} \frac{P_{dynamic}(T)}{P_{dynamic}(T_{min})} (1 - k_{leakage}) \right) (1 + \alpha_{glitching}(n) + \beta n^\eta) \end{aligned} \quad (\text{A.23})$$

where $\alpha_{glitching}(n)$ is the model for glitching as a fraction of the dynamic power due to non-spurious transitions.

In the vein of Srinivasan et al. [186], we consider a linear model for glitching with logic depth,

$$\alpha_{glitching}(n) = \frac{2}{n} \quad (\text{A.24})$$

This gives essentially the same overhead as assumed by Srinivasan et al. of 0.3 at seven pipeline stages with logic depth per stage of 16 FO4 delays, if we use their value of an unpipelined combinational logic delay of 110 FO4 delays¹. In our model we will use 180 FO4 delays for the unpipelined combinational logic delay.

Based on the above fits to glitching in 32-bit and 64-bit FPGA multipliers, we will also consider a sublinear model for glitching with logic depth,

$$\alpha_{glitching}(n) = 1 - \frac{n}{4}(1 - e^{-4/n}) \quad (\text{A.25})$$

This sublinear model has the same form as the fits in equations (A.18) and (A.20), as

$$p^{t_{comb\ total}/n} = e^{\ln(p)t_{comb\ total}/n} \quad (\text{A.26})$$

and $p < 1$, so $\ln(p) < 0$.

A.8 Graphs of optimal clock period with additional power overheads

Other power overheads have a significant impact on the optimal number of pipeline stages to minimize energy per operation as discussed in Section 3.6.2. The optimal clock period is smaller and the ratio of the clock period to the minimum pipeline stage delay (T/T_{min}) approaches one as a high performance design becomes optimal to minimize energy/operation with larger power overheads. This is illustrated in Figure A.10 and Figure A.11.

¹ The actual glitching combinational logic dynamic power overhead is 0.29 at seven pipeline stages or 15.7 FO4 delays, assuming an unpipelined combinational logic delay of 110 FO4 delays.

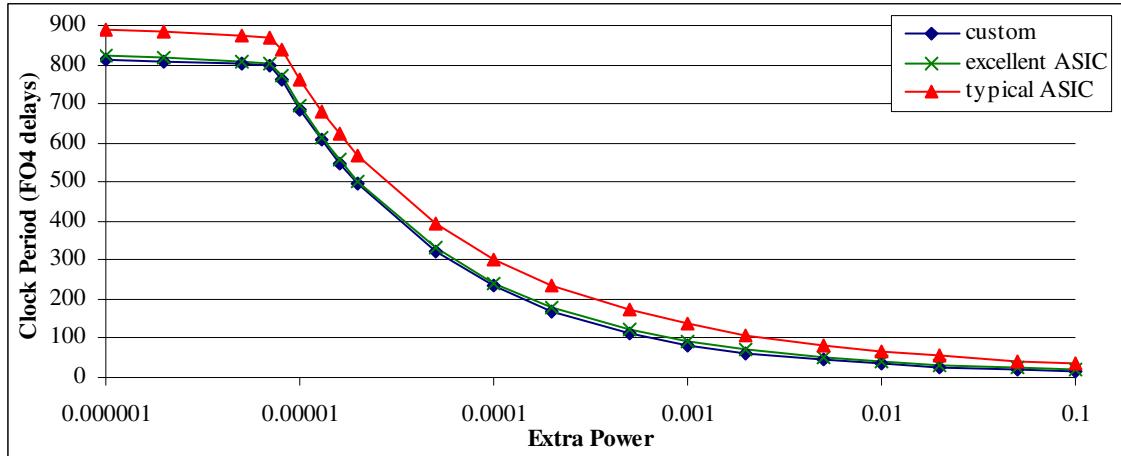


Figure A.10 The clock period to minimize energy/operation is shown for different power overheads.

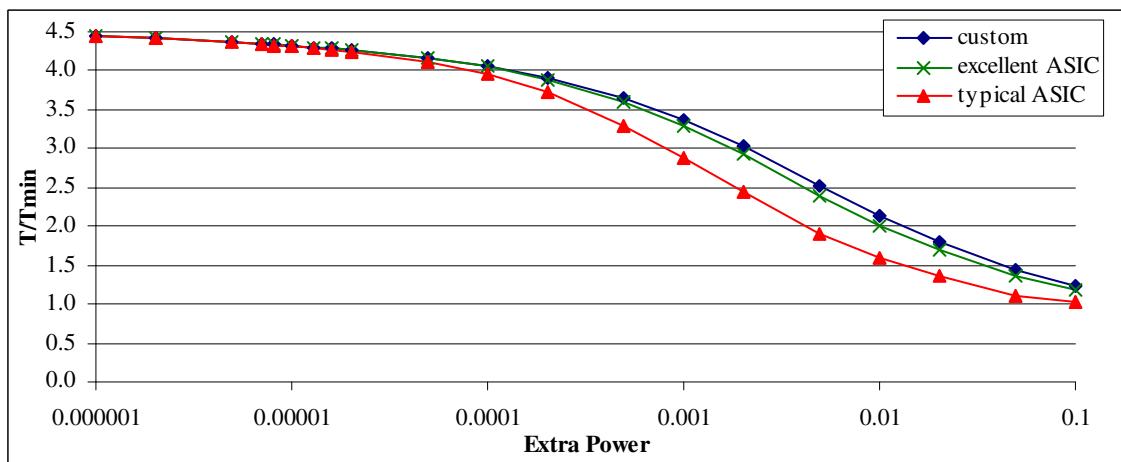


Figure A.11 The ratio of the clock period to the minimum pipeline stage delay T/T_{\min} to minimize energy/operation is shown for different power overheads.

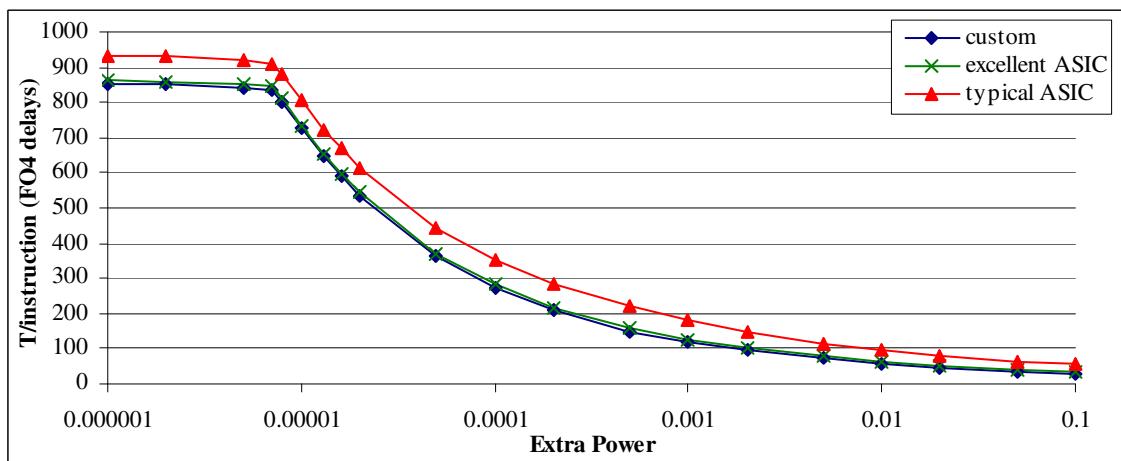


Figure A.12 The delay/instruction to minimize energy/operation is shown for different power overheads.

Appendix B Checking the Delay and Power Characterization of the 0.13um Libraries

In this appendix, we check the validity of the 0.13um library characterization versus the analytical models in Section 5.1 and determine empirical fits for the data. The analytical models help to identify where a library characterization is wrong, such as the leakage power for the 0.5V Vdd library as discussed in Section B.4.

The libraries were provided by Ashish Srivastava and Sarvesh Kulkarni at the University of Michigan. They found out that the 0.6V Vdd libraries had been incorrectly delay characterized using Vdd of 0.8V, thus for our work it was important to verify that the other libraries were characterized correctly and to scale delays accurately from a supply voltage of 0.8V to 0.6V. The libraries consist of inverter, NOR2, NOR3, NAND2 and NAND3 gates characterized for STMicroelectronics' 0.13um process technology in PowerArc. These libraries were used in Section 5.2, Chapter 4 and Chapter 7.

We examine the delay and power consumption normalized to the $V_{dd}=1.2V/V_{th}=0.23V$ data. The normalized data is averaged across the benchmarks. The libraries are compared at the minimum delay achievable for the particular supply and threshold voltages. The netlists used were the ISCAS'85 benchmarks that were delay minimized in Design Compiler with the PowerArc characterized 0.13um library with $V_{dd}=1.2V$ and $V_{th}=0.23V$.

B.1 Delay dependence on V_{dd} and V_{th} for the 0.13um library

Table B.1 presents the minimum delay for these netlists using libraries with different supply and threshold voltages in the same 0.13um technology, normalized to the minimum delay with $V_{dd}=1.2V/V_{th}=0.23V$. Delay scaling with Equation (5.9) and a typical value of $\alpha=1.3$ for 0.13um technology results in large errors fitting the delay data. Even the least squares fit of Equation (5.9), giving $\alpha=1.83$, to the delay data still has fit errors of up to 6.6%.

The larger values for α seem incorrect in the context presented by Sakurai and Newton in [173] and later research, which expect α values of around 1.2 to 1.3 in 0.13um technology. However, it should be remembered that slew is also affected by scaling of Vdd and Vth, and slew affects delay. Hence, higher order terms are not unexpected. In particular, integrating the saturation drain current in Equation (5.4), effectively averaging current over an interval per Veendrick [227], can give a term $(V_{dd} - V_{th})^{\alpha+1}$ with higher exponent.

Careful analysis of the derivation in Appendix B of [173] for the Sakurai-Newton alpha power law delay model, and substituting their expression for slew, suggests that the delay d may be of the form

$$d = k_1 \frac{V_{dd}}{(V_{dd} - V_{th})^\alpha} + k_2 \frac{V_{dd}^2}{(V_{dd} - V_{th})^{\alpha+1}} + k_3 \frac{(V_{dd} - V_{th})^{\alpha/2}}{V_{dd}} \ln\left(\frac{10(V_{dd} - V_{th})^{\alpha/2}}{eV_{dd}}\right) \quad (\text{B.1})$$

where α , k_1 , k_2 and k_3 are fitting variables for the experimentally measured delays. Various fits along these lines were tested, and the best fit to normalized delay was given by

$$d = 0.587 \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha+1}} + 0.241(V_{dd} - V_{th})^\alpha \quad (\text{B.2})$$

with a value for α of 1.101 which is closer to what might be expected for 0.13um technology. This fit is shown in the “higher order fit” column on the right of Table B.1, which achieves average error magnitude of 0.8% and the maximum error is only 2.1%. This fit does not directly include the impact of slew on delay. However, as the net scaling of circuit delay is accounted for, the scaling of slew is implicitly included.

Table B.1 This table lists the average of the minimum delay for the ISCAS'85 benchmarks using libraries with different supply and threshold voltages in the same 0.13um technology, normalized to the minimum delay with Vdd=1.2V/Vth=0.23V. Delay and slew were incorrectly characterized for Vdd=0.6V (shown in red – Vdd of 0.8V was used erroneously), so we examine delay fits to provide a delay estimate for Vdd=0.6V. The best “higher order” fit with Equation (B.2) achieved average error of only 0.8%. A least squares fit based on the simple Sakurai-Newton alpha power law scaling in Equation (5.9) gave a value for α of 1.83.

Estimates of the relative delay versus Vdd and Vth						
Vdd (V)	Vth (V)	Relative Delay	$\alpha=1.30$	$\alpha=1.66$	$\alpha=1.83$	Higher order fit
1.2	0.23	1.00	1.00	1.00	1.00	0.98
1.2	0.14	0.88	0.89	0.86	0.85	0.88
1.2	0.12	0.86	0.87	0.84	0.82	0.86
1.2	0.08	0.82	0.83	0.79	0.77	0.83
0.8	0.23	1.65	1.33	1.61	1.76	1.66
0.8	0.14	1.26	1.10	1.26	1.35	1.28
0.8	0.12	1.20	1.06	1.20	1.28	1.21
0.8	0.08	1.10	0.98	1.09	1.15	1.10
0.6	0.23	1.65	1.75	2.48	2.91	2.92
0.6	0.14	1.26	1.32	1.73	1.95	1.90
0.6	0.12	1.20	1.25	1.61	1.81	1.75
0.6	0.08	1.10	1.12	1.41	1.56	1.51
0.5	0.23	4.61	2.20	3.48	4.31	4.65
0.5	0.14	don't have this	1.51	2.16	2.55	2.59
0.5	0.12	2.33	1.41	1.97	2.31	2.32
0.5	0.08	1.95	1.24	1.67	1.92	1.91

Percentage error vs. measured delays						
Vdd (V)	Vth (V)	$\alpha=1.30$	$\alpha=1.66$	$\alpha=1.83$	Higher order fit	
1.2	0.23	0.0%	0.0%	0.0%	-1.7%	
1.2	0.14	1.2%	-2.0%	-3.4%	-0.1%	
1.2	0.12	1.1%	-2.8%	-4.5%	0.0%	
1.2	0.08	0.9%	-4.2%	-6.5%	0.6%	
0.8	0.23	-19.4%	-2.4%	6.6%	0.5%	
0.8	0.14	-13.0%	-0.1%	6.5%	0.9%	
0.8	0.12	-12.0%	0.0%	6.0%	0.8%	
0.8	0.08	-10.5%	-0.4%	4.7%	0.5%	
0.6	0.23 - 0.08	gate delay and slew were characterized incorrectly				
0.5	0.23	-52.3%	-24.4%	-6.5%	1.0%	
0.5	0.14	don't have this library				
0.5	0.12	-39.6%	-15.4%	-1.2%	-0.5%	
0.5	0.08	-36.5%	-14.2%	-1.4%	-2.1%	
Mean error magnitude		17.0%	6.0%	4.3%	0.8%	

Table B.2 This table shows the switching energy, which is the switching power multiplied by the clock period to factor out the change in the delay. Wire load switching power was measured as 37% of total switching power with the 0.13um Vdd=1.2V/Vth=0.23V on average across the ISCAS'85 benchmarks, and switching power for the output ports was 1%. The wire load and output port switching power scales directly with the supply voltage. The switching energy due to the gate input capacitance (in the last column) is determined by subtracting the switching energy of the wire load and output ports. Values for Vdd=0.6V are shown in red, as they are about 4% larger than they should be due to characterization at Vdd=0.8V (see Table B.3 for more details).

Switching Energy Normalized to Vdd=1.2V/Vth=0.23V					
Vdd (V)	Vth (V)	Total	Wire Load	Output Port	Gate Input Capacitance
1.2	0.23	1.000	0.370	0.010	0.621
1.2	0.14	1.031	0.370	0.010	0.651
1.2	0.12	1.038	0.370	0.010	0.658
1.2	0.08	1.046	0.370	0.010	0.667
0.8	0.23	0.415	0.164	0.004	0.247
0.8	0.14	0.436	0.164	0.004	0.268
0.8	0.12	0.442	0.164	0.004	0.274
0.8	0.08	0.451	0.164	0.004	0.282
0.6	0.23	0.234	0.092	0.002	0.139
0.6	0.14	0.245	0.092	0.002	0.151
0.6	0.12	0.249	0.092	0.002	0.154
0.6	0.08	0.254	0.092	0.002	0.159
0.5	0.23	0.155	0.064	0.002	0.089
0.5	0.14	did not have this library			
0.5	0.12	0.166	0.064	0.002	0.100
0.5	0.08	0.172	0.064	0.002	0.106

B.2 Switching power dependence on Vdd and Vth for the 0.13um library

The relative switching energy is shown in Table B.2. Switching of gate internal capacitances is not included – that is covered in Section 5.2.3. The switching energy is the switching power multiplied by the clock period (critical path delay) to factor out delay dependence. The switching power decreases quadratically with decreasing Vdd as per Equation (5.1).

As discussed in Section 5.2.2, by setting wire loads and output port loads to zero, we can determine their contribution to the switching power. The remainder of the switching power is then due to the gate input pin capacitances, as listed in column 6 of Table B.2.

We can then determine how gate input pin capacitance C_{in} varies with Vth and Vdd, by dividing by the V_{dd}^2 term in Equation (5.1). The gate input capacitance C_{in} is shown in the last column of Table B.3. We now see that the gate input capacitance was incorrectly characterized for the

$V_{dd}=0.6V$ cells, as the gate input capacitances are identical to those at $V_{dd}=0.8V$. To analyze this, a least squares fit to minimize the relative error of a first order Taylor series over the V_{dd} values of 1.2V, 0.8V and 0.5V gives

$$\text{Gate input capacitance } C_{in} = 0.579 + 0.121V_{dd} - 0.520V_{th} \quad (\text{B.3})$$

This fit has an average error magnitude of 1.0% and maximum error of 2.6%.

From this fit, the gate input capacitance values at $V_{dd}=0.6V$ are overestimated by 4.2% at $V_{th}=0.23V$ to 4.0% at $V_{th}=0.08V$. Compared to the total power (see data presented in Table B.7), the *overestimate* from switching power for C_{in} is as little as 0.7% of the total power at $V_{dd}=0.6V/V_{th}=0.08V$, and at most 1.5% at $V_{dd}=0.6V/V_{th}=0.23V$. Thus the net impact of the error is small and can be ignored.

Table B.3 This table shows the breakdown of circuit capacitance into that due to the wire loads, output ports, and gate input pins, normalized to the total capacitance at $V_{dd}=1.2V/V_{th}=0.23V$. The wire load and output port capacitances are fixed, while gate input capacitance varies with both V_{th} and V_{dd} . Capacitances here correspond to switching energy values in Table B.2 by multiplying by a factor of $(1.2/V_{dd})^2$. Values for $V_{dd}=0.6V$ are shown in red as they are wrong – they were characterized incorrectly with $V_{dd}=0.8V$, as is clear by comparing with the $V_{dd}=0.8V$ values, which are identical.

Capacitance Normalized to $V_{dd}=1.2V/V_{th}=0.23V$					
V_{dd} (V)	V_{th} (V)	Total	Wire Load	Output Port	Gate Input Capacitance
1.2	0.23	1.000	0.370	0.010	0.621
1.2	0.14	1.031	0.370	0.010	0.651
1.2	0.12	1.038	0.370	0.010	0.658
1.2	0.08	1.046	0.370	0.010	0.667
0.8	0.23	0.934	0.370	0.010	0.555
0.8	0.14	0.982	0.370	0.010	0.602
0.8	0.12	0.995	0.370	0.010	0.615
0.8	0.08	1.014	0.370	0.010	0.635
0.6	0.23	0.934	0.370	0.010	0.555
0.6	0.14	0.982	0.370	0.010	0.602
0.6	0.12	0.995	0.370	0.010	0.615
0.6	0.08	1.014	0.370	0.010	0.635
0.5	0.23	0.890	0.370	0.010	0.511
0.5	0.14	did not have this library			
0.5	0.12	0.954	0.370	0.010	0.574
0.5	0.08	0.992	0.370	0.010	0.613

B.3 Internal power dependence on Vdd and Vth for the 0.13um library

The internal power has two components: the short circuit power, and the switching power for internal “parasitic” capacitances. From Equation (5.5), for a fit to the internal power we expect terms of the form $(V_{dd} - 2V_{th})^a$ for the short circuit power. From Equation (5.1), we also expect a term of the form aV_{dd}^2 for the switching power of the internal capacitances. A good fit was provided by

$$\begin{aligned} P_{internal} &= P_{switching \ internal \ capacitances} + P_{short \ circuit} \\ &= f(0.413V_{dd}^2 + 0.958(V_{dd} - 2V_{th})^{3.183} + 0.039(V_{dd} - 2V_{th})^{0.162}) \end{aligned} \quad (\text{B.4})$$

where we can factor out the frequency f , to give the internal energy per cycle. Table B.4 shows the internal power and the fit to it. The fit has an average error magnitude of 0.4% and maximum error of 1.0%. The terms in Equation (B.4) are described in Section 5.2.3.

The incorrect characterization at Vdd=0.6V results in internal energy being underestimated by up to 6.3% at Vth=0.08V. It appears that this error is caused solely by the incorrect slew characterization which affects only short circuit power, as there is only a 0.6% underestimate in internal power at Vth=0.23V where short circuit energy gives only a 17% contribution to the total internal energy. Compared to the total power (see data presented in Table B.7), the *underestimate* of short circuit power due to incorrect slews is as little as 0.2% of the total power at Vdd=0.6V/Vth=0.23V, and at most 1.6% at Vdd=0.6V/Vth=0.12V. Thus the net impact of the error in total power at Vdd=0.6V is small and can be ignored, noting that the overestimate of switching power from C_{in} and underestimate of short circuit power cancel out to some extent, causing at most a 0.6% underestimate at Vdd=0.6V/Vth=0.08V and at most a 1.3% overestimate at Vdd=0.6V/Vth=0.23V.

Table B.4 The internal energy (internal power with delay factored out) averaged across the ISCAS'85 benchmarks is shown in column three for different Vdd and Vth values. Values were normalized to internal energy at Vdd=1.2V/Vth=0.23V. The internal energies for Vdd=0.6V are inaccurate (shown in red), due to the incorrect characterization of slews. The fit from Equation (B.4) has an average error magnitude of 0.4% and maximum error of 1.0%. The last two columns show the switching energy due to gate internal capacitances and the short circuit energy, estimated from the fit from Equation (B.4).

Components of Internal Energy			
Vdd (V)	Vth (V)	Internal Energy	Fit
1.2	0.23	1.000	0.999
1.2	0.14	1.369	1.368
1.2	0.12	1.474	1.474
1.2	0.08	1.718	1.719
0.8	0.23	0.330	0.328
0.8	0.14	0.416	0.419
0.8	0.12	0.449	0.451
0.8	0.08	0.534	0.532
0.6	0.23	0.178	0.179
0.6	0.14	0.196	0.207
0.6	0.12	0.206	0.219
0.6	0.08	0.237	0.253
0.5	0.23	0.127	0.127
0.5	0.14	did not have this library	
0.5	0.12	0.147	0.148
0.5	0.08	0.169	0.167

B.4 Leakage dependence on Vdd and Vth for the 0.13um library

The normalized leakage power is shown in Table B.5. As Vth is reduced, we see a substantial increase in leakage, about 56× from 0.23V to 0.08V. As Vdd is reduced the leakage decreases somewhat, by about 3× from 1.2V to 0.6V. From Equation (5.7) for the leakage power, we get the following fit

$$P_{leakage} = 158V_{dd}e^{-26.9V_{th}+0.770V_{dd}} \quad (\text{B.5})$$

The values from this fit are listed in column four of Table B.5.

The incorrect leakage characterization at Vdd=0.5V became apparent when trying to fit Equation (5.7) to the leakage data, as the V_{dd} term before the exponential would get a fitted exponent of -0.574 (fitting the leakage data for Vdd values of 1.2V, 0.8V and 0.5V), which is incorrect. In contrast, fitting the leakage data for Vdd values of 1.2V, 0.8V and 0.6V gives a fitted V_{dd}

exponent of 1.051, which is about what we expect – forcing this exponent to 1.0 does not increase the fit error for these data points. The fit in Equation (B.5) to the leakage data at Vdd of 1.2V, 0.8V and 0.6V gives average error magnitude of 2.6% (not bad considering the function is exponential) and maximum error of 6.0%.

The overestimate in leakage power characterization at Vdd=0.5V ranges from a 26% overestimate at Vth=0.08V to 35% overestimate at Vth=0.12V. The impact of this on total power ranges from a 2.3% overestimate at Vth=0.23V (as leakage is a small portion of total power at this high threshold voltage) to 16.2% at Vth=0.08V. These errors in leakage power at Vdd=0.5V are significant, and would have to be corrected, but we don't use the 0.5V characterization except in this chapter.

Table B.5 The leakage power averaged across the ISCAS'85 benchmarks is shown in the third column for different Vdd and Vth values, normalized to the leakage power at Vdd=1.2V/Vth=0.23V. The leakage characterization for Vdd=0.5V was incorrect and these values are listed in red. The fit in Equation (B.5) gives average error magnitude of 2.6% and maximum error of 6.0% versus the leakage values at Vdd of 1.2V, 0.8V and 0.6V.

Vdd (V)	Vth (V)	Normalized Leakage Power	Fit	Error
1.2	0.23	1.00	0.99	-1.3%
1.2	0.14	11.40	11.08	-2.8%
1.2	0.12	19.38	18.97	-2.1%
1.2	0.08	52.43	55.58	6.0%
0.8	0.23	0.47	0.48	2.1%
0.8	0.14	5.56	5.43	-2.4%
0.8	0.12	9.58	9.29	-3.0%
0.8	0.08	26.61	27.23	2.3%
0.6	0.23	0.30	0.31	3.9%
0.6	0.14	3.55	3.49	-1.8%
0.6	0.12	6.15	5.97	-2.9%
0.6	0.08	17.33	17.51	1.0%
0.5	0.23	0.31	0.24	-23.3%
0.5	0.14	did not have this library		
0.5	0.12	6.23	4.61	-26.0%
0.5	0.08	17.01	13.51	-20.6%

Table B.6 The critical path delay, leakage power, internal energy, and switching energy are shown for different Vdd and Vth values, normalized to the Vdd=1.2V/Vth=0.23V values. These values are the average across the ISCAS'85 netlists that were delay minimized with Design Compiler using the 0.13um library with Vdd=1.2V and Vth=0.23V. Corrected values are shown in blue. The impact of delay (1/critical_path_delay = frequency) has been factored out of the internal power and switching power. Primary input slew was 0.1ns. Output port loads were 3fF. Wire loads were $3+2\times\#fanouts$ fF. Temperature was 25°C.

Vdd (V)	Vth (V)	Delay	Leakage Power	Internal Energy	Switching Energy
1.2	0.23	1.000	1.000	1.000	1.000
1.2	0.14	0.880	11.402	1.369	1.031
1.2	0.12	0.860	19.383	1.474	1.038
1.2	0.08	0.822	52.431	1.718	1.046
0.8	0.23	1.651	0.474	0.330	0.415
0.8	0.14	1.264	5.563	0.416	0.436
0.8	0.12	1.203	9.577	0.449	0.442
0.8	0.08	1.098	26.605	0.534	0.451
0.6	0.23	2.924	0.299	0.179	0.228
0.6	0.14	1.902	3.553	0.207	0.240
0.6	0.12	1.753	6.153	0.219	0.242
0.6	0.08	1.508	17.332	0.253	0.247
0.5	0.23	4.607	0.240	0.127	0.155
0.5	0.14		don't have this library		
0.5	0.12	2.334	4.610	0.147	0.166
0.5	0.08	1.948	13.507	0.169	0.172

Table B.7 Detailed breakdown of total power into the various components of power consumption. At the top is shown power normalized to the total power at Vdd=1.2V/Vth=0.23V. At the bottom are shown the percentages of total power for the particular library. Switching power and short circuit power were determined by multiplying by 1/critical_path_delay. Note that the significance of leakage depends on the switching activity – switching activity was multiplied by a fraction so that with the Vdd=1.2V/Vth=0.23V library, leakage was about 1% of total power (1.1% as shown in the bottom table).

Vdd (V)	Vth (V)	Critical Path Delay	Leakage Power	Short Circuit Power	Switching power due to				Total Power
					Gate Internal Capacitances	Gate Input Pin Capacitances	Wire Loads	Output Ports	
1.2	0.23	1.000	0.011	0.174	0.256	0.347	0.207	0.005	1.000
1.2	0.14	0.880	0.121	0.378	0.290	0.414	0.235	0.006	1.444
1.2	0.12	0.860	0.206	0.440	0.297	0.428	0.240	0.006	1.617
1.2	0.08	0.822	0.557	0.588	0.311	0.453	0.251	0.007	2.168
0.8	0.23	1.651	0.005	0.017	0.069	0.083	0.056	0.001	0.231
0.8	0.14	1.264	0.059	0.053	0.090	0.118	0.073	0.002	0.395
0.8	0.12	1.203	0.102	0.067	0.094	0.127	0.076	0.002	0.469
0.8	0.08	1.098	0.283	0.105	0.104	0.144	0.084	0.002	0.721
0.6	0.23	2.924	0.003	0.004	0.022	0.025	0.018	0.000	0.073
0.6	0.14	1.902	0.038	0.013	0.034	0.043	0.027	0.001	0.155
0.6	0.12	1.753	0.065	0.017	0.036	0.047	0.029	0.001	0.196
0.6	0.08	1.508	0.184	0.030	0.042	0.057	0.034	0.001	0.348
0.5	0.23	4.607	0.003	0.002	0.010	0.011	0.008	0.000	0.033
0.5	0.14				don't have this library				
0.5	0.12	2.334	0.049	0.008	0.019	0.024	0.015	0.000	0.116
0.5	0.08	1.948	0.144	0.014	0.023	0.031	0.018	0.000	0.230

Vdd (V)	Vth (V)	Percentage of total power						Total Power	
		Switching power due to				Leakage Power	Short Circuit Power		
		Gate Internal Capacitances	Gate Input Pin Capacitances	Wire Loads	Output Ports				
1.2	0.23	1.1%	17.4%	25.6%	34.7%	20.7%	0.5%		
1.2	0.14	8.4%	26.2%	20.1%	28.6%	16.3%	0.4%		
1.2	0.12	12.7%	27.2%	18.4%	26.4%	14.9%	0.4%		
1.2	0.08	25.7%	27.1%	14.3%	20.9%	11.6%	0.3%		
0.8	0.23	2.2%	7.2%	29.8%	36.1%	24.1%	0.6%		
0.8	0.14	15.0%	13.4%	22.8%	30.0%	18.4%	0.5%		
0.8	0.12	21.7%	14.3%	20.2%	27.1%	16.3%	0.4%		
0.8	0.08	39.2%	14.6%	14.4%	19.9%	11.6%	0.3%		
0.6	0.23	4.4%	6.1%	29.9%	34.8%	24.2%	0.6%		
0.6	0.14	24.4%	8.5%	21.7%	27.4%	17.5%	0.5%		
0.6	0.12	33.3%	8.8%	18.6%	23.9%	15.0%	0.4%		
0.6	0.08	52.9%	8.6%	12.2%	16.2%	9.8%	0.3%		
0.5	0.23	7.7%	6.6%	29.1%	32.5%	23.5%	0.6%		
0.5	0.14			don't have this library					
0.5	0.12	42.3%	7.1%	16.4%	20.6%	13.3%	0.3%		
0.5	0.08	62.4%	6.1%	9.9%	13.3%	8.0%	0.2%		

Table B.8 The last column of this table shows the energy consumption, determined from the product of the critical path delay and the total power. Due to the increased delay with higher V_{th} and lower V_{dd}, the energy savings are not as large as the power reduction.

V _{dd} (V)	V _{th} (V)	Critical Path Delay	Total Power	Energy = Delay x Power
1.2	0.23	1.000	1.000	1.000
1.2	0.14	0.880	1.444	1.271
1.2	0.12	0.860	1.617	1.392
1.2	0.08	0.822	2.168	1.783
0.8	0.23	1.651	0.231	0.382
0.8	0.14	1.264	0.395	0.499
0.8	0.12	1.203	0.469	0.564
0.8	0.08	1.098	0.721	0.791
0.6	0.23	2.924	0.073	0.214
0.6	0.14	1.902	0.155	0.295
0.6	0.12	1.753	0.196	0.344
0.6	0.08	1.508	0.348	0.525
0.5	0.23	4.607	0.033	0.153
0.5	0.14	don't have this library		
0.5	0.12	2.334	0.116	0.271
0.5	0.08	1.948	0.230	0.448

B.5 Summary of corrections to the 0.13um library

The preceding sections analyzed the major characterization problems with the V_{dd}=0.5V and V_{dd}=0.6V characterizations. For the incorrectly characterized delay and power data, we now have reasonably accurate fitted values. Table B.6 presents a summary of the corrected data across the different V_{dd} and V_{th} values. The individual contributions to the total power are shown in Table B.7. Some additional problems have been addressed, as described below. Also note that the minimum input pin capacitance was 1.0fF, which appears to have been a limit set in characterization, which seems reasonable.

The NOR2XL, NOR3XL, NAND2XL and NAND3XL cells in the V_{dd}=0.5V libraries were only characterized for a capacitance range of 0.0fF to 0.1fF, which is insufficient. These XL cells were removed from the V_{dd}=0.5V library, and replaced in the netlists for static timing and power analysis at V_{dd}=0.5V and for the comparison to normalize versus V_{dd}=1.2V/V_{th}=0.23V. The NOR2X1 pin A capacitance was far too large at V_{dd}=0.5V/V_{th}=0.23V. It was six times that of pin B, a difference far greater than for input pins of any other cell. Examining other NOR2X1

cells, the pin A and pin B input capacitances were very close in value, so the NOR2X1 pin A capacitance was corrected to equal that of the pin B capacitance which looked correct from comparison to a range of other cells. Due to these issues, the $V_{dd}=0.5V$ library characterization was not used further.

There are also libraries for when the input voltage (V_{in}) swing exceeds V_{dd} , for $V_{in}=1.2V/V_{dd}=0.8V$. This is important, because the larger input voltage swing increases rise delay for a negative polarity gate but reduces fall delay (see more detailed discussion in Appendix D.2 and Section 7.4.2). I have not tested the $V_{in}=1.2V/V_{dd}=0.8V$ libraries outside of power minimization and checking input pin capacitances, but they appear to be correct except for PMOS leakage power. When $V_{in} > V_{dd}$, the PMOS transistors are reverse biased, and should have lower leakage. However, the $V_{in}=1.2V/V_{dd}=0.8V$ library gives identical leakage values to the $V_{in}=0.8V/V_{dd}=0.8V$ library. This results in an overestimate of leakage power when using dual supply voltages of 1.2V and 0.8V, but as only a small fraction of cells on the boundary of the VDDL region have VDDH inputs, the net error in total power is small and may be neglected.

The base library had nine inverter sizes: XL, X1, X2, X3, X4, X8, X12, X16, X20; and four sizes for NAND2, NAND3, NOR2 and NOR3: XL, X1, X2, X4. For $V_{in}=1.2V/V_{dd}=0.6V$, these were incorrectly characterized with $V_{in}=1.2V/V_{dd}=0.8V$. The delay was corrected in the manner described in Section 5.2.1, using the same delay scaling as was used to scale from $V_{in}=0.8V/V_{dd}=0.8V$ to $V_{in}=0.6V/V_{dd}=0.6V$.

For the extended cell sizes (X6, X10, X14, X18 for the inverter; and X5, X6, X7 and X8 for the other gates), there was no characterization for $V_{dd}=0.6V$ with $V_{in} > V_{dd}$. The incorrect delay characterization with $V_{in}=0.8V/V_{dd}=0.8V$ was corrected for $V_{in}=0.6V/V_{dd}=0.6V$ as described above, and was also used for $V_{in}=1.2V/V_{dd}=0.6V$ [private communication with Sarvesh Kulkarni]. Using $V_{in} = V_{dd}$ for the case where $V_{in} > V_{dd}$ turns out to be a little pessimistic for

delay, as $V_{in} > V_{dd}$ causes a slight net decrease in circuit delay of about 1% to 3% typically. $V_{in} > V_{dd}$ increases the rise delay of a negative polarity gate, but provides a larger decrease in fall delay at $V_{in}=1.2V/V_{dd}=0.8V$ versus $V_{in}=0.8V/V_{dd}=0.8V$. Using $V_{in} = V_{dd}$ for the case where $V_{in} > V_{dd}$ has no significant impact on total power, providing the correct V_{in} is used to calculate switching power for the gate input pins.

The University of Michigan researchers used the incorrect 0.6V V_{dd} delay data in their results with V_{th} of 0.12V in [124], and with V_{th} of 0.23V and 0.12V in [188]. Based on our empirical fit to the delay data with V_{dd} of 0.5V, 0.8V and 1.2V, this error results in the delay being 31% lower at 0.12V V_{th} and 43% lower at 0.23V V_{th} . These underestimates lead to an exaggeration of the multi- V_{dd} power savings. They reported average power savings of 30% or more using dual supply voltages of 1.2V and 0.6V [124][188]. Sarvesh Kulkarni has now rerun their analysis, delay scaling with our empirical fit in Equation (5.10). He wrote that multi- V_{dd} with 1.2V and 0.8V achieved better results than using 1.2V and 0.6V, and the average power savings were only 22%. We compare our multi- V_{dd} results to their updated results in Chapter 7.

Appendix C Complexity of Power Minimization with Gate Sizing

Gate sizing is a sub-problem of technology mapping, which was observed to be NP-complete in 1987 [118]. Technology mapping considers different logic mappings in addition to gate sizing. The technology mapping problem is to find the minimal cost implementation of a circuit from a library of logic cells. The cost metric may be delay, area or power. A combinational circuit can be represented as a directed acyclic graph. The technology mapping problem is equivalent to finding the minimum cost cover of the directed acyclic graph. This is the same problem as in compiler code generation, which was shown to be NP-complete [1].

In contrast, the “gate sizing” problem assumes that the logic gate (e.g. NAND2, NOR3, XOR2) used to implement each node of the graph is fixed, but there are different standard cell implementations of a logic gate in the library with different costs. For example, NOR2 cells with NMOS transistor widths of 1 \times , 2 \times , 4 \times and 8 \times the minimum width and a fixed PMOS to NMOS transistor width ratio of 4:1 to balance rise and fall delays. Cells with larger transistor sizes have increased area and consume more power. Larger cell sizes will be faster if driving a large load, but may be slower if driving a small load due to loading the fanins. Chakraborty and Murgai showed that gate resizing to minimize circuit delay is NP-complete, if the delay of a gate depends on the fanout capacitive loads, which it does for any real circuit [31]. They showed this by mapping the 3SAT problem, Boolean satisfiability where each clause has at most three literals, onto the gate sizing problem. It has also been shown that gate sizing assuming different load independent rise and fall delays is an NP-complete problem, even assuming circuits with chain or tree topology [158]. This is contrary to technology mapping for trees being a polynomial time solvable problem, where it was assumed that rise and fall delays are equivalent [118]. It has been common for circuit optimization researchers to assume equivalent rise and fall delays, but this is quite inaccurate for today’s technologies, especially when multi-Vdd is also considered. For

example, see Figure 7.5 and Figure 7.6 where the change in rise delay for different gate size alternatives can be double the change in fall delay, or nearly quadruple when a 0.8V gate is driven with a 1.2V input.

We are interested in power minimization subject to a delay constraint. Satisfying the delay constraint is an NP-complete problem as described above. Often we will start power minimization from a circuit that has been sized to meet the delay constraint, in which case meeting the delay constraint is trivial, but the complexity of gate sizing to minimize the power is not clear. So we will show that power minimization with gate sizing is an NP-complete problem.

C.1 Proof of NP-Completeness of Power Minimization with Gate Sizing

To our knowledge, it has not been shown previously that the problem of optimally gate sizing a circuit to minimize power is NP-complete. To prove that the problem is NP-complete, we must show that a solution can be verified in polynomial time and that there is a polynomial time algorithm that maps an NP-complete problem onto our problem [49].

We pose the power minimization problem as a decision problem. Given a directed acyclic graph $G=(V,E)$ and a set S of discrete sizes that each gate $v \in V$ can be, is there a circuit assignment with a size for each gate such that the total circuit power P_{total} is less than or equal to a given constraint P_{max} ? The total power for the circuit is the sum of the power for each gate. A gate's power depends on the load capacitance, thus the size of fanout gates, and the input slew from fanins. A gate's delay and signal slew also depend on these values.

Verifying a gate size assignment solution in polynomial time is straightforward. Performing static timing analysis and static power analysis for a combinational circuit, in the manner detailed in Section Appendix E, has linear computation runtime with circuit size. We then check that the total circuit power $P_{total} \leq P_{max}$.

To map an NP-complete problem onto our problem, we consider mapping graph coloring to gate sizing. The graph coloring decision problem is: can we assign a color to each node on an undirected graph, such that no two adjacent vertices have the same color, using at most k colors?

The first step is to make the undirected graph in the coloring problem a directed acyclic graph. We can pick any node in the (connected) undirected graph, and perform breadth-first search from that node assigning directed edges from that node. The polynomial time procedure for converting the undirected graph $G(V,E)$ to a directed graph is:

1. The current set $C = \{w\}$, where $w \in V$ is an arbitrarily chosen node. (Topological level 1.)
The set of nodes which has been done is $D = \emptyset$.
2. Add C to the nodes which have been done, $D = D \cup C$. Terminate if $D = V$.
3. The next level is nodes connected to C , $N = \{v \in V : v \notin D \text{ and } \exists u \in C \text{ such that } (u,v) \in E\}$.
Make edges directed from C to N .
4. If any nodes in N have connections to nodes in N , arbitrarily remove nodes from N until no connections within this set (e.g. in reverse lexical order of node naming).
5. Update $C = N$. (Topological Level = level +1.) Go to step 2.

To illustrate this procedure, we use the W_6 wheel graph [235] shown in Figure C.1(a). A four coloring of the vertices of it is shown in Figure C.1(b). If only three colors are used to color the vertices of the W_6 wheel graph then some of the connected vertices will have the same color.

Following this procedure for the graph in Figure C.1(a), as illustrated in Figure C.1(c):

1. Pick node a to start with. $C = \{a\}$. $D = \emptyset$. Level = 1.
2. $D = \{a\}$
3. $N = \{b, c, d\}$, making edges directed.
4. Node d connects to node c , so remove d from N . Node c connects to b , so remove c from N .
End result $N = \{b\}$.
5. $C = N = \{b\}$. Level = 2.
2. $D = \{a\} \cup \{b\} = \{a, b\}$
3. $N = \{c, e\}$, making edges directed from C to N .
4. Node e connects to node c , so remove e from N . End result $N = \{c\}$.
5. $C = N = \{c\}$. Level = 3.
2. $D = \{a, b\} \cup \{c\} = \{a, b, c\}$
3. $N = \{d, e, f\}$, making edges directed from C to N .
4. Node f connects to nodes d and e , so remove f from N . End result $N = \{d, e\}$.
5. $C = N = \{d, e\}$. Level = 4.
2. $D = \{a, b, c\} \cup \{d, e\} = \{a, b, c, d, e\}$
3. $N = \{f\}$, making edges directed from C to N .
4. No connections between nodes in N .
5. $C = N = \{f\}$. Level = 5.
2. $D = \{a, b, c, d, e\} \cup \{f\} = V$. Finished.

This procedure is at worst $O(|V||E|)$, because in each iteration of the procedure we remove at least one vertex and there are at most $O(|E|)$ edges to traverse on an iteration. There are at most $|V|(|V| - 1)/2$ edges in E , hence the complexity is at worst $O(|V|^3)$. So we can convert the undirected graph to a directed graph in polynomial time. The resulting directed graph is acyclic by construction, as no later nodes are assigned an edge going to a node at an earlier topological level.

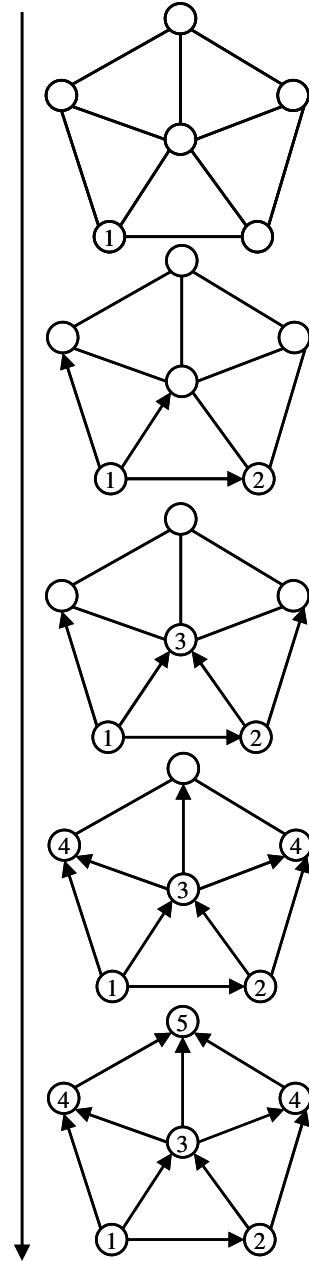
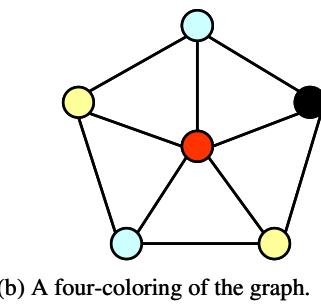
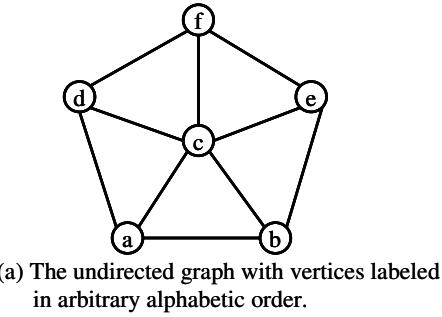


Figure C.1 On the left is shown the W_6 wheel graph [235] and a four coloring of the vertices of it. If only three colors are used to color the vertices of the graph then some of the connected vertices will have the same color. On the right is shown the procedure for converting the graph from an undirected graph to a directed acyclic graph, with the topological level specified inside each node.

We can then map the k -coloring decision problem to the power minimization decision problem.

We specify a library of cell sizes where sizes correspond to integers: 1, 2, 3, and so forth.

Likewise, we assign integers 1, 2, ..., k for the k colors, and each gate size represents a different

color. For a real circuit we would want to specify separate rise and fall timing characteristics, but it is not important for this proof, so we will assume they are the same. As the delay is unconstrained, the delay of each gate $v \in V$ can be say $d_v = 1$. We use the output slew s_v of a gate v to encode the gate size $j \in S$, $s_v = j$. Then we define the power of a gate v of size j as follows:

$$P_v = \begin{cases} |V|^j, & \text{if there does not exist } u \in \text{fanin}(v) \text{ such that } s_u = j \\ |V|^{k+1}, & \text{if there exists } u \in \text{fanin}(v) \text{ such that } s_u = j \end{cases} \quad (\text{C.1})$$

and the total power is given by

$$P_{\text{total}} = \sum_{v \in V} P_v \quad (\text{C.2})$$

The problem is to find an assignment of a gate size to each gate, such that the total circuit power P_{total} is less than or equal to $|V|^{k+1}$, where $|V|$ is the cardinality of V . We need to show that there is a valid k -coloring if and only if there is a valid sizing with $P_{\text{total}} \leq |V|^{k+1}$.

Firstly, we show that a valid k -coloring implies a valid sizing with power $P_{\text{total}} \leq |V|^{k+1}$. As we have a valid k -coloring, no adjacent vertices have the same color (size). Hence the power of each gate is $|V|^j$ where j is the integer corresponding to the size and color. The largest gate size used is less than or equal to k , so we have

$$P_{\text{total}} = \sum_{v \in V} P_v = \sum_{v \in V} |V|^{s_{\text{ize}}(v)} \leq |V| \times |V|^{\max_{v \in V} \{s_{\text{ize}}(v)\}} \leq |V| \times |V|^k = |V|^{k+1} \quad (\text{C.3})$$

Thus the power constraint is satisfied.

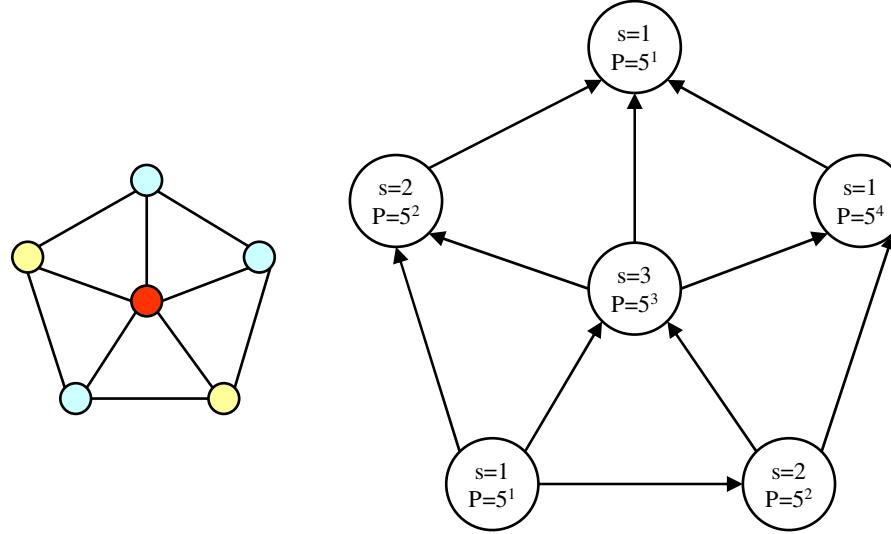
Secondly, we show that an invalid coloring implies an invalid sizing, violating the power constraint $P_{\text{total}} \leq |V|^{k+1}$. If the coloring is invalid, we either have two adjacent vertices of the same color, or there are more than k colors. This also implies we are not dealing with the trivial case where there is only one vertex. If two vertices have the same color, there is a gate with a fanin of

the same size and from Equation (C.1) that gate has power $|V|^{k+1}$. If there are more than k colors, then there is a gate with size of $k+1$ and power of $|V|^{k+1}$. As there is more than one vertex, another vertex has at least a power of $|V|$, so $P_{total} \geq |V|^{k+1} + |V| > |V|^{k+1}$ and the power constraint is violated.

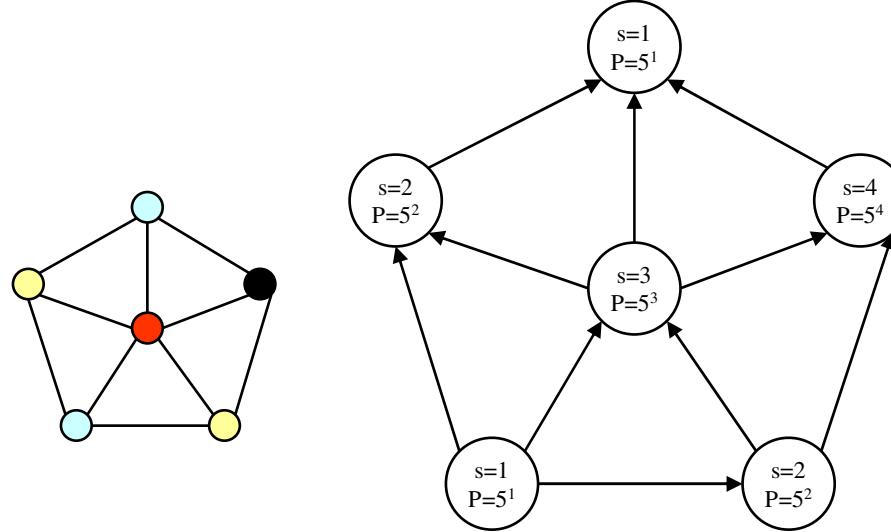
The largest cell size in the minimum power circuit that satisfies the delay constraint corresponds to the minimum number of colors needed to color the graph. This is illustrated with the W_6 wheel graph in Figure C.2.

Thus we have a polynomial time mapping of an NP-complete problem to our gate sizing problem. As we can also verify a gate sizing solution in polynomial time, gate sizing to minimize power is an NP-complete problem. A further consequence is that performing transistor-level sizing, transistor threshold voltage assignment and gate supply voltage assignment to minimize power is an NP-complete problem. This is because gate sizing to minimize power is a subset of this problem and technology mapping to minimize power is a superset of this problem, but both are NP-complete, so the intermediate problem (in the sense of solution space size) with multi-V_{th} and multi-V_{dd} is NP-complete.

To formulate the NP-completeness proof, we used a rather unpleasant non-convex formulation for gate power that does not resemble typical gate power. We can try to reduce the complexity of the gate sizing/multi-V_{th}/multi-V_{dd} problem by modeling gate delay and power using convex models, hoping that they are sufficiently accurate to provide good results with convex optimization.



(a) Assignment of three colors to the graph fails to ensure each neighbor has a different color. Correspondingly, the power constraint $P_{total} \leq 5^{3+1} = 625$ is violated, as $P_{total} = 810$.



(b) Assignment of four colors to the graph where each neighbor has a different color. Correspondingly, the power constraint $P_{total} \leq 5^{4+1} = 3125$ is satisfied, as $P_{total} = 810$.

Figure C.2 Here we show two different color assignments to the undirected graph on the left, and corresponding size assignments and power for the mapped directed graph on the right. s is the size and output slew of a gate. P is the gate power. The index of the power j , where power is 5^j , is the gate size if no fanins are the same size (color). The invalid assignment of three colors on the left in (a) where two neighbors have the same color corresponds to a power constraint violation in the directed graph. In (b) the assignment of four colors on the left ensures neighbors have different color, and the corresponding maximum gate size for the minimum power assignment on the right is four.

Appendix D Geometric Programming

D.1 Additional details for the data sampling

Rise slew time s_{rise} was measured from 10% $V_{dd,gate}$ to 90% $V_{dd,gate}$. Fall slew time s_{fall} is measured from 90% $V_{dd,gate}$ to 10% $V_{dd,gate}$. For equivalent slew times when $V_{dd,in}$ exceeds $V_{dd,gate}$, the slew rate dV/dt of the input is faster: a rising input takes a time interval of $s_{in,rise} V_{dd,gate}/V_{dd,in}$ from 10% $V_{dd,gate}$ to 90% $V_{dd,gate}$. As is typically done in SPICE characterization, input signals were approximated as voltage ramps. The equivalent time for the input to ramp linearly from 0V to Vdd or Vdd to 0V is the 10%-90% signal slew time $\times 1.25$, which is the characterized slew time.

For delay analysis, the input capacitances determined from power analysis were used. In retrospect this is inaccurate, as the average input capacitance seen during the switching interval is about 2/3 of this value [138]. To accurately characterize input capacitance for static timing analysis, delay characterization is generally performed with the gate driving an inverter of given input capacitance, as determined from the (dis)charging current – i.e. switching power. The delay and slew lookup tables in the standard cell library are then calibrated versus the corresponding input capacitance for the inverter. This was not done for our SPICE characterizations due to the additional computation time, which was already quite long, and due to the complexity of accurately calibrating the equivalent input capacitance for delay analysis versus the input capacitance for power analysis.

For delay (including slew) and leakage analysis, only the worst timing transitions and worst leakage states were considered. The worst output high and worst output low leakage states for a gate are the dominant leakage states as described in [183], namely where only the transistors nearest to the supply rail or ground rail are off. It is straightforward to include more posynomials to model additional leakage states and multiple timing arcs, but this will increase computation time.

From discussion with Bhushan Gupta, a reasonable upper limit for the supply voltage was 10% above the nominal voltage of 1.2V for STMicroelectronics' 0.13um technology. Ranges for other variables were chosen based on discussion with Dennis Sylvester. The SPICE zero bias threshold voltage parameter $vth0$ was adjusted to provide a range of threshold voltages from $\pm 0.13V$ of the nominal threshold voltage. Adjusting $vth0$ represents the change in the threshold voltage of a process to first order. The SPICE characterization ranges for parameters were listed in Table 6.2, other than the PMOS and NMOS width ranges which are listed in Table D.1. Possible noise issues with skewed drive strengths from varying NMOS and PMOS widths and varying the NMOS and PMOS threshold voltages were ignored. However, this is of practical concern for real circuits, which generally motivates using pull-up and pull-down drive strengths that are not too skewed versus each other.

In typical standard cell libraries, standard cell gates have PMOS to NMOS (P:N) width ratio of about 2:1 to balance rise and fall delays, as NMOS transistors have about double the drain current for the same width. However, skewed P:N ratios can be advantageous. To reduce power with smaller PMOS transistor capacitances, a ratio of as low as 1.5:1 may be better. Moreover, sometimes the rise and fall drive strengths needed are different. For example if a rising input signal to an inverter arrives much later than the falling input signal, then it is important that the inverter have a fast fall transition, but the inverter's rising output transition can be slower.

For SPICE characterization, the PMOS width was determined versus the NMOS width, varying width ratios from 4:1 to 3:4 of the equivalent inverter PMOS to NMOS ratio (2:1), as listed in the second and third columns of Table D.1. The maximum NMOS widths were set too low for the NOR gates – they should have been at least 2.6um. Consequently for NOR gates in the geometric program formulation, using the posynomial models beyond the NMOS width characterization range in Table D.1 is an extrapolation and will be less accurate.

Table D.1 This table lists the SPICE characterization range for the NMOS and PMOS widths. Maximum NMOS widths for the NOR gates were too low (shown in red) – they should have been at least 2.6um.

Logic gate	Minimum PMOS to NMOS ratio	Maximum PMOS to NMOS ratio	Minimum NMOS Width (um)	Maximum NMOS Width (um)	Minimum PMOS Width (um)	Maximum PMOS Width (um)
Inverter	3 : 4	4 : 1	0.26	6.93	0.26	20.80
NAND2	3 : 8	4 : 2	0.26	6.93	0.26	13.87
NAND3	3 : 12	4 : 3	0.26	10.40	0.26	13.87
NAND4	3 : 16	4 : 4	0.26	13.87	0.26	13.87
NOR2	6 : 4	8 : 1	0.26	1.733	0.39	13.87
NOR3	9 : 4	12 : 1	0.26	1.156	0.59	13.87
NOR4	12 : 4	16 : 1	0.26	0.867	0.78	13.87

The SPICE characterization temperature was 100°C and the worst case NMOS and worst case PMOS process corners were used. These conditions correspond to a worst case delay corner, except that nominal supply voltage was used. The higher temperature increases leakage. The worst case NMOS and PMOS corners correspond to higher V_{th} and lower leakage.

Perl scripts were used to generate the SPICE files. An example SPICE file is shown in Figure D.1. Analysis was performed in Hspice [14] with STMicroelectronics' 0.13um technology files.

The ISCAS'85 circuit format features XOR and XNOR gates, inverters, and some NAND, NOR, OR and AND gates with a large number (i.e. more than four) of inputs. A short Perl script was written by Brandon Thompson to map the circuits from the ISCAS'85 format to our library of inverter, NAND2, NAND3, NAND4, NOR2, NOR3, and NOR4. I corrected some errors in this script, and made changes to reduce the number of levels of the resulting logic to try and reduce the critical path delay of mapped circuits. A shorter critical path delay provides more timing slack for power reduction. For example, an XNOR2 gate is mapped to four NOR2 gates, comprising 3 levels of logic.

Given the large number of SPICE data points to fit, it may be surprising that we can do so reasonably accurately. The next subsection examines delay data for an inverter over a range of

supply voltages and input slews, illustrating a variable transformation to simplify modeling of the data.

```

.option acct method=gear brief
.lib "spice.lib"
.global vdd 0
.temp 100

Vdd vdd 0 0.600
xnor4 1 2 3 4 5 nor4
cload 5 0 3.0f

.subckt nor4 in1 in2 in3 in4 out
xp4 out in4 xtr4 vdd PMOS_MODEL w=0.78u l=0.18u
xp3 xtr4 in3 xtr3 vdd PMOS_MODEL w=0.78u l=0.18u
xp2 xtr3 in2 xtr2 vdd PMOS_MODEL w=0.78u l=0.18u
xp1 xtr2 in1 vdd vdd PMOS_MODEL w=0.78u l=0.18u
xn4 out in4 GND GND NMOS_MODEL w=0.26u l=0.18u
xn3 out in3 GND GND NMOS_MODEL w=0.26u l=0.18u
xn2 out in2 GND GND NMOS_MODEL w=0.26u l=0.18u
xn1 out in1 GND GND NMOS_MODEL w=0.26u l=0.18u
.ends nor4

*specify 1ps time steps for analysis, and 12ns length for analysis
.tran 1p 12n

*specify input ramps to get worst case rise and fall times
Vin1 1 0 PULSE 1.300 0 0n 0.020n 0.020n 2n 4n
Vin2 2 0 PULSE 1.300 0 0n 0.020n 0.020n 4n 8n
Vin3 3 0 PULSE 1.300 0 0n 0.020n 0.020n 4n 8n
Vin4 4 0 PULSE 1.300 0 0n 0.020n 0.020n 4n 8n

*measure fall and rise times (i.e. slew from 10% to 90%)
.measure tran fall_slew trig v(5) val=0.54 fall=1 TD=10ns targ v(5) val=0.06 fall=1
.measure tran rise_slew trig v(5) val=0.06 rise=1 TD=8ns targ v(5) val=0.54 rise=1
*measure 0->1 90%-10% delay and 1->0 10%-90% delay
.measure tran fall_delay trig v(1) val=0.13 rise=1 TD=10ns targ v(5) val=0.54 fall=1
.measure tran rise_delay trig v(1) val=1.17 fall=1 TD=8ns targ v(5) val=0.06 rise=1

* Measure the input capacitance from Qin=CV, measure Qin over full input transition
.measure tran cin_rise INTEG i(Vin1) FROM=8ns TO=10ns
.measure tran cin_fall INTEG i(Vin1) FROM=10ns TO=12ns

* Measure the load energy: should be cload*Vdd^2
.measure tran cload_rise_current INTEG i(cload) FROM=8ns TO=10ns
.measure tran cload_fall_current INTEG i(cload) FROM=10ns TO=12ns

* Measure the charged drawn from VDD, should give total power VDD*Q
.measure tran vdd_rise_current INTEG i(Vdd) FROM=8ns TO=10ns
.measure tran vdd_fall_current INTEG i(Vdd) FROM=10ns TO=12ns

* Measure the leakage charge going to GND, should give total leakage energy VDD*Q
* m1 is the name of the transistor in the NMOS_MODEL subcircuit
.measure tran gnd_leakage_rise_current INTEG i3(xnor4.xn1.m1) FROM=9.6ns TO=10ns
.measure tran gnd_leakage_fall_current INTEG i3(xnor4.xn1.m1) FROM=11.6ns TO=12ns
.end

```

Figure D.1 Example SPICE file for a NOR4 gate with 1.3V input voltage swing with slew of 0.02ns, 0.6V supply voltage, and a load of 3fF.

D.2 Simplifying inverter delay fits vs. input slew and input voltage

Before we examine posynomial fits, we show that it is possible to condense the data and reduce the difficulty of fitting it. While the particular modeling approach used in this section was not used with the posynomial models, it does illustrate how the posynomial models may provide fairly accurate fits to the data. This was tried in the initial exploration into posynomial fits versus multiple supply voltages and multiple threshold voltages, where previous research had not considered allowing these as optimization variables or model parameters. Previous researchers had assumed that V_{dd} and V_{th} were fixed, and that transistor widths or gate sizes were the only optimization variables.

We shall examine a variable transformation for the input slew and delay versus input voltage swing which simplifies fitting. As mentioned above, for the same input slew time, $s_{in,rise}$ or $s_{in,fall}$, if the input voltage swing $V_{dd,in}$ is larger, the slew rate dV/dt is larger. In terms of $V_{dd,in}$ and $V_{dd,gate}$, the input slew time from 0V to $V_{dd,gate}$ or vice versa is

$$s'_{in,rise} = s_{in,rise} \frac{V_{dd,gate}}{V_{dd,in}} \quad (\text{D.1})$$

$$s'_{in,fall} = s_{in,fall} \frac{V_{dd,gate}}{V_{dd,in}} \quad (\text{D.2})$$

For a falling input, there is a delay from when the input reaches 90% of $V_{dd,in}$ (where 10%-90% delays are measured) to when the input reaches 90% of the supply voltage $V_{dd,gate}$ of the gate being driven. When $V_{dd,in}$ is equal to $V_{dd,gate}$ these times are the same. In particular, we may consider *subtracting* the *additional delay* to try a transformation to remove the $V_{dd,in}$ variable, as the gate's output does not start changing (in our 10%-90% delay calculation with $V_{dd,in} = V_{dd,gate}$) until after this. The additional delay for the falling input is

$$0.9s_{in,fall} \frac{(V_{dd,in} - V_{dd,gate})}{V_{dd,in}} \quad (D.3)$$

Similarly if $V_{dd,in} > V_{dd,gate}$, a rising input arrives at 10% $V_{dd,gate}$ before it reaches 10% $V_{dd,in}$. The delay measurement from when the input is at 10% of $V_{dd,in}$ to when the output is at 90% of $V_{dd,gate}$ will be less than if we measured from when the input is at 10% of $V_{dd,gate}$ to when the output is at 90% of $V_{dd,gate}$. We can add the delay that was not included from the input rising from 10% of $V_{dd,gate}$ to 10% of $V_{dd,in}$. That delay is

$$0.1s_{in,rise} \frac{(V_{dd,in} - V_{dd,gate})}{V_{dd,in}} \quad (D.4)$$

The variable transformations to delay are

$$d'_{rise} = d_{rise} + 0.9s_{in,fall} \frac{(V_{dd,in} - V_{dd,gate})}{V_{dd,in}} \quad (D.5)$$

$$d'_{fall} = d_{fall} + 0.1s_{in,rise} \frac{(V_{dd,in} - V_{dd,gate})}{V_{dd,in}} \quad (D.6)$$

Note that for this analysis we have assumed 10%-90% delays and a negative polarity gate. The same analysis can be performed for a variable transformation to remove $V_{dd,in}$ for 50%-50% delays et al.

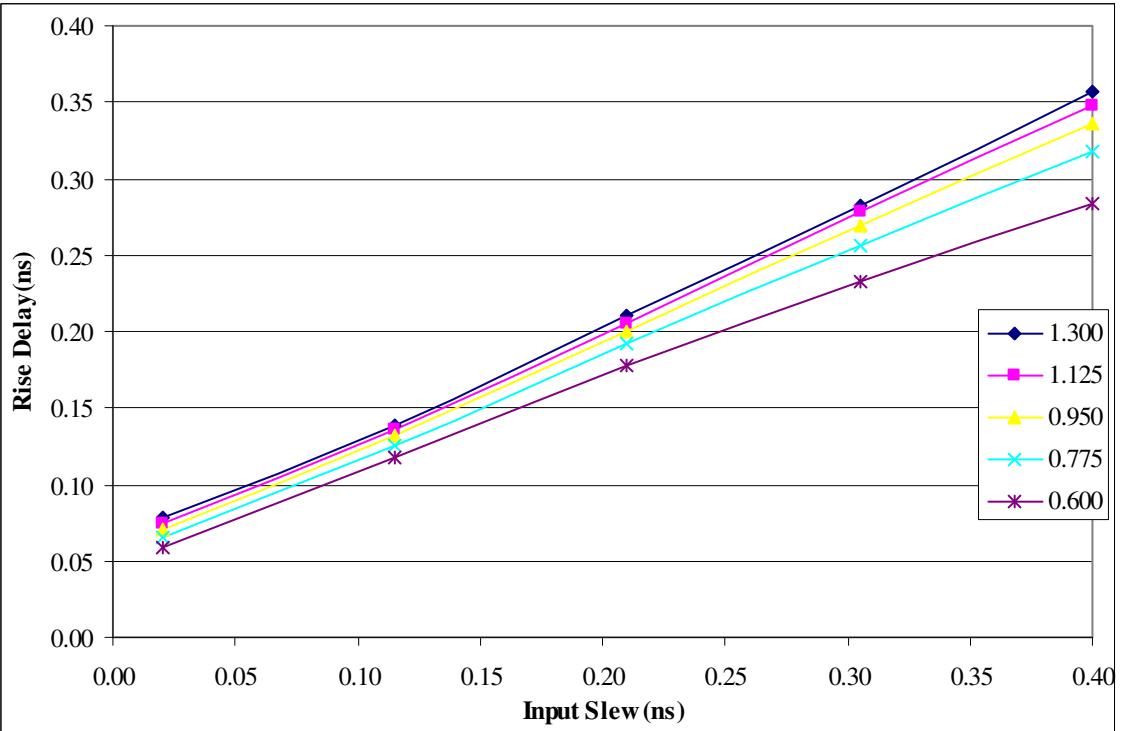


Figure D.2 Graph of measured inverter *rise* delay versus the input slew, with $V_{dd,gate}$ of 0.6V and varying $V_{dd,in}$ from 0.6V to 1.3V. Load capacitance, threshold voltages and transistor widths were constant.

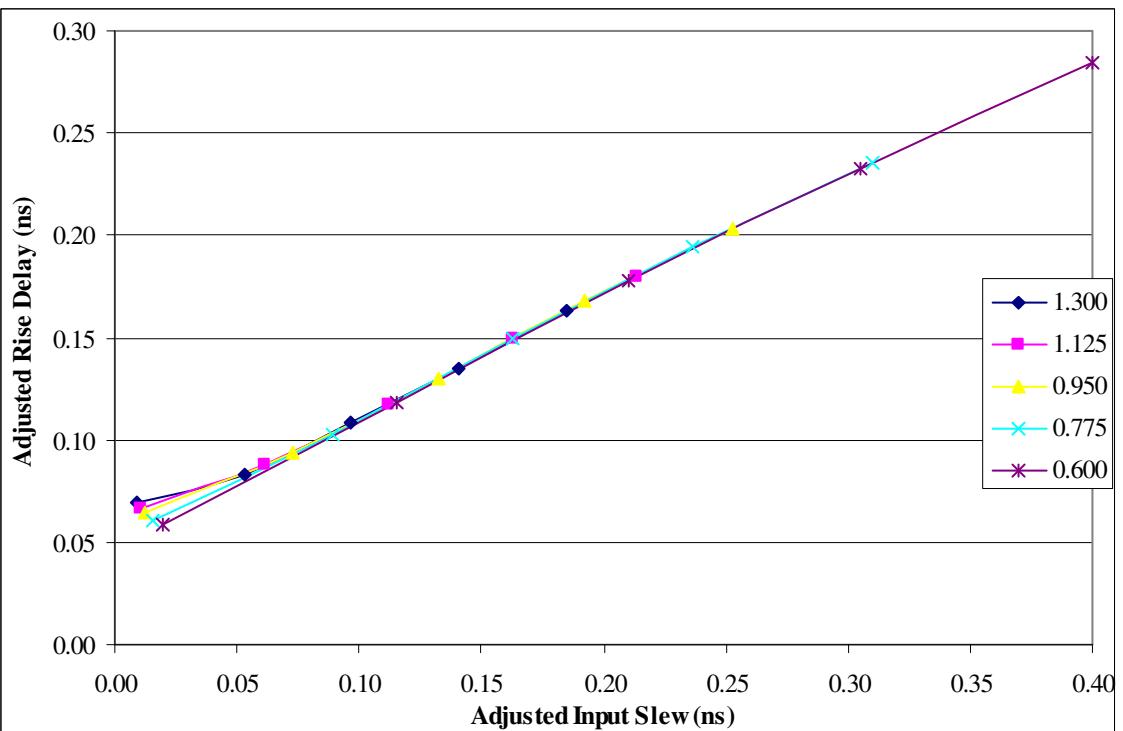


Figure D.3 After applying the variable transformations to adjust the input slew and output rise delay data from Figure D.2, the data lies almost exactly on the same curve and is easier to model.

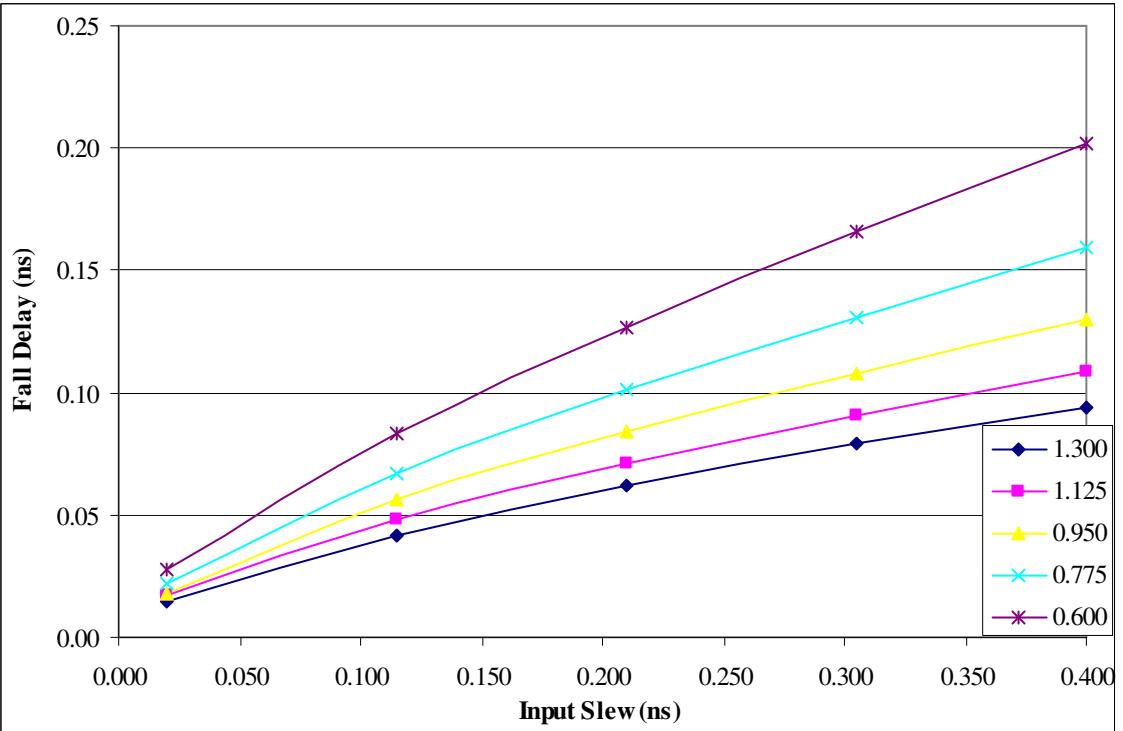


Figure D.4 Graph of measured inverter *fall* delay versus the input slew, with $V_{dd,gate}$ of 0.6V and varying $V_{dd,in}$ from 0.6V to 1.3V. Load capacitance, threshold voltages, and transistor widths were constant.

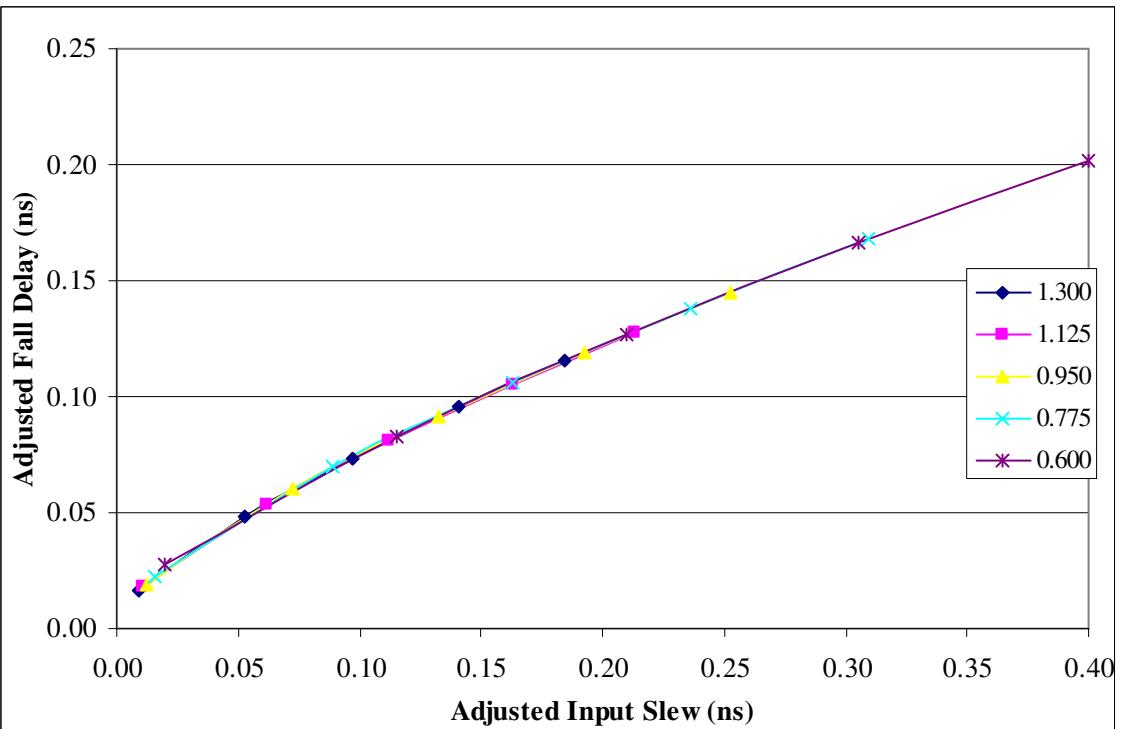


Figure D.5 After applying the variable transformations to adjust the input slew and output fall delay data from Figure D.4, the data lies almost exactly on the same curve and is easier to model.

Figure D.3 shows the inverter *rise* delay with after applying the variable transformations to the input slew and rise delay data in Figure D.2. Figure D.5 shows the inverter *fall* delay after applying the variable transformations to the input slew and fall delay data in Figure D.4. As can be seen from Figure D.3 and Figure D.5, the variable transformation has shifted the five curves that correspond to five different $V_{dd,in}$ values to lying on almost exactly the same curve, effectively removing the $V_{dd,in}$ variable.

Note the disparity that occurs in Figure D.3 for the smallest rise delay values. This disparity makes it more difficult to model small values accurately. This sort of problem also occurs when fitting standard cell library data with small values [138].

This subsection illustrated how the large data sets of 60,000 to 160,000 points may be modeled with relatively few fitting coefficients. As the next subsection details, we used posynomials with up to 27 coefficients to fit the data, which is quite small compared to the number of data points being fit.

D.3 Posynomial fits

To fit the data, posnomials summing three monomials were used. Adding more monomials did not generally improve the fit accuracy. The posynomial expressions for a negative polarity gate k are

$$d_{rise,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.7})$$

$$d_{fall,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.8})$$

$$S_{out,rise,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.9})$$

$$S_{out,fall,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.10})$$

$$C_{in,fall,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.11})$$

$$C_{in,rise,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.12})$$

$$P_{leak,high,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.13})$$

$$P_{leak,low,k} = \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \quad (\text{D.14})$$

$$\begin{aligned} E_{sw,k} = & \frac{1}{2} \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,rise,k}^{\alpha_{i2}} C_{load,fall,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \\ & + \frac{1}{2} \sum_{i=1}^3 c_i V_{dd,in,k}^{\alpha_{i1}} S_{in,fall,k}^{\alpha_{i2}} C_{load,rise,k}^{\alpha_{i3}} V_{dd,gate,k}^{\alpha_{i4}} W_{n,k}^{\alpha_{i5}} W_{p,k}^{\alpha_{i6}} (e^{V_{thn,k}})^{\alpha_{i7}} (e^{V_{thp,k}})^{\alpha_{i8}} \end{aligned} \quad (4.15)$$

where α_{ij} and c_i are fitting constants for the posynomial expression for each type of logic gate; $C_{load,fall}$ and $C_{load,rise}$ are respectively the load capacitance for a falling or a rising output; and $s_{in,fall}$ and $s_{in,rise}$ are respectively the maximum fanin fall or rise slew.

D.3.1 Posynomial fits with a reduced number of parameters

The runtime for geometric programming is quite slow, so we tried to reduce the number of parameters in the posynomials. In particular, we know that some characteristics should be independent of some parameters. For example, gate leakage should be independent of load capacitance and input slew.

The starting point for the reduction is the posynomial fit with all the parameters. For a variable x_j , we know the range of the parameter, $\min\{x_j\}$ to $\max\{x_j\}$, and the posynomial indices α_{ij} . From this, the maximum change a variable x_j causes in the value of a monomial term is given by

$$\max_i \left\{ \left(\frac{\min\{x_j\}}{\max\{x_j\}} \right)^{\alpha_{ij}}, \left(\frac{\max\{x_j\}}{\min\{x_j\}} \right)^{\alpha_{ij}} \right\} \quad (4.16)$$

Then the parameter having least effect on the posynomial can be determined. The posynomial is then fit to the data, omitting the parameter with the least effect on the posynomial's value. If the posynomial fit with a reduced number of parameters is within 1% of the accuracy using the full number of parameters, we accept this reduction, and try reducing the posynomial by an additional parameter. The MATLAB runtime to reduce the number of parameters is about two days for all the gates.

The accuracy of the posynomial fits with a reduced number of parameters is shown in Table D.2. The relative root mean square error is at most 1% worse than the accuracy for the posynomials with a full number of parameters listed in Table 6.3, except for leakage through the PMOS transistors when the pull-up network is off ($P_{leak,low}$). In some cases the accuracy improved by up

to 0.9%, which is not unexpected as fitting of the posynomials is a nonconvex problem and thus the solutions are not guaranteed to be globally optimal. Larger error was allowed when $V_{in} > V_{dd,gate}$ for $P_{leak,low}$, as the leakage is a negligible portion of the total power when the PMOS transistors are reverse biased.

The number of parameters in the reduced posynomials of sufficient accuracy is shown in Table D.3. As expected, input slew s_{in} and load capacitance C_{load} have minimal impact on leakage. The leakage through the NMOS transistors in the pull-down network when the output is high is primarily determined by the NMOS threshold voltage V_{thn} , the NMOS gate width W_n , and the gate supply voltage $V_{dd,gate}$. The leakage through the PMOS transistors in the pull-up network when the output is low is primarily determined by the PMOS threshold voltage V_{thp} , the input voltage $V_{dd,in}$, the PMOS gate width W_p , and the gate supply voltage $V_{dd,gate}$. Input capacitance is independent of input slew and load capacitance, except for NAND2 where C_{load} had some influence on the measured $C_{in,fall}$ values. Input capacitance for inverter, NOR2, NOR3 and NOR4 gates was independent of V_{thn} ; whereas the small increase in NMOS input capacitance with reduced V_{thn} was more significant for the NAND gates where NMOS widths are larger to maintain drive strength as more transistors are in series. NMOS threshold voltage generally has only a small impact on rise delay and rise slew, just as PMOS threshold voltage has only a small impact on fall delay and fall slew.

In total, the number of parameters appearing in the posynomials was reduced by 21%. Reducing the number of parameters in the posynomials increases the sparseness of the matrix representing the geometric program, but does not reduce the total number of variables or constraints in the geometric program. As such, the reduction in computation time that may be achieved is roughly linear in the reduction in the number of parameters. The reduction in geometric programming runtime by using posynomials with a reduced number of parameters was 21% for c17 and 26% for c432nr. Comparing to using posynomials with the full number of parameters, the accuracy of

the delay, the total dynamic power and the leakage power using posynomials with fewer parameters was within 1% of the results using the full number of terms for c432nr. Thus reducing the number of parameters gives a runtime improvement with minimal loss in accuracy.

Another issue is whether the posynomials underestimate the power and delay, and whether it is possible to have accurate posynomial models while ensuring the results are conservative.

Table D.2 Relative root mean square error of posynomial fits with a reduced number of parameters for the SPICE data. When output low leakage is significant, $V_{dd,in}=V_{dd,gate}$, and the accuracy is within 10%.

	Symbol	Inverter	NAND2	NAND3	NAND4	NOR2	NOR3	NOR4
Rise delay	d_{rise}	8.2%	6.9%	6.0%	4.8%	5.4%	3.8%	3.7%
Fall delay	d_{fall}	7.1%	6.1%	5.3%	4.8%	5.7%	5.9%	4.8%
Rise slew	$s_{out,rise}$	7.0%	7.0%	5.6%	4.7%	7.1%	4.3%	2.6%
Fall slew	$s_{out,fall}$	6.4%	7.6%	7.5%	5.8%	5.4%	5.4%	4.5%
Input capacitance C_{in} for rising input	$C_{in,rise}$	2.1%	1.6%	1.3%	1.1%	2.0%	2.0%	1.9%
Input capacitance C_{in} for falling input	$C_{in,fall}$	2.1%	1.5%	1.3%	1.2%	1.7%	1.7%	1.5%
Dynamic switching energy and short circuit power	E_{sw}	5.5%	5.2%	5.6%	5.3%	3.1%	2.5%	2.1%
Leakage when pull-down network is off	$P_{leak,high}$	5.4%	10.1%	10.3%	8.2%	5.9%	5.5%	5.4%
Leakage when pull-up network is off	$P_{leak,low}$	33.9%	45.1%	36.7%	30.3%	6.4%	3.7%	2.1%

Table D.3 This table lists the number of parameters that could be omitted in posynomials while achieving within 1% of the relative RMS accuracy when using the full number of parameters (which is eight). It was not possible to reduce the number of parameters for the dynamic energy posynomial models without reducing accuracy by more than 1%. For cases, where it was possible to reduce the number of parameters, the variables that could be omitted are listed in order of omission. For example for posynomial fits for $s_{out,rise}$, V_{thn} could be omitted for inverter, NAND2, NOR2, NOR3, and NOR4; and in the case of NOR2, W_n could also be omitted. For $P_{leak,low}$, the parameters omitted for NAND2 differed from the order of omitted variables for other gates: V_{thn} and s_{in} were omitted in the posynomial fit of $P_{leak,low}$ for NAND2.

	Symbol	Inverter	NAND2	NAND3	NAND4	NOR2	NOR3	NOR4	Omitted variables (in order of omission)
Rise delay	d_{rise}	1	1	1	1	1	0	0	V_{thn}
Fall delay	d_{fall}	1	1	1	1	1	1	1	V_{thp}
Rise slew	$s_{out,rise}$	1	1	0	0	2	1	1	V_{thn}, W_n
Fall slew	$s_{out,fall}$	1	2	2	1	0	0	0	$V_{thp}, V_{dd,gate}$
Input capacitance C_{in} for rising input	$C_{in,rise}$	3	2	2	2	3	3	3	$C_{load}, s_{in}, V_{thn}$
Input capacitance C_{in} for falling input	$C_{in,fall}$	3	1	2	2	3	3	3	$s_{in}, C_{load}, V_{thn}$
Switching energy & short circuit power	E_{sw}	0	0	0	0	0	0	0	-
Leakage when pull-down network is off	$P_{leak,high}$	5	4	3	2	4	5	5	$s_{in}, V_{thp}, W_p, C_{load}, V_{dd,in}$
Leakage when pull-up network is off	$P_{leak,low}$	4	2	4	4	3	2	1	$C_{load}, V_{thn}, W_n, s_{in}$

D.3.2 Conservative posynomial fits

Designers typically wish to have conservative timing and power models. If the optimization tools report the design meets certain performance targets, then they want a guarantee that the actual fabricated chip will have high yield with these performance targets. To this end, it is preferable to use conservative models which do not underestimate the performance criteria. For all the characteristics of interest to us, this implies that the posynomial models must not underestimate the value, but we want to have as close a fit as possible to the data.

The MATLAB fmincon function was used to perform constrained conservative fitting of the posynomials to the data for the inverter. That is the posynomial fit may be constrained to not underestimate any data points. The initial starting point for conservative fitting was the reduced parameter posynomials with an additional monomial term added. The added monomial is initially a constant, which is $\max(y_{data} - y_{fit})$. Adding the constant term to the initial starting point ensures a feasible starting point, at the price of using four monomials in the fit. The RMS relative errors for the conservative posynomial fits of the inverter data are shown in Table D.4. Unfortunately, the relative root mean square errors are substantially higher than those in Table D.2. The fmincon iterations diverged for inverter leakage when the output was low, and failed to find a solution. We also tried using only three monomials, starting with the reduced posynomials multiplied by $\max(y_{data}/y_{fit})$ to ensure a conservative starting point, but the fits were even more inaccurate.

Runtime with fmincon in MATLAB to conservatively fit the inverter posynomials was about one week on an Athlon MP 2000+. This was very slow, because the optimization is nonlinear and nonconvex.

Due to the large error (overestimates) with the conservative posynomial fits and the long run times to fit the data, it doesn't make sense to use the conservative posynomial fits. However, this limits us to using posynomial fits that may underestimate the delay and power in some cases.

Table D.4 Relative root mean square error for conservative posynomial fits with a reduced number of parameters to the inverter SPICE data. Compare the fit error to the RMS errors for the (non-conservative) posynomial fits of inverter data in Table D.2. The fmincon iterations diverged for inverter leakage when the output was low, and failed to find a solution.

	Symbol	Conservative inverter fit
Rise delay	d_{rise}	21%
Fall delay	d_{fall}	23%
Rise slew	$s_{out,rise}$	17%
Fall slew	$s_{out,fall}$	21%
Input capacitance C_{in} for rising input	$C_{in,rise}$	7%
Input capacitance C_{in} for falling input	$C_{in,fall}$	6%
Dynamic switching energy and short circuit power	E_{sw}	17%
Leakage when pull-down network is off	$P_{leak,high}$	9%
Leakage when pull-up network is off	$P_{leak,low}$	-

D.4 Formulating constraints for the geometric program solver

The constraints in Section 6.4 were written so as to be easy to read. However for the geometric program solver MOSEK, the constraints must be in the form $f(\mathbf{x}) \leq 1$ where $f(\mathbf{x})$ is a posynomial.

The constraints may be rewritten in the following manner to be represented in this form:

$$p(\mathbf{x}) \leq r \rightarrow \frac{p(\mathbf{x})}{r} \leq 1 \quad (4.17)$$

$$r \leq m(\mathbf{x}) \leq s \rightarrow \frac{r}{m(\mathbf{x})} \leq 1, \frac{m(\mathbf{x})}{s} \leq 1 \quad (4.18)$$

$$m(\mathbf{x}) = \max_i \{ p_i(\mathbf{x}) \} \rightarrow \forall i, \frac{p_i(\mathbf{x})}{m(\mathbf{x})} \leq 1 \quad (4.19)$$

$$m(\mathbf{x}) = \min_i \{ n_i(\mathbf{x}) \} \rightarrow \forall i, \frac{m(\mathbf{x})}{n_i(\mathbf{x})} \leq 1 \quad (4.20)$$

where $p(\mathbf{x})$ and $p_i(\mathbf{x})$ are posynomials, $m(\mathbf{x})$ and $n_i(\mathbf{x})$ are monomials, r and s are positive constants, and \mathbf{x} is the vector of positive-valued variables. Note that a monomial is a posynomial, but a posynomial is not a monomial. In particular, dividing by a posynomial is not possible for

constraints in the geometric program, but you may divide by a monomial because the inverse of a monomial is a monomial (divide by the positive-valued coefficient and variable indices change sign). The product of posynomials is a posynomial.

The transformed min and max constraints are not strictly the same, as there is no upper bound for max and no lower bound for the minimum. However, these bounds are tight when they affect the geometric program solution, because exceeding these bounds makes the solution suboptimal for our min and max constraints: a larger slew or larger output delay (e.g. due to lower input voltage swing) increases the critical path delay. Similarly, as MOSEK did not accept equality constraints in the geometric program, we specify the load capacitance constraints in the form

$$C_{load,v} \geq C_{wire} + \sum_{j \in fanout(v)} C_{in,j} \quad (4.21)$$

and the output arrival time constraints are expressed in the form

$$t_{out,rise,v} \geq t_{in,fall,v} + d_{rise,v} \quad (4.22)$$

A larger load capacitance increases the switching power and increases the gate delay, so the bound will be tight.

D.5 Voltage discretization for c432 at a delay constraint of 2.12ns

The optimization run results for dual voltage discretization of ISCAS'85 benchmark c432 with redundancies removed (c432nr) are listed in Table D.5. The power consumption for steps in each discretization is graphed in Figure D.6, Figure D.7 and Figure D.8. Note that these values are not graphed sequentially; values are clustered around where bisection zeroes in, and the curves would look different if more points with a differing number of assignments were included. The discretized state space is clearly non-convex for NMOS threshold voltage in Figure D.7, which indicates that the discretization heuristic can get stuck in local minima. NMOS threshold voltage

discretization results in power about 1.2% worse than the continuous V_{thn} lower bound. Whereas the best dual Vdd and dual PMOS threshold voltage solutions are within 0.2% of the continuous lower bounds. We have seen cases for c17 where the solution found by the discretization heuristic is 4% worse than the optimal discrete dual voltage solution.

To find a good discrete dual voltage solution for c432nr at a delay constraint of 2.12ns took 33 geometric programming optimization runs, excluding repeat assignments that were not re-run. The number of runs for discretizing each variable is $O(\log_2 N)$, where N is the number of inputs and gates for Vdd discretization (258 total), or the number of gates for threshold voltage discretization (222 total). In practice, as many gates in the continuous solution are already at the maximum or minimum value, there are less discretization steps required – particularly in the case of Vdd.

Table D.5 Dual Vdd, dual Vthn, dual Vthp discretization on c432nr from the continuous solution at the top of the table, for a delay constraint of 2.12ns. Supply voltage Vdd is discretized, then NMOS threshold voltage Vthn, and then PMOS threshold voltage Vthp. For continuous runs and before a variable is discretized, there will be some gates with values in-between the maximum and minimum. Results were not re-run in cases where all the gates were assigned the same (noted “repeat”) as in a previous discretization step. Power for continuous runs is highlighted in yellow. The best discretization result in each section is highlighted in blue. The variable discretization is being performed on is highlighted in green. The power consumption for steps in each discretization is graphed in Figure D.6, Figure D.7 and Figure D.8.

Power (uW)	Vdd discretization				Vthn discretization				Vthp discretization			
	Max Vdd(V)	#	Min Vdd(V)	#	Max Vthn(V)	#	Min Vthn(V)	#	Max Vthp(V)	#	Min Vthp(V)	#
888.140	1.300	42	0.600	3	0.357	58	0.097	30	0.338	9	0.078	122
891.239	1.295	42	1.245	216	0.357	58	0.097	30	0.338	9	0.078	120
892.377	1.256	258		0	0.357	58	0.097	30	0.338	9	0.078	102
889.413	1.257	255	0.600	3	0.357	58	0.097	30	0.338	9	0.078	103
889.330	1.258	248	0.600	3	0.357	58	0.097	30	0.338	9	0.078	102
891.256	1.258	254	1.003	4	0.357	58	0.097	30	0.338	9	0.078	101
889.413	1.257	255	0.600	3	0.357	58	0.097	30	0.338	9	0.078	103
repeat												
889.413	1.257	255	0.600	3	0.357	58	0.097	30	0.338	9	0.078	103
repeat												
907.548	1.266	255	0.600	3	0.148	192	0.097	30	0.338	23	0.078	118
902.984	1.258	255	0.600	3	0.179	125	0.097	97	0.338	21	0.078	101
901.950	1.257	255	0.600	3	0.357	58	0.109	164	0.338	9	0.078	93
891.130	1.253	255	0.600	3	0.357	58	0.097	97	0.338	9	0.078	102
903.965	1.254	255	0.600	3	0.207	91	0.098	131	0.338	19	0.078	102
901.950	1.257	255	0.600	3	0.357	58	0.109	164	0.338	9	0.078	93
repeat												
894.421	1.250	255	0.600	3	0.357	58	0.109	131	0.338	9	0.078	104
904.886	1.259	255	0.600	3	0.294	75	0.106	147	0.338	9	0.078	98
901.950	1.257	255	0.600	3	0.357	58	0.109	164	0.338	9	0.078	93
repeat												
899.094	1.255	255	0.600	3	0.357	58	0.106	147	0.338	9	0.078	94
901.049	1.256	255	0.600	3	0.357	66	0.108	156	0.338	9	0.078	92
901.950	1.257	255	0.600	3	0.357	58	0.109	164	0.338	9	0.078	93
repeat												
899.094	1.255	255	0.600	3	0.357	58	0.106	147	0.338	9	0.078	94
902.975	1.259	255	0.600	3	0.347	71	0.107	151	0.338	9	0.078	96
901.501	1.256	255	0.600	3	0.357	62	0.108	160	0.338	9	0.078	93
899.828	1.256	255	0.600	3	0.357	58	0.107	151	0.338	9	0.078	94
900.514	1.257	255	0.600	3	0.357	68	0.108	154	0.338	9	0.078	93
901.275	1.256	255	0.600	3	0.357	64	0.108	158	0.338	9	0.078	92
repeat												
900.514	1.257	255	0.600	3	0.357	68	0.108	154	0.338	9	0.078	93
907.413	1.261	255	0.600	3	0.357	68	0.107	154	0.101	126	0.078	96
904.841	1.257	255	0.600	3	0.357	68	0.107	154	0.141	67	0.078	155
907.199	1.253	255	0.600	3	0.357	68	0.107	154	0.338	9	0.078	213
900.514	1.257	255	0.600	3	0.357	68	0.108	154	0.338	9	0.078	93
906.853	1.260	255	0.600	3	0.357	68	0.107	154	0.107	97	0.078	125
902.576	1.254	255	0.600	3	0.357	68	0.108	154	0.250	38	0.078	184
900.734	1.255	255	0.600	3	0.357	68	0.108	154	0.338	9	0.078	153
904.428	1.256	255	0.600	3	0.357	68	0.108	154	0.166	54	0.078	168
902.413	1.254	255	0.600	3	0.357	68	0.108	154	0.293	24	0.078	198
901.619	1.254	255	0.600	3	0.357	68	0.108	154	0.338	9	0.078	183
902.237	1.254	255	0.600	3	0.357	68	0.108	154	0.279	32	0.078	190
904.441	1.253	255	0.600	3	0.357	68	0.108	154	0.284	17	0.078	205

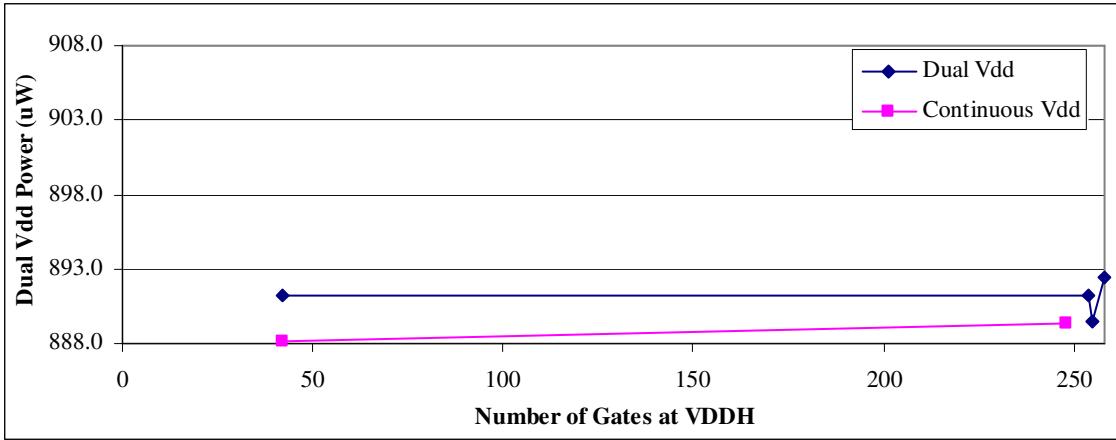


Figure D.6 Power consumption for the dual Vdd discretization steps in Table D.5.

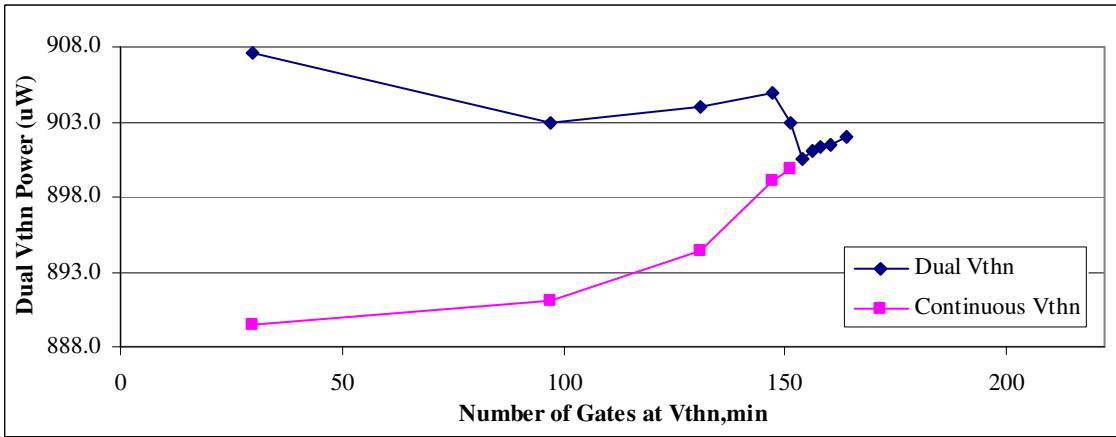


Figure D.7 Power consumption for the dual Vthn discretization steps in Table D.5, after Vdd discretization. Note that the power for dual Vthn solutions is noticeably non-convex versus the number of gates at $V_{thn,min}$.

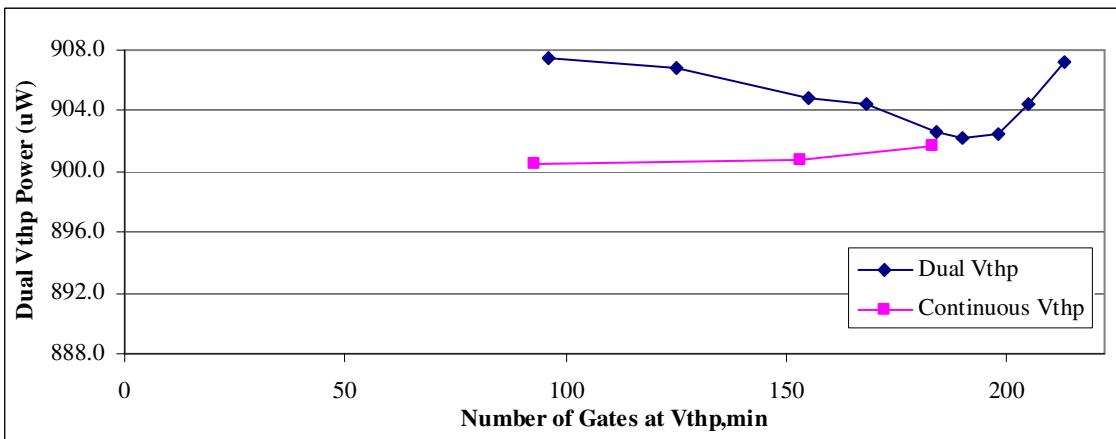


Figure D.8 Power consumption for the dual Vthp discretization steps in Table D.5, after Vdd and Vthn discretization.

D.6 Geometric programming result details

Benchmark c17 has high switching activity, 0.47 probability on average that a gate or input switches, and a small critical path delay, so the leakage power is small relative to the dynamic power. With Vdd of 1.3V and nominal threshold voltages, 0.227V for NMOS and 0.208V for PMOS, at the minimum delay achievable (0.195ns), the leakage is only 0.036% of the total power. Thus there is little trade-off between dynamic and leakage power – threshold voltages can be minimized without making leakage that large to give timing slack to reduce dynamic power. To make the scenario more interesting for c17 from an optimization perspective, we normalize the leakage to 1% of the total power at Vdd=1.3V/Vthn=0.227V/Vthp=0.208V at the minimum delay for these voltage values, by dividing dynamic power by 28.1.

The other benchmarks have quite high switching activity, 0.38 for c432nr for example, but the critical path delay is much larger and thus leakage power is more significant. For example for c432nr at Vdd=1.3V, Vthn=0.227V and Vthp=0.208V, the minimum delay achievable is 2.17ns and the leakage power is about 0.8% of total power at this point. Thus no normalization is needed for the larger benchmarks. Also note that leakage has been emphasized by using posynomial fits to SPICE data characterized at a temperature of 100°C, where the leakage is more than an order of magnitude larger than at room temperature.

Power was minimized over a range of delay constraints, from the minimum delay (T_{\min}) achievable with continuous voltages to three times this. As c17 is small with fast solver run times, data was collected over a hundred steps from the minimum delay. Whereas for other benchmarks with longer run times, only ten delay increments were used. As the most rapid changes occur near T_{\min} , more points were collected near there: for c17, the delay constraint on iteration i was $T_{\min} + 2i^2T_{\min}/10,000$; for the other benchmarks, the delay constraint on iteration i was $T_{\min} + 2i^2T_{\min}/100$. This provides a good spread of points, focusing more on results near T_{\min} .

D.6.1 c17

The supply voltage Vdd scales down steadily to reduce dynamic power as the delay constraint relaxes, as shown in Figure D.9. As the supply voltage is reduced, the maximum NMOS and PMOS threshold voltages also decrease to avoid too large a delay penalty – see Figure D.10 and Figure D.11. At large delay constraints where Vdd is near the minimum allowed (0.6V), the NMOS and PMOS threshold voltage increase to reduce leakage. The percentage of gates at high supply voltage, low NMOS threshold voltage, and low PMOS threshold voltage for the dual voltage solutions are shown in Figure D.12, Figure D.13 and Figure D.14 respectively.

The average PMOS to NMOS width ratio is shown in Figure D.15. The ratios have been normalized versus widths of an inverter of similar drive strength: dividing NMOS width by two for the NAND2 gates in c17. As was illustrated in Figure 6.4, for our 0.18um channel length the NMOS drain current is 2.9× that of the PMOS drain current for a gate when Vdd is 1.3V, and the NMOS drain current is 3.7× that of the PMOS drain current for a gate when Vdd is 0.6V. Consequently, for equal rise and fall times we would expect a width ratio in the range of about 3:1 at Vdd of 1.3V towards 4:1 at Vdd of 0.6V. Indeed, the average PMOS to NMOS width ratio in Figure D.15 does increase as the delay constraint is relaxed and Vdd is reduced. However, the average width ratios are below that needed for equal rise and fall times, because less capacitance in the circuit reduces the power and also reduces delay. Additionally, on average the NAND2 gates have a high output 65% of the time, so PMOS threshold voltages are lower as they have less effect on leakage power, and thus PMOS widths can be reduced.

As was illustrated in Figure 6.4, there is increased NMOS drive strength when $V_{dd,in} > V_{dd,gate}$, so optimized multiple supply voltage circuits will have a higher mean PMOS to NMOS width ratio than single supply voltage circuits because the NMOS width can be less. The two curves with the lowest average PMOS to NMOS width ratio in Figure D.15 have single Vdd. The two curves with the highest average PMOS to NMOS width ratio in Figure D.15 are for continuous voltages and

“2vdd, 2vth”. This ignores the average width ratio for the dual voltage result where fixed global voltage values jumps around between substantially different points in the state space. For example, going from fives gates at VDDH=1.3V to all gates at VDDL=0.6V from delay constraint 0.288ns to 0.292ns, while there is no change in the number of gates at VDDH and VDDL for the “2vdd, 2vth” solution with variable global voltage values decreasing only slightly from VDDH=0.77V and VDDL=0.68V.

The average width ratio for the dual voltage result with fixed global voltage values ($V_{dd}=1.3V \& 0.6V$, $V_{thn}=0.227V \& 0.097V$, $V_{thp}=0.208V \& 0.078V$) jumps higher at a couple of delay points to allow V_{dd} to be reduced, as can be seen comparing Figure D.12 and Figure D.15. However at a delay constraint of 0.292ns, it is possible for the widths and thus circuit capacitance to be substantially reduced which also enables reducing V_{dd} , though threshold voltages are also lower. Consequently, there is a jump in the percentage of leakage power for dual voltages with fixed global voltage values, shown in Figure D.16, as dynamic power drops 38% from 0.288ns to 0.292ns delay constraint. For the other solutions with variable voltage values, the percentage of leakage changes little until a large delay constraint of about 0.42ns where it is finally worthwhile to increase threshold voltage values.

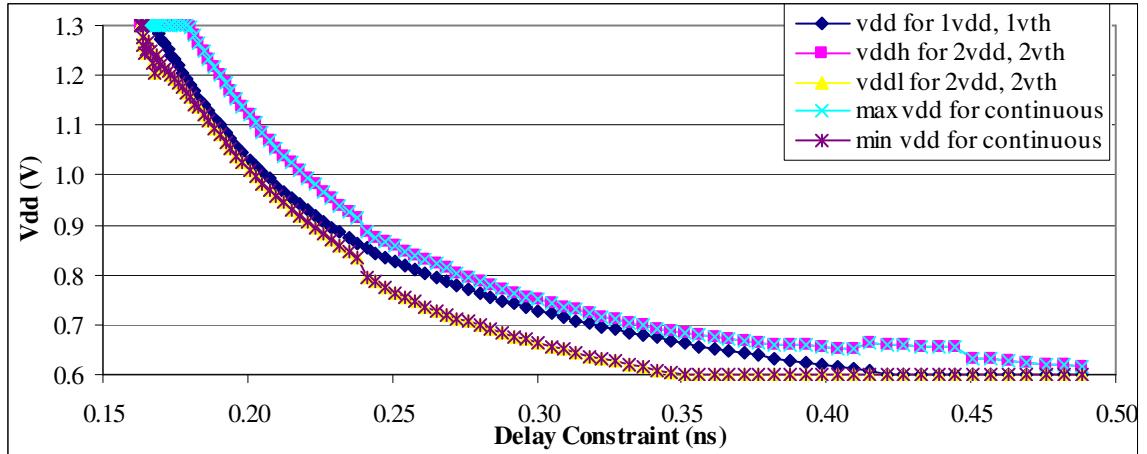


Figure D.9 Supply voltage versus delay constraint for c17.

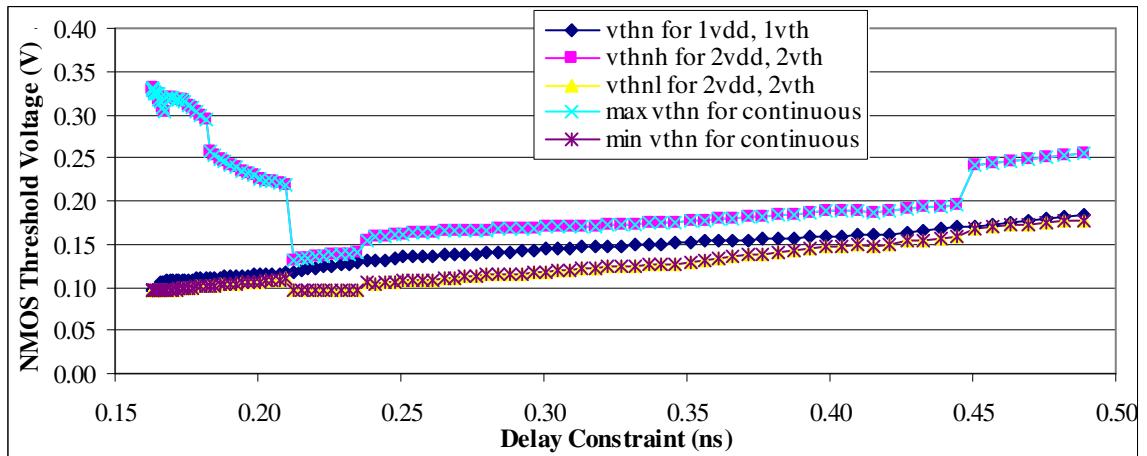


Figure D.10 NMOS threshold voltage versus delay constraint for c17.

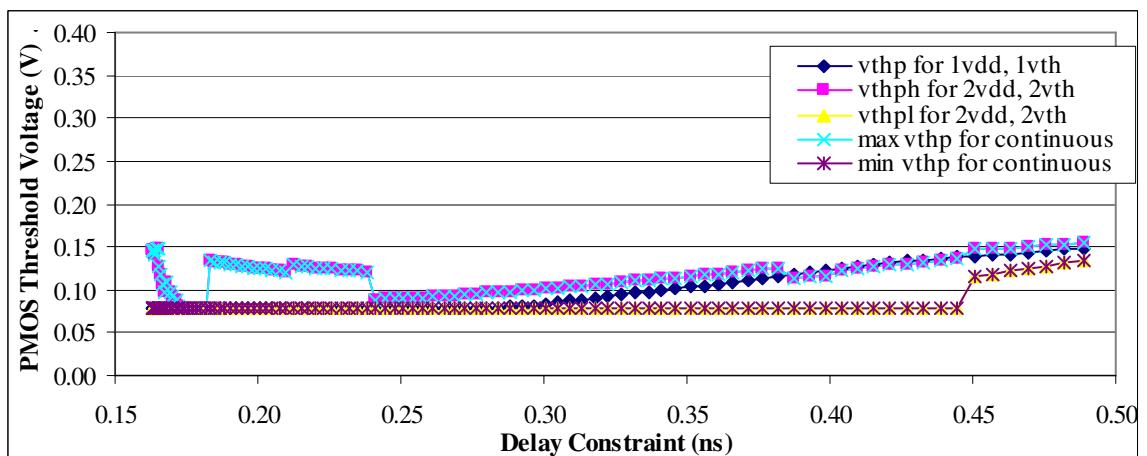


Figure D.11 PMOS threshold voltage versus delay constraint for c17.

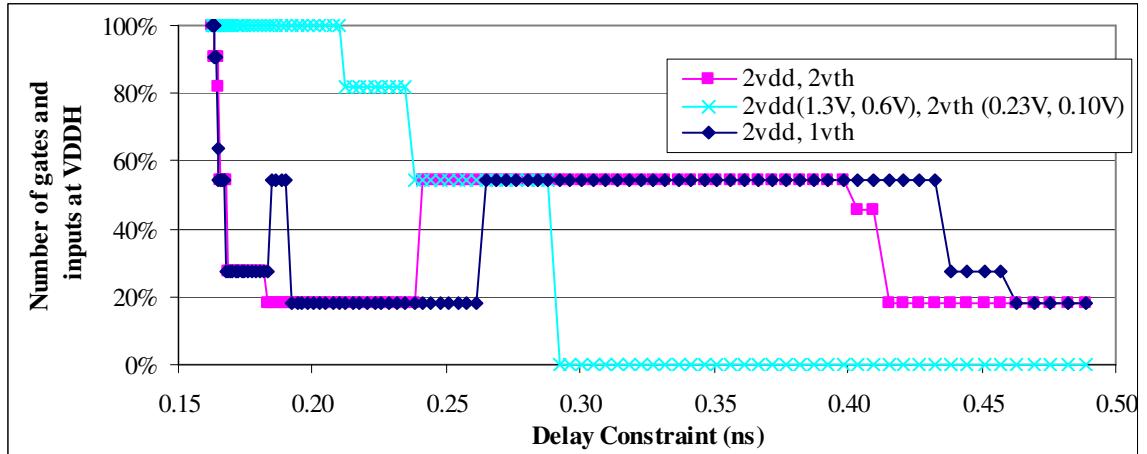


Figure D.12 Percentage of gates at the high supply voltage (VDDH) for the dual supply voltage c17 results.

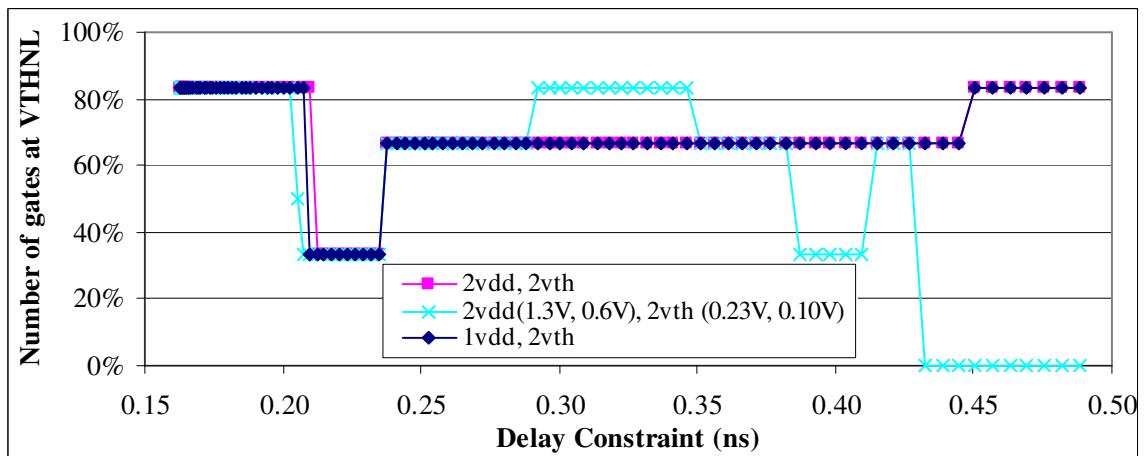


Figure D.13 Percentage of gates at the low NMOS threshold voltage (VTHNL) for the dual threshold voltage c17 results.

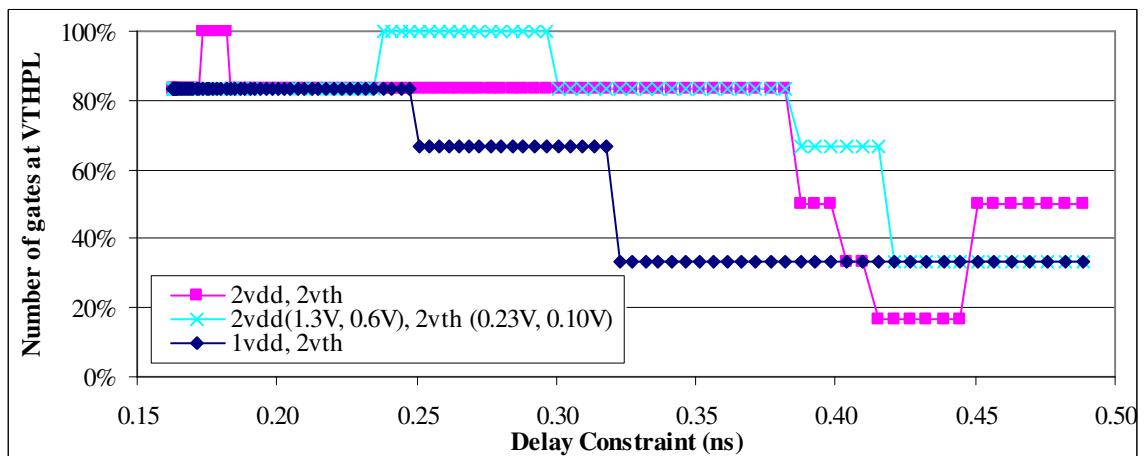


Figure D.14 Percentage of gates at the low PMOS threshold voltage (VTHPL) for the dual threshold voltage c17 results.

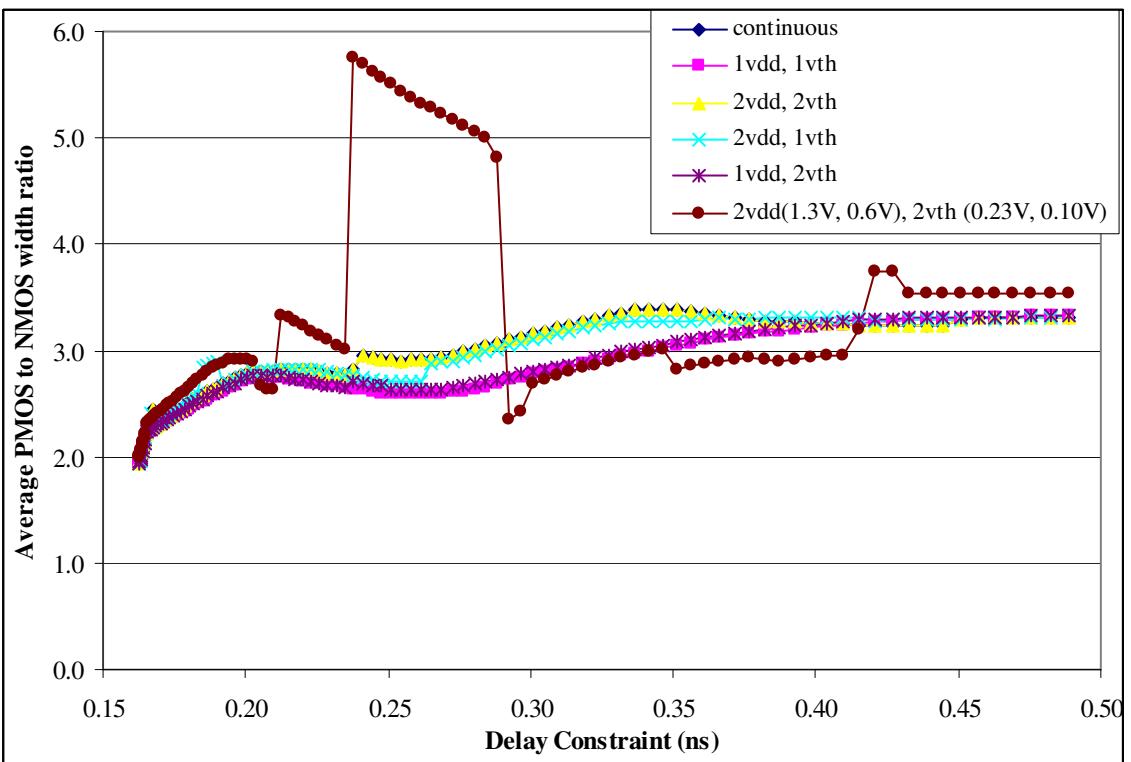


Figure D.15 Average PMOS width to NMOS width ratio for c17.

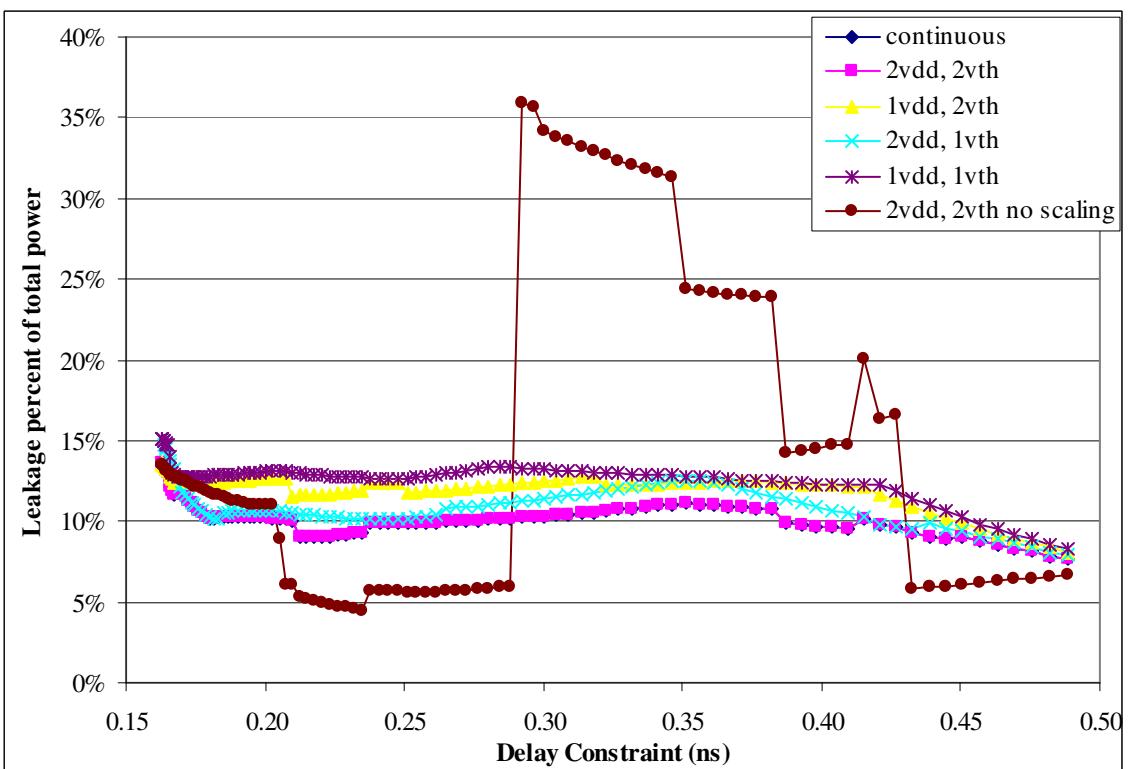


Figure D.16 Leakage as a portion of total power for c17.

D.6.2 c432nr

The maximum supply voltage scales down steadily to reduce dynamic power as the delay constraint relaxes, while the minimum supply voltage starts near 0.6V and quickly drops to that minimum Vdd as shown in Figure D.17. However, only three gates are at the minimum supply voltage in the dual voltage and continuous voltage solutions until a delay constraint of 4.09ns. As gates on critical paths already have minimum threshold voltage, there is no additional reduction in Vth as supply voltage is reduced. As the delay constraint is relaxed, the minimum NMOS threshold voltage increases to reduce leakage, while the PMOS threshold voltage only increases later at a more relaxed delay constraint, because leakage through the pull-down NMOS transistors is a larger component of the leakage power than leakage through the PMOS pull-up transistors. The minimum and maximum supply voltages, NMOS threshold voltages, and PMOS threshold voltages are shown in Figure D.17, Figure D.18 and Figure D.19 respectively. The number of gates at high supply voltage, low NMOS threshold voltage, and low PMOS threshold voltage for the dual voltage solutions are shown in Figure D.20, Figure D.21 and Figure D.22 respectively.

The average PMOS to NMOS width ratio is shown in Figure D.23. The ratios have been normalized versus widths in an inverter of similar drive strength: dividing NMOS width by 2, 3 and 4 for NAND2, NAND3 and NAND4 respectively; and dividing PMOS width by 2, 3 and 4 for NOR2, NOR3 and NOR4 respectively. The average P:N width ratio increases only slightly as the delay constraint is relaxed, varying from about 2.6:1 to 2.8:1. As discussed for the c17 results, the width ratio is less than that required for equal rise and fall times (about 2.9:1 when Vdd is 1.3V to 3.7:1 when Vdd is 0.6V for nominal Vth values). The lower PMOS widths reduce circuit capacitance, thus reducing power consumption and also reducing delay. On average the gates in c432nr have a high output 58% of the time, which also reduces the impact of PMOS threshold voltages on the total leakage power, because leakage is more often through the NMOS pull-down network in a gate. As very few gates change to VDDL until larger delay constraints, few gates

have $V_{dd,in} > V_{dd,gate}$, and the increased NMOS drive strength when $V_{dd,in} > V_{dd,gate}$ has little impact on the P:N width ratio.

As the delay constraint is relaxed, there is more timing slack to increase the NMOS threshold voltage, and the leakage power decreases faster than the decrease in dynamic power due to gate downsizing – see Figure D.24. However, as the delay constraint is further relaxed, dynamic power decreases more rapidly than leakage power. The increased delay constraint contributes to the reduction in dynamic power, as the larger clock period reduces the switching activity per unit time. For the dual voltage solution with fixed global voltage values, “2vth (0.23V, 0.10V)” on the graphs, leakage increases significantly from 3.09ns to 4.09ns, as threshold voltages drop to the minimum to enable all the gates to be assigned to VDDL=0.6V.

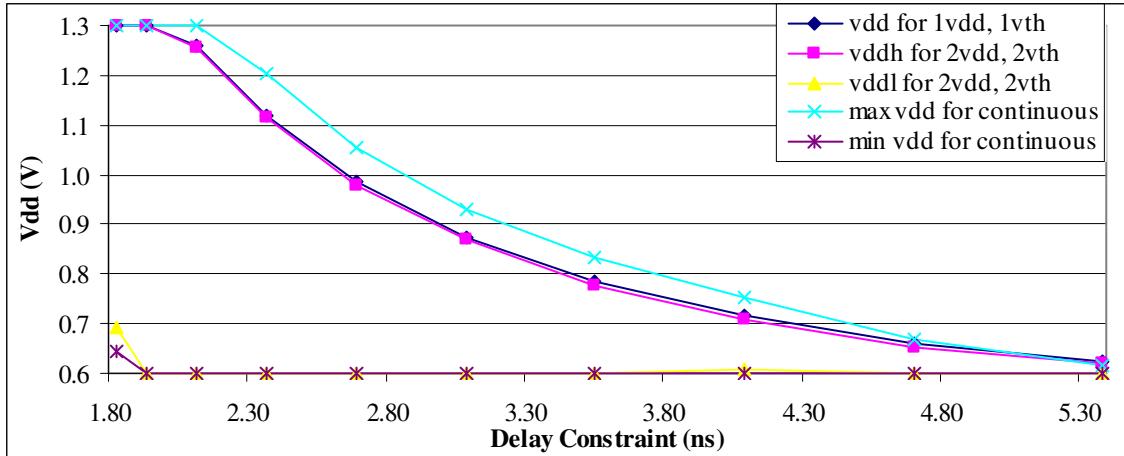


Figure D.17 Supply voltage versus delay constraint for c432nr.

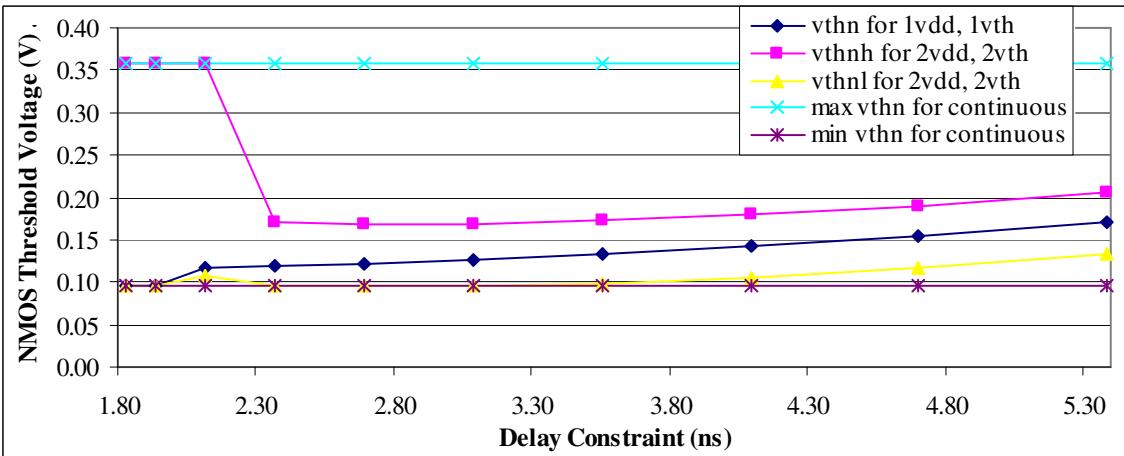


Figure D.18 NMOS threshold voltage versus delay constraint for c432nr.

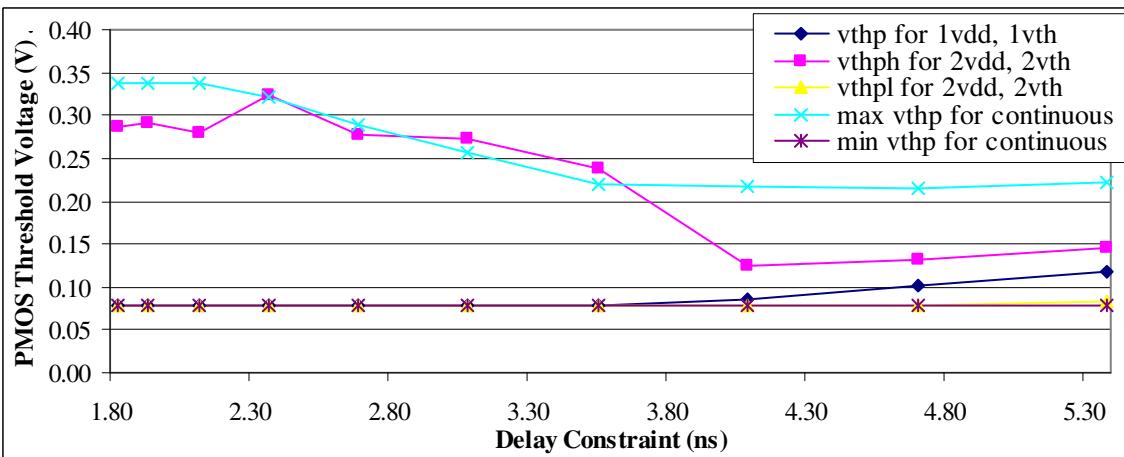


Figure D.19 PMOS threshold voltage versus delay constraint for c432nr.

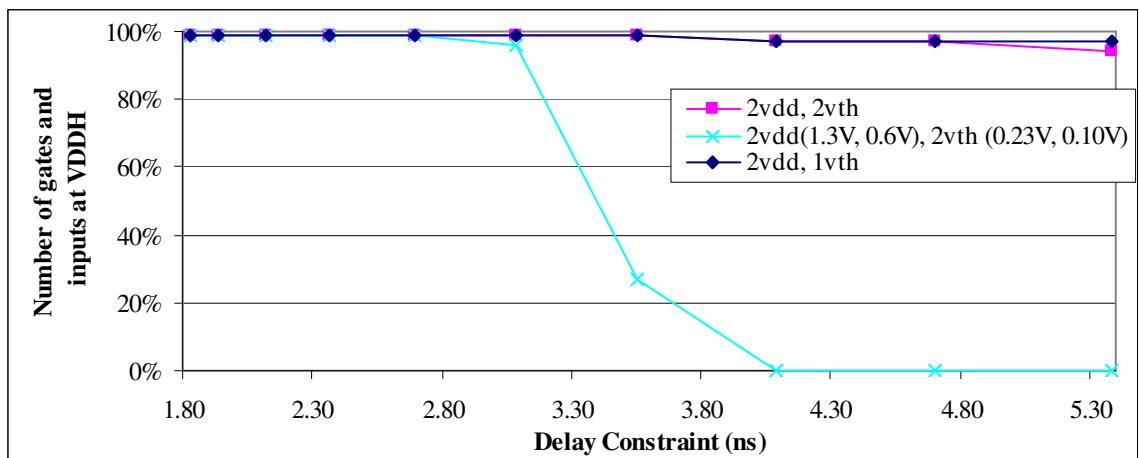


Figure D.20 Percentage of gates at the high supply voltage (VDDH) for the dual supply voltage c432nr results.

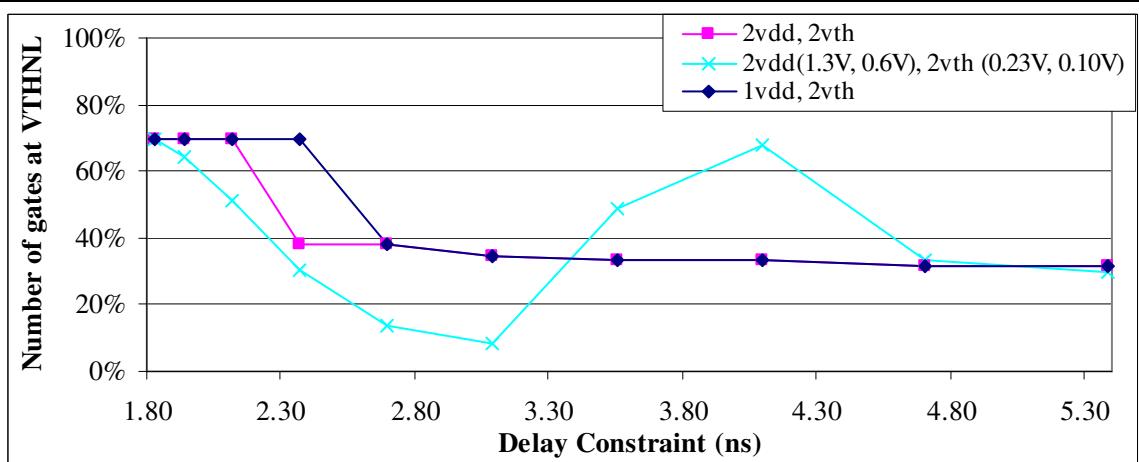


Figure D.21 Percentage of gates at the low NMOS threshold voltage (VTHNL) for the dual threshold voltage c432nr results.

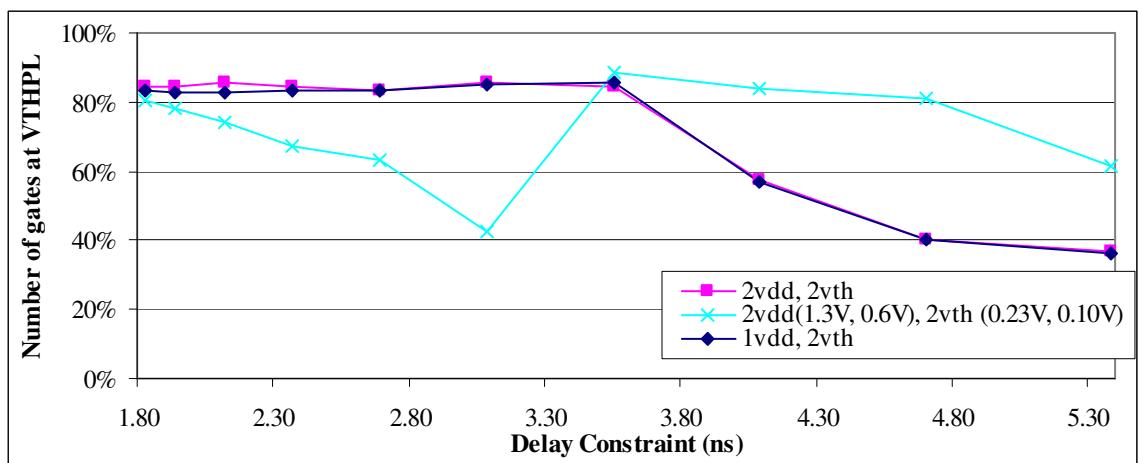


Figure D.22 Percentage of gates at the low PMOS threshold voltage (VTHPL) for the dual threshold voltage c432nr results.

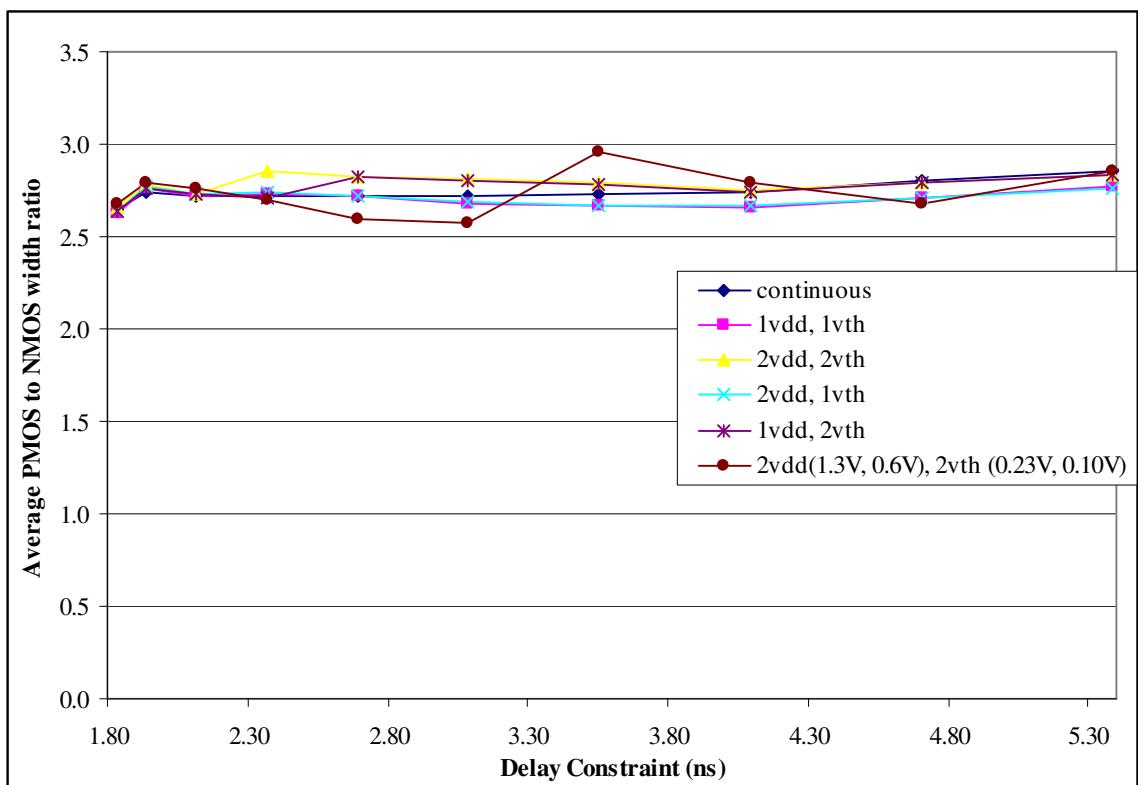


Figure D.23 Average PMOS width to NMOS width ratio for c432nr.

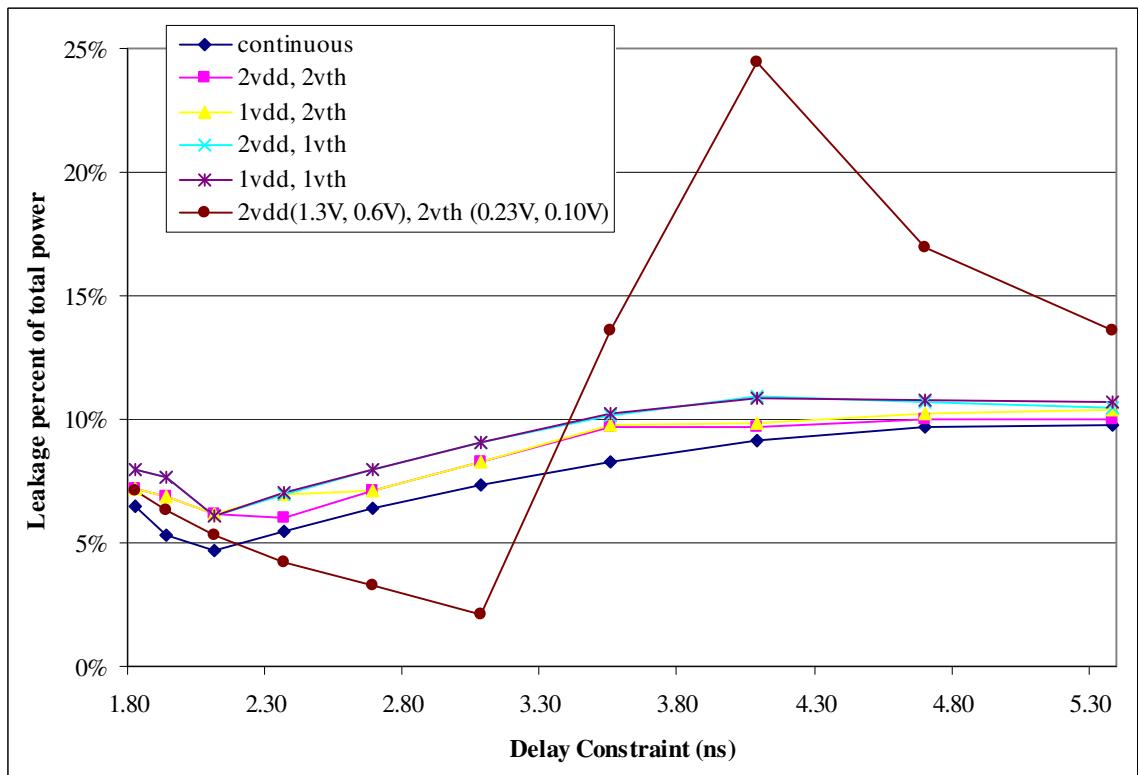


Figure D.24 Leakage as a portion of total power for c432nr.

D.6.3 Power versus delay curves for c17 and c432nr with fixed global voltage values

Fixing the global voltages values can be quite suboptimal in comparison to results achieved with voltage scaling. To meet the minimum delay achieved by the continuous solution, the high supply voltage is the maximum supply voltage of 1.3V and low V_{th} is the minimum threshold voltage, 0.10V for V_{thn} and 0.08V for V_{thp}. The low supply voltage of 0.6V and nominal threshold voltage, 0.23V for V_{thn} and 0.21V for V_{thp}, were chosen to minimize power at large delay constraints. There is a distinct power gap between the curves with fixed voltages and the continuous voltage lower bound, as shown in Figure D.25 and Figure D.26 for c17 and c432nr respectively.

D.6.4 Power versus delay curves for c17 and c432nr without wire loads

The wire loads reduce the steepness of the power versus delay curves, as the multiplicative benefits of gates being downsized (downsizing a gate reduces the load on fanins, which can in turn be downsized) are reduced. This can be seen comparing Figure D.27 and Figure D.28 for c17, and Figure D.29 and Figure D.30 for c432nr.

D.6.5 c880

The large gap of up to 22.2% between the single voltage and continuous voltage results for c880 can be explained by the number of gates that can be set to the minimum supply voltage in the continuous voltage solution. We will illustrate this by discussing the results at the 1.78ns delay constraint. For both single and continuous voltage results, leakage is only 5% to 6% of the total power, though the single voltage solution has the minimum PMOS threshold voltage of 0.078V and NMOS threshold voltage of 0.122V, which is quite near the minimum of 0.097V. In the single voltage solution, all the inputs and gates have 1.295V supply voltage to meet the tight delay constraint. Whereas in the continuous voltage solution, 47 inputs and 209 gates have V_{dd} of 1.300V, 6 inputs and 66 of the gates have the minimum V_{dd} of 0.600V, and the other 7 inputs

and 280 gates have intermediate Vdd values. That is, there is substantial timing slack on some paths in c880 that do not converge with timing critical paths, thus input drivers and gates on those paths can be set to a low supply voltage. This enables the continuous voltage solution to substantially reduce the dynamic power and hence total power. Whereas to meet the delay constraint on the timing critical paths, the single voltage solution must use a high supply voltage.

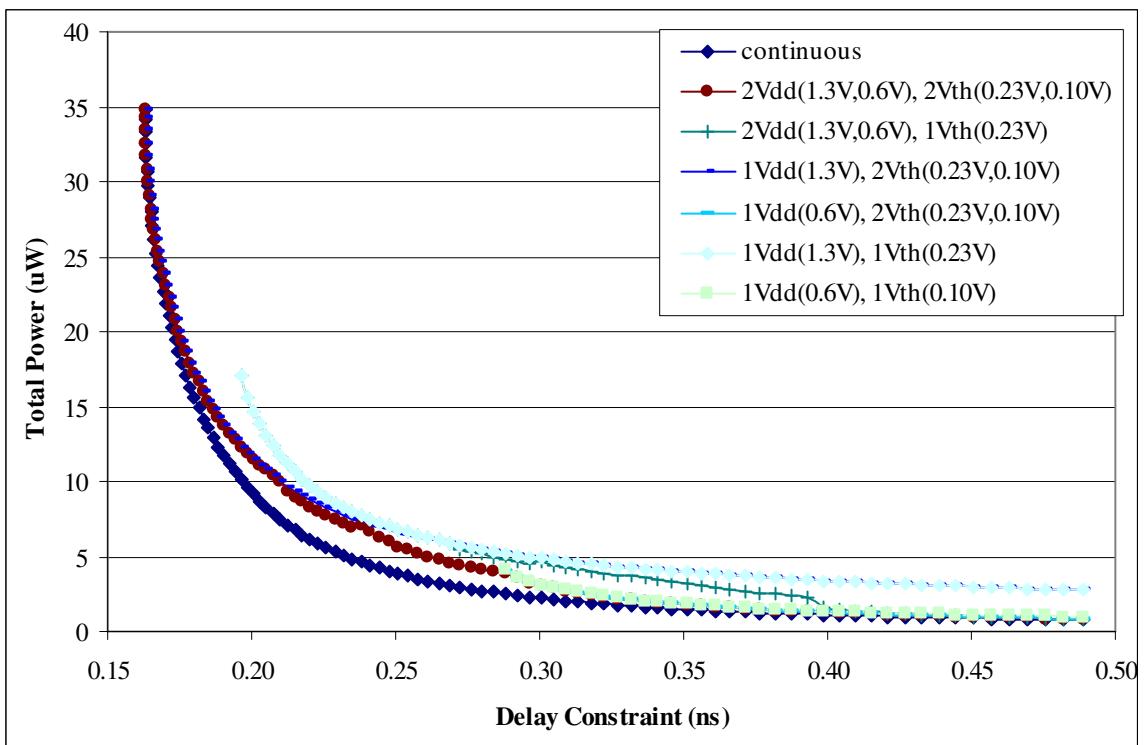


Figure D.25 Dual supply voltage (2Vdd), dual threshold voltage (2Vth) and single voltage (1Vdd, 1Vth) results with fixed global voltage values for c17 across a range of delay constraints.

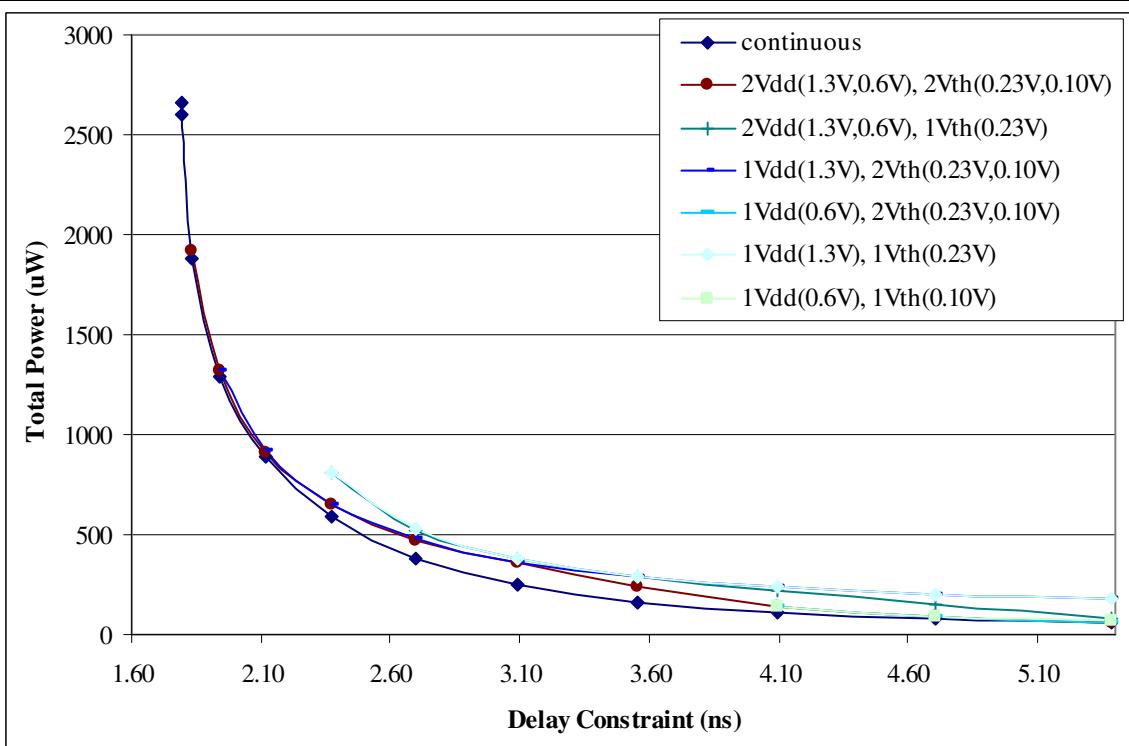


Figure D.26 Dual supply voltage (2Vdd), dual threshold voltage (2Vth) and single voltage (1Vdd, 1Vth) results with fixed global voltage values for c432nr across a range of delay constraints.

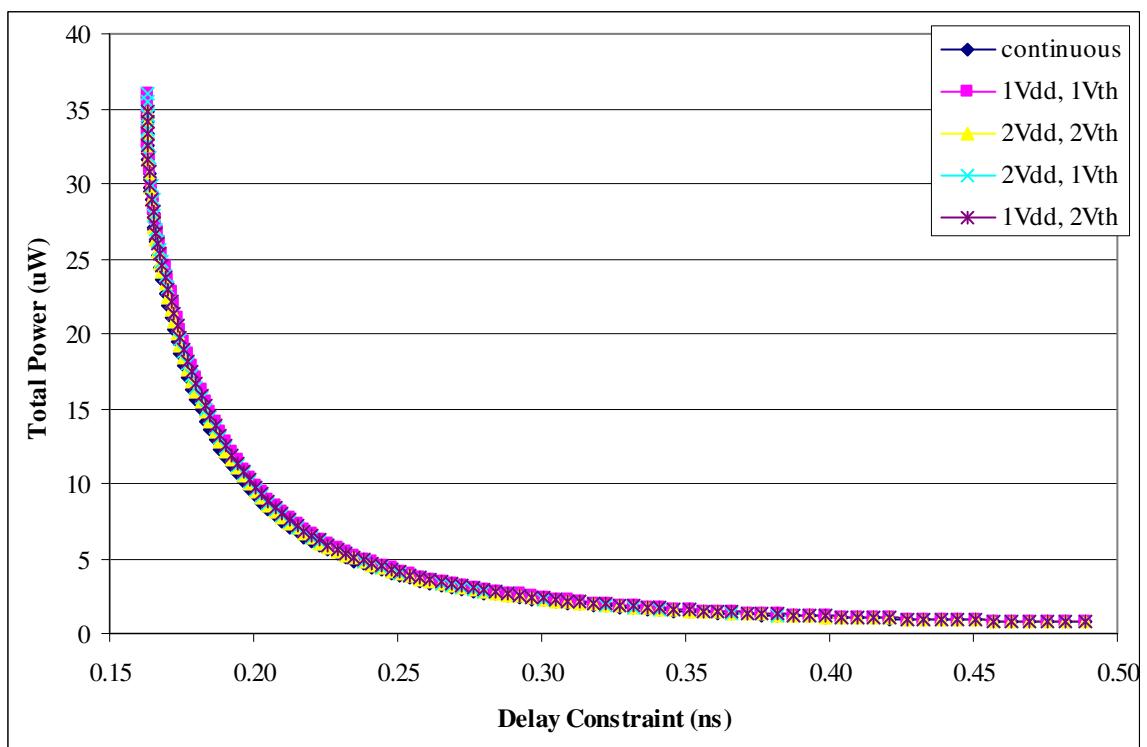


Figure D.27 Continuous voltage, dual supply voltage (2Vdd), dual threshold voltage (2Vth) and single voltage (1Vdd, 1Vth) results for c17 *with wire loads* across a range of delay constraints.

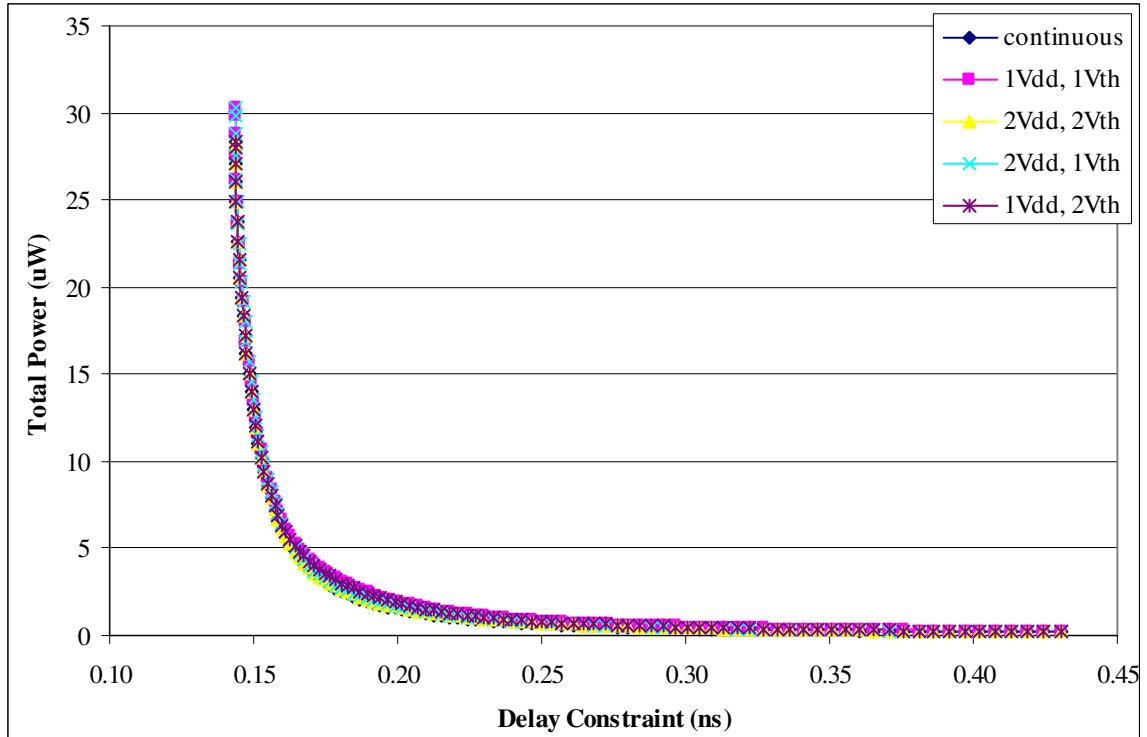


Figure D.28 Continuous voltage, dual supply voltage (2Vdd), dual threshold voltage (2Vth) and single voltage (1Vdd, 1Vth) results for c17 *without wire loads* across a range of delay constraints.

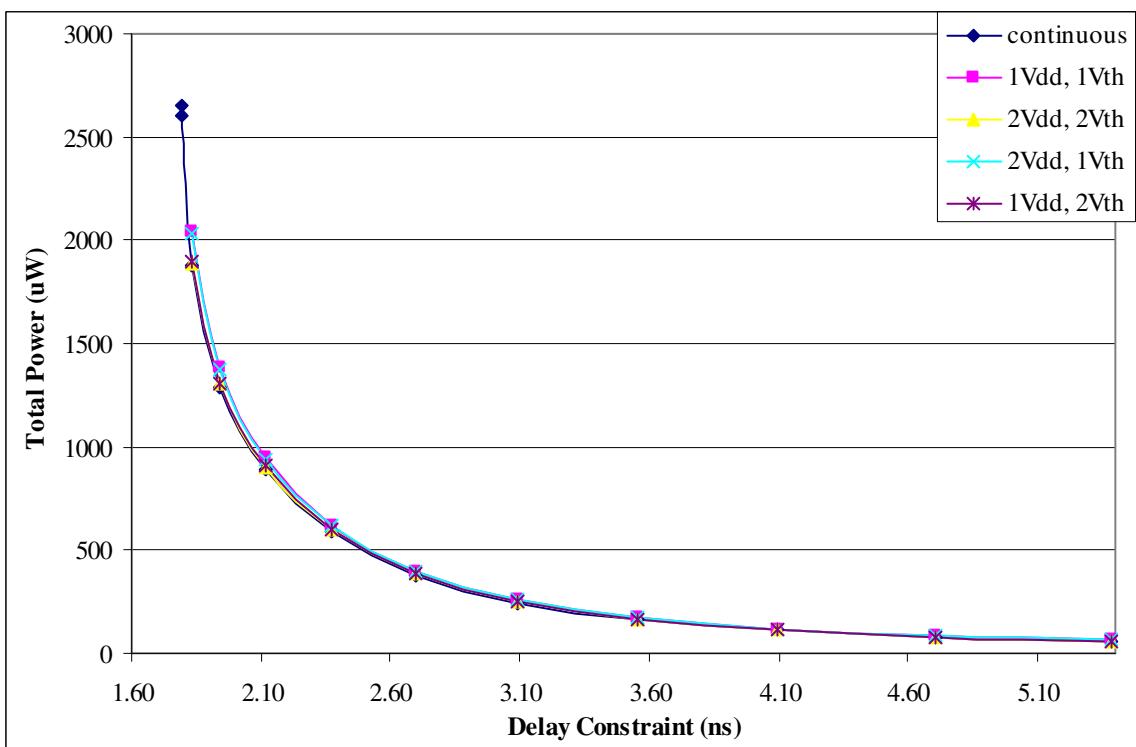


Figure D.29 Continuous voltage, dual supply voltage (2Vdd), dual threshold voltage (2Vth) and single voltage (1Vdd, 1Vth) results for c432nr *with wire loads* across a range of delay constraints.

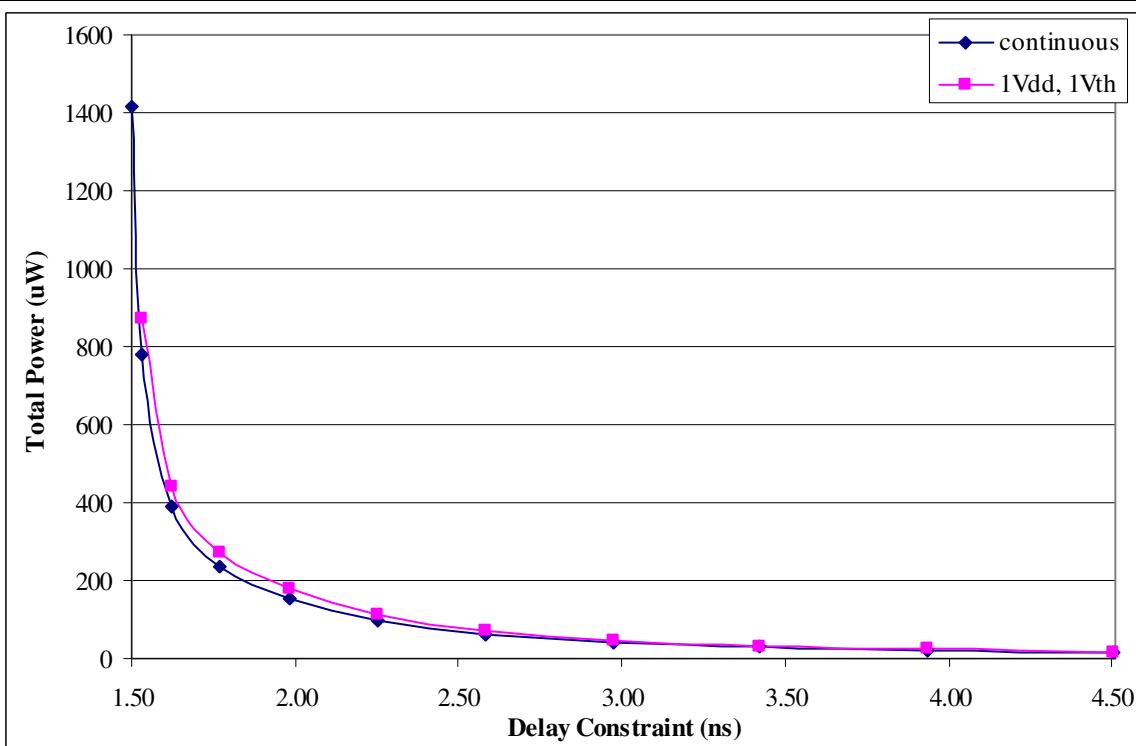


Figure D.30 Continuous voltage and single voltage (1Vdd, 1Vth) results for c432nr *without wire loads* across a range of delay constraints.

Appendix E Static timing and power analysis

This appendix details the accurate reference models for delay and power analysis. This analysis is typically performed in an ASIC flow, and it is essential that timing and power be analyzed in a manner at least this accurate, to guarantee that a chip will not consume too much and will meet delay constraints. The delay and power numbers derived by this analysis are used to specify the delay and power changes that are encoded in the linear program. The static timing and power analysis is also performed after each optimization iteration to ensure accurate delay and power values.

E.1 Static Timing Analysis

Static timing analysis is used to determine the delay from the inputs to the outputs of the circuit. A logic cell has a number of input pins and a number of output pins. In the absence of tri-state buffers and similar devices, it is assumed that each input is driven by only one gate; a gate's output may drive more than one gate input or primary output of the circuit. The output signal from a timing arc is modeled as a linear ramp from 0V to Vdd or Vdd to 0V, with a certain signal slew (time to rise or fall) and arrival time. Arrival time may be measured from when the input is at 50% of Vdd to when the output is at 50% Vdd, or other alternatives such as 40%-60% or 10%-90% – for example from a rising input at 10% of Vdd to falling output at 90% Vdd, and from falling input at 90% of Vdd to rising output at 10% Vdd. The resulting delay analysis is the same providing a consistent approach is used across the circuit, but negative gate delays due to a slow input transition and fast output transition are more likely to occur with 50%-50% delay analysis versus 10%-90% delay analysis. The Synopsys PowerArc characterized libraries that we used had 50%-50% delays, but the static timing analysis software that I wrote supports any of these alternatives. Delays and signal slews of rising and falling signals are specified in lookup tables in the standard cell library. These lookup tables are indexed by the slew on the input signal (input

slew) and the sum of the capacitances driven by the output pin (load capacitance). These tables of the form $f(s_{in}, C_{load})$ typically consist of between a 6×6 to 8×8 grid of points.

For our purposes of analyzing combinational logic, we use the same longest path static timing analysis approach as Design Compiler. Shortest paths may be ignored as we do not need to consider hold time violations for registers, and in such cases for a sequential netlist buffer insertion may be used to fix hold time violations. Secondly, power minimization tends to reduce the number of hold time violations due to short paths, as gate delay typically increases when gates are downsized, V_{th} is increased, or V_{dd} is reduced to reduce power. We did not perform any false path analysis to rule out static timing paths that could not occur, though there are known false paths in some of our benchmarks [83], because false paths were not excluded for the TILOS-like sizer or in Design Compiler which we compare results to.

For a gate i , the timing lookups for output slew s and delay d on a timing arc from input j to output k are as follows. If output k has negative polarity with respect to j , the lookups are

$$\begin{aligned} t_{jk,rise} &= t_{j,fall} + d_{jk,rise}(s_{j,fall}, C_{load,k}) \\ s_{jk,rise} &= s_{jk,rise}(s_{j,fall}, C_{load,k}) \\ t_{jk,fall} &= t_{j,rise} + d_{jk,fall}(s_{j,rise}, C_{load,k}) \\ s_{jk,fall} &= s_{jk,fall}(s_{j,rise}, C_{load,k}) \end{aligned} \tag{E.1}$$

If output k has positive polarity with respect to j , the lookups are

$$\begin{aligned} t_{jk,rise} &= t_{j,rise} + d_{jk,rise}(s_{j,rise}, C_{load,k}) \\ s_{jk,rise} &= s_{jk,rise}(s_{j,rise}, C_{load,k}) \\ t_{jk,fall} &= t_{j,fall} + d_{jk,fall}(s_{j,fall}, C_{load,k}) \\ s_{jk,fall} &= s_{jk,fall}(s_{j,fall}, C_{load,k}) \end{aligned} \tag{E.2}$$

And if output k is nonunate with respect to j , then

$$\begin{aligned}
t_{jk,rise} &= \max\{t_{j,fall} + d_{jk,rise}(s_{j,fall}, C_{load,k}), t_{j,rise} + d_{jk,rise}(s_{j,rise}, C_{load,k})\} \\
s_{jk,rise} &= \max\{s_{jk,rise}(s_{j,fall}, C_{load,k}), s_{jk,rise}(s_{j,rise}, C_{load,k})\} \\
t_{jk,fall} &= \max\{t_{j,rise} + d_{jk,fall}(s_{j,rise}, C_{load,k}), t_{j,fall} + d_{jk,fall}(s_{j,fall}, C_{load,k})\} \\
s_{jk,fall} &= \max\{s_{jk,fall}(s_{j,rise}, C_{load,k}), s_{jk,fall}(s_{j,fall}, C_{load,k})\}
\end{aligned} \tag{E.3}$$

where $t_{j,fall}$ and $t_{j,rise}$ are the arrival times at input pin j , $s_{j,fall}$ and $s_{j,rise}$ are the corresponding signal slews, and $C_{load,k}$ is the load capacitance on output pin k . Positive polarity refers to a rising input causing a rising output if there is a logical transition, or a falling input causing a falling output. Negative polarity refers to a rising input causing a falling output if there is a logical transition, or a falling input causing a rising output. If the gate is nonunate with respect to an input, a rising (or falling) output transition may be caused either by a fall or rising input, depending on the value of other inputs, such as occurs in an XOR gate.

To determine the value of $d(s_{in}, C_{load})$ and $s_{out}(s_{in}, C_{load})$ at input slew and load capacitance points that are not grid points in the lookup table, interpolation is performed as follows.

E.2 Nonlinear delay interpolation between table lookup grid points

Values in a circuit will not correspond exactly to grid points for the standard cell library lookup tables. *Nonlinear delay interpolation* is a standard approach for interpolating standard cell library lookup tables [138], solving $z=ax+by+cxy+d$ to fit the four nearest grid points of the form (x,y,z) and then finding the value of z at the given x and y values. This is the method used by Synopsys Design Compiler [202] for static timing and power analysis with our libraries. We used the nonlinear delay interpolation method for consistency between our results and Design Compiler. Our software also includes the least squares interpolation method which is also commonly used, where $z=ax+by+c$ is fit to the nearest four points to minimize the sum of the squares of the fit errors.

The four nearest grid points are determined as follows:

Table lookup index arrays are ordered from smallest to largest value

$$S = \{s_1, s_2, \dots, s_p\} \text{ and } C = \{C_1, C_2, \dots, C_q\}.$$

If $s_{in} \leq \min(S)$, $s_{lower} = s_1$ and $s_{upper} = s_2$,

else if $s_{in} \leq \max(S)$, $\exists r : s_{lower} = s_r < s_{in} \leq s_{upper} = s_{r+1}$,

else there is an error as the maximum input slew of the cell is exceeded.

Similarly, if $C_{load} \leq \min(C)$, $C_{lower} = C_1$ and $C_{upper} = C_2$,

else if $C_{load} \leq \max(C)$, $\exists r : C_{lower} = C_r < C_{load} \leq C_{upper} = C_{r+1}$,

else there is an error as the maximum load capacitance of the cell is exceeded.

The four nearest points are then:

$$\begin{aligned} & (s_{lower}, C_{lower}, f(s_{lower}, C_{lower})), \\ & (s_{upper}, C_{lower}, f(s_{upper}, C_{lower})), \\ & (s_{lower}, C_{upper}, f(s_{lower}, C_{upper})), \\ & (s_{upper}, C_{upper}, f(s_{upper}, C_{upper})) \end{aligned} \quad (\text{E.4})$$

where the f values are from the library.

We then solve for the coefficients a , b , c and d in the function $f = as + bC + csC + d$ at these four grid points. Then we calculate $f(s_{in}, C_{load}) = as_{in} + bC_{load} + cs_{in}C_{load} + d$. As this calculation is a core part of the inner loop of the optimization routine which analyzes delay and power tradeoffs, we want to minimize the number of calculations to calculate this fit. We do this by shifting the origin of the (s_{in}, C_{load}) variables to (s_{lower}, C_{lower}) to determine $f(s_{in}, C_{load})$ as follows:

$$\begin{aligned}
\Delta s_{in} &= s_{in} - s_{lower} \\
\Delta s_{upper} &= s_{upper} - s_{lower} \\
\Delta C_{load} &= C_{load} - C_{lower} \\
\Delta C_{upper} &= C_{upper} - C_{lower}
\end{aligned}$$

$$\begin{aligned}
f(s_{in}, C_{load}) &= f(s_{lower}, C_{lower}) + \left(\frac{\Delta s_{in}}{\Delta s_{upper}} \right) (f(s_{upper}, C_{lower}) - f(s_{lower}, C_{lower})) \\
&+ \left(\frac{\Delta C_{load}}{\Delta C_{upper}} \right) (f(s_{lower}, C_{upper}) - f(s_{lower}, C_{lower})) \\
&+ \left(\frac{\Delta s_{in}}{\Delta s_{upper}} \right) \left(\frac{\Delta C_{load}}{\Delta C_{upper}} \right) (f(s_{lower}, C_{lower}) + f(s_{upper}, C_{upper}) - f(s_{upper}, C_{lower}) - f(s_{lower}, C_{upper}))
\end{aligned} \tag{E.5}$$

This implementation has 3 divisions, 4 multiplications and 12 additions/subtractions in the nondegenerate case (i.e. $s_{lower} \neq s_{upper}$, $C_{lower} \neq C_{upper}$). This can be further reduced to 2 divisions, 3 multiplications and 12 additions/subtractions as follows:

$$\begin{aligned}
\Delta s_{in} &= s_{in} - s_{lower} \\
\Delta s_{upper} &= s_{upper} - s_{lower} \\
s_{ratio} &= \frac{\Delta s_{in}}{\Delta s_{upper}} \\
\Delta C_{load} &= C_{load} - C_{lower} \\
\Delta C_{upper} &= C_{upper} - C_{lower}
\end{aligned} \tag{E.6}$$

$$\begin{aligned}
f(s_{in}, C_{load}) &= f(s_{lower}, C_{lower}) + s_{ratio} (f(s_{upper}, C_{lower}) - f(s_{lower}, C_{lower})) \\
&+ \left(\frac{\Delta C_{load}}{\Delta C_{upper}} \right) \left(s_{ratio} (f(s_{lower}, C_{lower}) + f(s_{upper}, C_{upper}) - f(s_{upper}, C_{lower}) - f(s_{lower}, C_{upper})) \right)
\end{aligned}$$

Having determined a gate's rise and fall output delay and slew on each timing arc from interpolating lookup tables, we must now specify how multiple rise and fall timing arcs are merged at the gate output.

E.3 Calculating the timing envelopes on outputs

There are multiple timing arcs through a gate that may lead to a rising or falling transition at the gate output. To propagate this information to the gate’s fanouts, we need to summarize this information with a single output rise and single output fall arrival time and slew. In particular, standard cell libraries do not typically specify what happens in the event of multiple input arrivals causing an output transition.

To agree with Design Compiler timing results, we used $\max\{t_{arrival}\}$ and $\max\{s_{out}\}$ over gate timing arcs to provide the bounding signal for a rising output and for a falling output. The delay and output slew of a gate v at output pin k are determined by

$$\begin{aligned} t_{k,rise} &= \max_{j \in \text{input_pins}(v)} \{t_{jk,rise}\} \\ t_{k,fall} &= \max_{j \in \text{input_pins}(v)} \{t_{jk,fall}\} \\ s_{k,rise} &= \max_{j \in \text{input_pins}(v)} \{s_{jk,rise}\} \\ s_{k,fall} &= \max_{j \in \text{input_pins}(v)} \{s_{jk,fall}\} \end{aligned} \tag{E.7}$$

This predominantly determines the delay and slew propagated to fanouts, but in addition wire RC (resistance \times capacitance) delays are included as described in Section E.5. This “maximum slew method” can be overly conservative, as an input that arrives early with large input slew may cause large output slew, but a fast late arriving input may determine the actual final output transition at the gate with smaller output slew – taking the maximum of the output slews due to the early and late arriving inputs is pessimistic.

Less conservative analysis methods are supported in the software: full timing envelope, half timing envelope, and the latest arriving method. Assuming 50%-50% delay analysis, these methods are as below for a rising output transition [134].

Latest arriving:

$$\begin{aligned} t_{k,rise} &= \max_{j \in \text{input_pins}(v)} \{t_{jk,rise}\} \\ s_{k,rise} &= s_{jk,rise}, \text{ where } j = \operatorname{argmax}_{j \in \text{input_pins}(v)} \{t_{jk,rise}\} \end{aligned} \quad (\text{E.8})$$

Full envelope:

$$\begin{aligned} t_{k,rise} &= \left(\max_{j \in \text{input_pins}(v)} \{t_{jk,rise} + s_{jk,rise}/2\} + \max_{j \in \text{input_pins}(v)} \{t_{jk,rise} - s_{jk,rise}/2\} \right) / 2 \\ s_{k,rise} &= \left(\max_{j \in \text{input_pins}(v)} \{t_{jk,rise} + s_{jk,rise}/2\} - \max_{j \in \text{input_pins}(v)} \{t_{jk,rise} - s_{jk,rise}/2\} \right) \end{aligned} \quad (\text{E.9})$$

Half envelope:

$$\begin{aligned} t_{k,rise} &= \max_{j \in \text{input_pins}(v)} \{t_{jk,rise}\} \\ s_{k,rise} &= 2 \left(\max_{j \in \text{input_pins}(v)} \{t_{jk,rise} + s_{jk,rise}/2\} - \max_{j \in \text{input_pins}(v)} \{t_{jk,rise}\} \right) \end{aligned} \quad (\text{E.10})$$

Here $t_{jk,rise} - s_{jk,rise}/2$ is the start time of the rising ramp, and $t_{jk,rise} + s_{jk,rise}/2$ is the end time of the rising ramp. The latest arriving method is optimistic in some cases [134]. The full and half timing envelope methods are less pessimistic than the “maximum slew” method used by Design Compiler.

The half envelope method is less pessimistic than the full envelope, but still conservative according to [134] given that a “causality assumption” is met. The causality assumption is that if two input signal ramps are not overlapping, then the corresponding output signals will be non-overlapping in the same order. This translates to the requirement that [134]

$$1 \geq 2 \times \max_{s_{in}} \left\{ \frac{\Delta d}{\Delta s_{in}} \right\} + \max_{s_{in}} \left\{ \frac{\Delta s_{out}}{\Delta s_{in}} \right\} \quad (\text{E.11})$$

where Δd and Δs_{out} are respectively the change in gate delay and output slew due to a change in input slew Δs_{in} , assuming arrival times are measured from input at 50% of Vdd to output at 50%

of Vdd. Other cases can be handled by replacing the factor of 2 in Equation (E.11) by $1/\text{falling_input_arrival_threshold}$, for example it would be $1/0.9$ for 10%-90% delays. However, careful analysis of both the STMicroelectronics CORE9GPHS library [191] and Synopsys PowerArc characterized libraries for STMicroelectronics HCMOS9D process indicates that the “causality assumption” in [134] is not always true. For example, for the X6 drive strength inverter in the PowerArc Vdd=1.2V/Vth=0.23V library, $\max\{\Delta d/\Delta s_{in}\}$ is 0.27 and $\max\{\Delta s_{out}/\Delta s_{in}\}$ is 0.47, which violates the causality assumption. So there are cases where the half envelope method is optimistic. Ranges for $\Delta d/\Delta s_{in}$ and $\Delta s_{out}/\Delta s_{in}$ are listed in Table E.1.

In practice, these different timing analysis methods varied by at most 2% for the circuits we examined. We now address static power analysis.

Table E.1 This table lists the minimum and maximum impact of a change in input slew Δs_{in} to a gate on the gate's delay Δd and output slew Δs_{out} for the 0.13um PowerArc characterized libraries that were used in these experiments. The libraries were provided by Ashish Srivastava and Sarvesh Kulkarni – the level converters are the strength 5 level converters discussed in [123]. The second set of libraries with more cell sizes has a larger range of slew sensitivities. These slew sensitivities are comparable to those found in STMicroelectronics CORE9GPHS library for the HCMOS9D process. Note that the Vdd=0.6V delay and slews were incorrectly characterized with Vdd=0.8V; corrected by $(0.6/0.8) \times ((0.8 - V_{th})/(0.6 - V_{th}))^{1.3}$ multiplying the delays (but not slews) per the Sakurai-Newton alpha power law delay model [173] with alpha of 1.3 (see Section 5.2.1 for more details).

Library Cells	Vin(V)	Vdd(V)	Vth(V)	$\Delta d/\Delta s_{in}$		$\Delta s_{out}/\Delta S_{in}$	
				max	min	max	min
Inverter: XL, X1, X2, X3, X4, X8, X12, X16, X20	1.2	1.2	0.23	0.25	-0.08	0.33	-0.01
	1.2	1.2	0.14	0.25	-0.11	0.40	-0.02
	1.2	1.2	0.12	0.25	-0.11	0.42	-0.01
	1.2	1.2	0.08	0.26	-0.13	0.48	-0.01
	1.2	0.8	0.23	0.41	-0.22	0.29	-0.04
	1.2	0.8	0.14	0.40	-0.24	0.31	-0.03
	1.2	0.8	0.12	0.40	-0.25	0.32	-0.02
	1.2	0.8	0.08	0.39	-0.26	0.33	-0.02
	0.8	0.8	0.23	0.33	-0.03	0.29	-0.04
	0.8	0.8	0.14	0.27	-0.06	0.35	-0.03
	0.8	0.8	0.12	0.26	-0.07	0.35	-0.02
	0.8	0.8	0.08	0.25	-0.09	0.41	-0.02
	1.2	0.6	0.23	0.54	-0.29	0.29	-0.04
	1.2	0.6	0.12	0.47	-0.29	0.32	-0.02
	0.6	0.6	0.23	0.43	-0.03	0.29	-0.04
	0.6	0.6	0.14	0.33	-0.07	0.35	-0.03
	0.6	0.6	0.12	0.31	-0.08	0.35	-0.02
	0.6	0.6	0.08	0.29	-0.10	0.41	-0.02
Inverter: XL, X1, X2, X3, X4, X6, X8, X10, X12, X14, X16, X18, X20	1.2	1.2	0.23	0.33	-0.25	0.57	-0.23
	1.2	1.2	0.12	0.32	-0.32	0.62	-0.20
	0.6	0.6	0.23	0.52	-0.14	0.55	-0.22
	0.6	0.6	0.12	0.43	-0.19	0.67	-0.23
	0.8	1.2	dual Vth	0.23	0.00	0.41	0.00
Level converters (two drive strengths)	0.6	1.2	dual Vth	0.33	-0.01	0.43	-0.02

E.4 Static power analysis

The power consumption consists of three portions: switching power for (dis)charging the input capacitance of gates and wire loads; internal power of gates, including (dis)charging internal capacitances and short circuit current from supply to ground when both pull-up and pull-down transistor chains are conducting; and leakage power – primarily subthreshold leakage, but the

SPICE decks also include modeling of gate leakage, which contributes on the order of 1% of the total leakage.

The switching power due to charging and discharging capacitances $C_{load,j}$ over each output j of a gate v with supply voltage $V_{dd,v}$ is given by

$$P_{switching,v} = \frac{1}{2T} \sum_{j \in output_pins(v)} \Pr(\text{transition on } j \text{ from 0 to 1, or 1 to 0}) C_{load,j} V_{dd,v}^2 \quad (\text{E.12})$$

where $\Pr(\text{transition ...})$ denotes the probability of a signal transition, and T is the clock period – $1/T$ is the clock frequency f . This assumes the capacitance is charged to V_{dd} , where it stores charge $C_{load}V_{dd}$, and discharged to 0V.

The internal power of a gate v comprises power due to (dis)charging internal capacitances in the logic cell and any short circuit current from supply to ground when the gate is switching. The internal power of a gate is determined on timing arcs from input to output, and in some libraries may include power for transitions on a given input pin:

$$\begin{aligned} P_{internal,v} = & \frac{1}{T} \sum_{j \in output_pins(v)} \sum_{k \in input_pins(v)} \sum_{l \in arc_transition_conditions(v,k,j)} \Pr(l) E_{internal}(s_{in,k}, C_{load,j}, l) \\ & + \frac{1}{T} \sum_{k \in input_pins(v)} \sum_{l \in input_transition_conditions(v,k)} \Pr(l) E_{internal}(s_{in,k}, l) \end{aligned} \quad (\text{E.13})$$

where $\Pr(l)$ is the probability of transition l occurring, and $E_{internal}(s_{in,k}, l)$ or $E_{internal}(s_{in,k}, C_{load,j}, l)$ is the corresponding internal energy specified in the library. Note that in the Liberty .lib format, internal energy is referred to as internal power, which is a misnomer because the actual units are energy until we multiply by the switching frequency.

An example of an internal power condition (*arc_transition_condition* in the above equation when combined with timing arc input pin and output pin, and fall/rise information) for a NOR2 gate,

with inputs A and B and output Z, is !B (i.e. B=0), when a falling transition from 1 to 0 on input A causes the output Z to rise.

The leakage power is

$$P_{leakage,v} = \sum_{l \in input_conditions(v)} \Pr(l) P_{leakage}(l) \quad (\text{E.14})$$

where $P_{leakage}(l)$ is the leakage power specified for the cell for an input state l with probability $\Pr(l)$; the *input_conditions* are Booleans representing sets of input states for which input power was characterized. For example, for a NOR3 gate the list of input values for which leakage is specified may be “default”, $\neg A \times \neg B \times \neg C$, $A \times \neg B \times \neg C$, $\neg A \times B \times \neg C$, and $\neg A \times \neg B \times C$. Generally, the input conditions corresponding to the larger leakage currents are characterized. “Default” characterization conditions may be specified in the library to handle cases that are not in the list of input states, though the default leakage is typically an average of leakage over the dominant leakage states.

The switching probabilities and gate input state probabilities for leakage are determined by gate level logic simulation. Synopsys VCS [204] was used for this purpose. In general, it would be preferable to have specific simulation benchmarks to stimulate a netlist, for example to avoid toggling a reset signal. Unfortunately, such vector sequences are not available for the benchmarks that we used. We assumed independent random primary inputs, with equal probabilities of 0 or 1. This seems a reasonable assumption for our combinational benchmarks, as they don’t have sequential elements that require a reset signal.

E.5 Wire delays and wire capacitance (wire load model)

Unless the circuit being optimized has had cells placed and wires routed, the resistance and capacitance for the wires are unknown. A wire load model is used to estimate the resistance and capacitance versus the number of fanouts that a gate output connects to. Typically, the wire load

model may be specified by area, such that wire lengths are longer and corresponding wire loads are larger in a bigger design. The wire load models are specified in the standard cell library, though they could also be generated from a design layout. Back annotated capacitances with specific capacitances for each wire in the layout are not supported in the software currently, though it would be easy to do so.

The wire capacitances add directly to the load capacitances that the gate drives. The wire RC (resistance \times capacitance) also adds to the delay, specifically the RC delay d_{RC} is given by:

$$\begin{aligned} C_{\text{wire/fanout}} &= \frac{\text{WireLoadModel_Capacitance}(\text{total_cell_area}, \# \text{fanout})}{\# \text{fanout}} \\ R_{\text{wire}} &= \text{WireLoadModel_Resistance}(\text{total_cell_area}, \# \text{fanout}) \\ d_{RC} &= R_{\text{wire}}(C_{\text{wire/fanout}} + C_{in}) \end{aligned} \quad (\text{E.15})$$

where C_{in} is the input capacitance of the gate input pin that is being driven or the capacitance of the output port; $C_{\text{wire/fanout}}$ is the wire capacitance on the wire to this pin; and R_{wire} is the total wire resistance. The total cell area determines which wire load model is used, and #fanout is the number of fanouts for the net. This is the calculation that Design Compiler uses, so we do the same to ensure timing analysis accuracy versus Design Compiler.

Wire delays only contribute a small fraction of the total delay on the small benchmark circuits – less than 1%. On the other hand, wire load capacitance is very significant. For example with the 0.13um PowerArc Vdd=1.2V/Vth=0.23V library and the Design Compiler delay minimized ISCAS'85 benchmarks, setting wire loads to zero, instead of $3+2\times\#\text{fanouts}$ fF, reduces delay by 20% on average and reduces switching power by 37% when measured at the original clock frequency. In particular, wire load capacitance increases the circuit delay and limits how much gates may be downsized due to driving wire loads as well as gate inputs. Without the wire loads, gates may be further downsized, increasing the delay reduction and power savings. Hence it is essential to include wire loads in analysis and optimization.

Appendix F Design Compiler and VCS Scripts

Scripts were used to map the Verilog benchmarks to the standard cell libraries, perform delay minimization, perform switching activity analysis, and perform power minimization. Examples of these scripts are included here.

```
define_name_rules simple -allowed "A-Za-z0-9_" -map { {"\*cell\*","U"}, {"*-return","RET"} }
define_name_rules no_underscores -map { {"_",""} }
define_name_rules no_slashes -map { {"/",""} }
define_name_rules no_backslashes -map { {"\"",""} }

link_library=target_library={stdcells_vdd12_vth023.db}
read -f verilog netlist.v

set_flatten true
verilogout_show_unconnected_pins = "true"

set_input_transition 0.1 all_inputs()
set_load 0.003 all_outputs()
set_max_delay 0.0 -to all_outputs()

compile -map_effort high -ungroup_all
compile -incr -map_effort high -ungroup_all

change_names -rules simple
change_names -rules no_underscores
change_names -rules no_slashes
change_names -rules no_backslashes

report_timing -significant_digits 4 -transition_time -capacitance -input_pins > timing.log
report_cell > cell.log
report_power -analysis_effort high -verbose > power.log
report_power -analysis_effort high -verbose -net -cell -cumulative >> power.log

write -f verilog -hier -output netlist_delay_minimized.v
quit
```

Figure F.1 Design Compiler script for mapping the Verilog netlist (netlist.v) to the target standard cell library stdcells_vdd12_vth023.db, which is the PowerArc characterized Vdd=1.2V/Vth=0.23V library, and performing delay minimization. The input transition slew is set to 0.1ns. The load capacitance is set to 0.003pF. The maximum delay on all outputs is set to 0.0ns to produce a delay minimized netlist. Additional commands were included to force certain naming conventions for the wire names and gate instance names in the netlist that is written out, which simplifies parsing the Verilog file.

```

`timescale 1ns / 1ps
`include "c17_delay_minimized.v"

module sim;
    integer datafile;

    reg a;
    reg b;
    reg c;
    reg d;
    reg e;

    integer i;

    c17 c17 ( a, b, c, d, e, f, g );

initial
begin
    $read_lib_saif ("stdcells_vdd12_vth023.saif");
    $set_toggle_region (c17);

    a = 0;
    b = 0;
    c = 0;
    d = 0;
    e = 0;

    #1;

    $toggle_start;

    for (i=0; i<10000;i=i+1)
    begin
        a = ($random)%2;
        b = ($random)%2;
        c = ($random)%2;
        d = ($random)%2;
        e = ($random)%2;
        #1;
    end

    $toggle_stop;
    $toggle_report("c17_delay_minimized.saif",1.0e-9,"sim.c17");

    $finish;
end
endmodule

```

Figure F.2 The file c17_sim.vcs used to run VCS simulation for c17. It is invoked with the command line: vcs -R -P vpower.tab libvpower.a -v stdcells_vdd12_vth023.v c17_sim.vcs. The inputs of c17 are a, b, c, d, and e, and the outputs are f and g. Inputs are initialized to zero, then toggled randomly to 0 or 1. stdcells_vdd12_vth023.saif specifies which input states and transitions need to be recorded for each library cell, and stdcells_vdd12_vth023.v specifies the functional behavior (e.g. NOT, NAND, NOR) of each cell.

```

link_library=target_library={stdcells_vdd12_vth023.db}
read -f verilog netlist_delay_minimized.v

set_input_transition 0.1 all_inputs()
set_load 0.003 all_outputs()
set_max_delay 0.7714 -to all_outputs() //  $1.1 \times T_{\min}$ 

read_saif -input netlist_delay_minimized.saif -instance sim/netlist

set_max_dynamic_power 0
set_max_leakage_power 0
set_size_only find(cell, "*")

compile -incr -map_effort high

report_timing -significant_digits 4 -transition_time -capacitance -input_pins > timing_at_1.1Tmin.log
report_cell > cell_at_1.1Tmin.log
report_power -analysis_effort high -verbose > power_at_1.1Tmin.log
report_power -analysis_effort high -verbose -net -cell -cumulative >> power_at_1.1Tmin.log

write -f verilog -hier -output netlist_power_minimized_at_1.1Tmin.v
quit

```

Figure F.3 Design Compiler script for sizing only power minimization starting from the delay minimized netlist, using the same standard cell library. In this particular script, the delay constraint was set to 1.1 times the minimum delay T_{\min} .

Appendix G Details of the Linear Programming Approach

G.1 Pseudo-code for trying different optimization parameters

The initial settings for the delay minimization objective parameters are $\tau = 0.99$ and $k = 0.01$ (see Equation (4.21)). If the user specifies that slew analysis should be aggressive, the initial settings for slew analysis don't consider alternate cells (`doAlternates = false`) and only consider the current input slew and load capacitance (`doCurrentConditionsOnly = true`); if the user specifies that analysis should be conservative these two Boolean variables have opposite Boolean values. For both aggressive and conservative settings, `compareTotalTimingArcDeltaDelays` is initially false, that is the worst case delay change is used to determine the best cell alternative for a gate as described in Section 4.4.3. Section 4.4.2 discussed the motivation for considering different approaches to slew analysis to determine the transitive fanout delay impact of slew (β values in the delay constraints of Equation (4.20) and Equation (4.22)). Figure G.1 outlines in pseudo-code the approach to trying different τ , k , `doAlternates` and `doCurrentConditionsOnly` values when delay reduction is failing to get the critical path delay T below T_{\max} , or when there has been minimal progress in power reduction (less than 1% improvement in the last five iterations since a parameter change). These heuristic approaches to slew analysis and weightings in the delay optimization objective produced the best results over a variety of different values and changes that were tried. Setting $\tau = 0.98$ and $k = 0.001$ increases the weighting on delay reduction in Equation (4.21), but generally produces worse results than $\tau = 0.99$ and $k = 0.01$.

```

while (# of iterations < maximum number of iterations) {
    if ((not repeated circuit) & ( $T < T_{\max}$ )) {
        priorityMinPower = true; // do power minimization
    } else {
        priorityMinPower = false; // do delay reduction

        if (# of delay iterations since last power iteration or since last parameter change > 3) {
            if (doCurrentConditionsOnly & ( $\tau \neq 0.98$ ) & ( $k \neq 0.001$ )) {
                doCurrentConditionsOnly = false;
            } else if (!doCurrentConditionsOnly & !doAlternates & ( $\tau \neq 0.98$ ) & ( $k \neq 0.001$ )) {
                doCurrentConditionsOnly = true;
                 $\tau = 0.98$ ;  $k = 0.001$ ;
            } else if (doCurrentConditionsOnly & ( $\tau == 0.98$ ) & ( $k == 0.001$ )) {
                doCurrentConditionsOnly = false;
            } else if (doAlternates & ( $\tau \neq 0.98$ ) & ( $k \neq 0.001$ )) {
                 $\tau = 0.98$ ;  $k = 0.001$ ;
            }
        } else if ((circuit hasn't changed in 3 iterations) ||
                    (power reduction in 5 iterations since last parameter change < 1%)) {
            if (aggressive) {
                if (!compareTotalTimingArcDeltaDelays & ( $\tau \neq 0.98$ ) & ( $k \neq 0.001$ )) {
                     $\tau = 0.98$ ;  $k = 0.001$ ;
                } else if (!compareTotalTimingArcDeltaDelays & ( $\tau == 0.98$ ) & ( $k == 0.001$ )) {
                    compareTotalTimingArcDeltaDelays = true;
                     $\tau = 0.99$ ;  $k = 0.01$ ;
                } else if (compareTotalTimingArcDeltaDelays & ( $\tau \neq 0.98$ ) & ( $k == 0.001$ )) {
                     $\tau = 0.98$ ;  $k = 0.001$ ;
                } else if (compareTotalTimingArcDeltaDelays & ( $\tau == 0.98$ ) & ( $k == 0.001$ )) {
                    // try original settings again
                    compareTotalTimingArcDeltaDelays = false;
                     $\tau = 0.99$ ;  $k = 0.01$ ;
                }
            } else { // conservative
                if (doAlternates) {
                    doAlternates = false;
                    doCurrentConditionsOnly = true;
                } else if (!compareTotalTimingArcDeltaDelays) {
                    compareTotalTimingArcDeltaDelays = true;
                } else if (!doAlternates & ( $\tau == 0.98$ ) & ( $k == 0.001$ )) {
                    // could be cycling with delay decreases/increases too large to converge but delay
                    // was increased to meet delay constraints, instead set doAlternates true
                    doAlternates = true;
                     $\tau = 0.99$ ;  $k = 0.01$ ;
                } else if (compareTotalTimingArcDeltaDelays) {
                    compareTotalTimingArcDeltaDelays = false;
                }
            }
        }
    }
}

Do power minimization or delay reduction as specified by the PriorityMinPower variable
}

```

Figure G.1 Pseudo-code for choosing slew analysis settings (aggressive or conservative specified by the user), and τ and k values for delay minimization objective (per Equation (4.21)).

G.2 Impact of excluding slew analysis on linear programming results

We can examine the importance of the β terms in Section 4.4.2 that encapsulate the transitive fanout delay impact of slew, by setting $\beta = 0$ and performing optimization. Table G.1 and Table G.2 compare results with slew sensitivity set to zero versus more accurate slew analysis. For most of the benchmarks in Table G.1, the inaccuracy due to ignoring slew results in a violation of the delay constraints, and the delay reduction phase of the optimization flow fails to satisfy the delay constraints due to ignoring slew. The addition of a lower threshold voltage in Table G.2 adds timing slack, which makes it possible to satisfy delay constraints, but the results are still on average 6.2% worse due to ignoring slew. This indicates how important it is to account for slew both in static timing analysis and in the actual optimization formulation.

For this analysis the switching activity was multiplied by a fraction such that leakage power comprised 1% of total power. The delay constraint was $1.2 \times T_{\min}$, where T_{\min} is the delay found by delay minimization in Design Compiler with Vdd of 1.2V and Vth of 0.23V. The Design Compiler delay minimized netlists at Vdd=1.2V/Vth=0.23V were used as the starting point for optimization.

Table G.1 Comparison of the minimum power found by linear programming optimization with zero slew sensitivity (β set to zero) versus the minimum power found with more accurate slew analysis. This was for the PowerArc characterized 0.13um library with Vdd of 1.2V and Vth of 0.23V.

Netlist	Zero slew sensitivity		Power Worse by
	Power (mW)	Minimum Power (mW)	
c17	0.0128	0.0127	1.0%
c432	0.3643	0.2331	56.3%
c499	1.3445	0.5277	154.8%
c880	0.3789	0.3222	17.6%
c1355	1.6950	0.6253	171.1%
c1908	1.1324	0.4794	136.2%
c2670	1.7137	0.7496	128.6%
c3540	2.1487	0.9634	123.0%
c5315	2.5223	1.2330	104.6%
c6288	5.9060	2.9827	98.0%
c7552	4.2357	2.0081	110.9%
		Average	100.2%

Table G.2 Comparison of the minimum power found by linear programming optimization with zero slew sensitivity (β set to zero) versus the minimum power found with more accurate slew analysis. This was for the PowerArc characterized 0.13um libraries with Vdd of 1.2V and 0.8V and Vth of 0.23V and 0.08V, and a clustered voltage scaling methodology

Netlist	Zero slew sensitivity Power (mW)	Minimum Power (mW)	Power Worse by
c17	0.0122	0.0119	2.0%
c432	0.2394	0.2228	7.5%
c499	0.5392	0.5071	6.3%
c880	0.3289	0.2912	12.9%
c1355	0.6468	0.6150	5.2%
c1908	0.4888	0.4576	6.8%
c2670	0.6823	0.6614	3.2%
c3540	0.9754	0.9244	5.5%
c5315	1.2036	1.1398	5.6%
c6288	3.1283	2.9034	7.7%
c7552	2.0367	1.9373	5.1%
Average		6.2%	

G.3 Problems with delay reduction at a tight delay constraint

At a relatively tight delay constraint, the linear programming delay reduction phase can perform poorly, limiting power savings. Table G.3 shows a comparison of the LP optimized power for a delay constraint of $1.1T_{\min}$ with starting points of $1.0T_{\min}$ and $1.1T_{\min}$. When the initial circuit has no slack, starting with the $1.1T_{\min}$ TILOS power minimized netlist, and the delay constraint is fairly tight, the LP delay reduction phase can struggle to reduce delay below $T_{\max}=1.1\times T_{\min}$, which results on average in 4.6% worse results. In practice, this situation can be identified and avoided by starting from a delay minimized netlist which has sufficient timing slack to avoid this problem. This problem also doesn't occur in situations where a lower threshold voltage library is being considered to reduce total power and the delay constraint is not as tight because of the option of using low Vth. For circuits where power dissipation is a major design constraint there is usually some timing slack, as a tight delay constraint corresponds to many gates being upsized (upsized to increase speed, but they also load their fanins, which must be upsized further to achieve the same speed increase) with very high dynamic power compared to a more relaxed delay constraint (e.g. see Figure G.2).

Table G.3 This table compares the minimum power found by the linear program optimization approach at a delay target of $1.1 \times T_{\min}$, where T_{\min} is the delay of the TILOS delay minimized netlist. The best results are produced by starting with the delay minimized netlist. Starting with the netlist that has been power-minimized by the TILOS approach for a constraint of $1.1 \times T_{\min}$ reduces the slack available for the linear program, and in some cases the delay reduction phase has problems meeting the delay constraints which produces significantly worse LP optimization results – these results are highlighted in red.

Netlist	Starting with TILOS delay minimized netlist, total power (mW)	Starting with TILOS $1.1 \times T_{\min}$ power minimized netlist, total power (mW)	Power Increase
c432	0.1699	0.1840	8.3%
c880	0.2841	0.3160	11.2%
c1355	0.6212	0.6276	1.0%
c1908	0.3905	0.4149	6.2%
c2670	0.5589	0.5549	-0.7%
c3540	0.6366	0.6478	1.8%
c5315	1.0622	1.1790	11.0%
c7552	0.9593	0.9622	0.3%
Huffman	0.3328	0.3390	1.9%
Average:			4.6%

G.4 Comparison versus TILOS

The ISCAS'85 netlists and SAIF files for comparison with TILOS and Sarvesh Kulkarni and Ashish Srivastava's results were provided by them, synthesized from the same behavioral netlists to size X1 gates, and then sized with TILOS to minimize power at a given delay constraint. Their TILOS sizer does not perform buffer insertion or other netlist manipulations to reduce the number of logical levels and minimize delay, unlike Design Compiler. Thus the TILOS delay minimized netlists have on average 20% more logical levels and 35% worse delay (measured with the $V_{dd}=1.2V/V_{th}=0.12V$ library), but on average 26% fewer gates than the netlists from Design Compiler – compare Table G.9 and Table 6.2, noting that delays in Table 6.2 are worse due to using the $V_{dd}=1.2V/V_{th}=0.23V$ library instead of the $V_{dd}=1.2V/V_{th}=0.12V$ library. Netlists c2670 and c7552 mapped by Design Compiler have fewer outputs, as redundant outputs and accompanying Verilog “assign” statements were manually removed to simplify parsing the netlists. The c5315 netlist from Sarvesh Kulkarni and Ashish Srivastava is missing 24 outputs, as they remove output ports which are driven by wires that also fanout to other gates.

Sarvesh Kulkarni and Ashish Srivastava provided power results from their TILOS-like sizer for a sweep of delay constraints from $1.0 \times T_{\min}$ to $1.5 \times T_{\min}$ (normalized results are shown in Figure G.2).

The first concern was to double-check timing and power analysis accuracy on the two sets of netlists that were provided at delay constraints of $1.0 \times T_{\min}$ and $1.1 \times T_{\min}$. Having checked accuracy, I then compare results at several delay constraints: $1.1 \times T_{\min}$, $1.2 \times T_{\min}$ and $1.5 \times T_{\min}$. The next subsection discusses the accuracy of static timing and power analysis versus Design Compiler, and then Appendix G.4.2 details the comparison.

For results in this section, the PowerArc characterized 0.13um library at 25°C with Vdd of 1.2V and Vth of 0.12V was used. The channel length was 0.13um. This was characterized for STMicroelectronics 0.13um HCMOS9D process. There were thirteen inverter cell sizes: XL, X1, X2, X3, X4, X6, X8, X10, X12, X14, X16, X18, X20; and eight sizes for NAND2, NAND3, NOR2 and NOR3: XL, X1, X2, X4, X5, X6, X7, X8. Note that results with the TILOS-like sizer started from a netlist of size X1 gates and then only considered upsizing gates. Consequently, size XL cells were not used for the TILOS-like sizing results, though the LP approach did allow the minimum XL cell size.

The results in this section are with the PowerArc 0.13um library with Vdd of 1.2V and Vth of 0.12V. The input drivers were voltage ramps with 0.1ns slew. The output port load capacitance was set to 3fF, which is reasonable if the combinational outputs drive flip-flops, and in addition there is a wire load to the port. The wire load model used was $3 + 2 \times \text{fanout}$ fF. The same library, wire load, input slew, and load conditions were used in [124] and [188]. Switching activities were multiplied by a fraction such that leakage power was about 20% of total power, for later comparison of power trade-offs with dual Vth and dual Vdd.

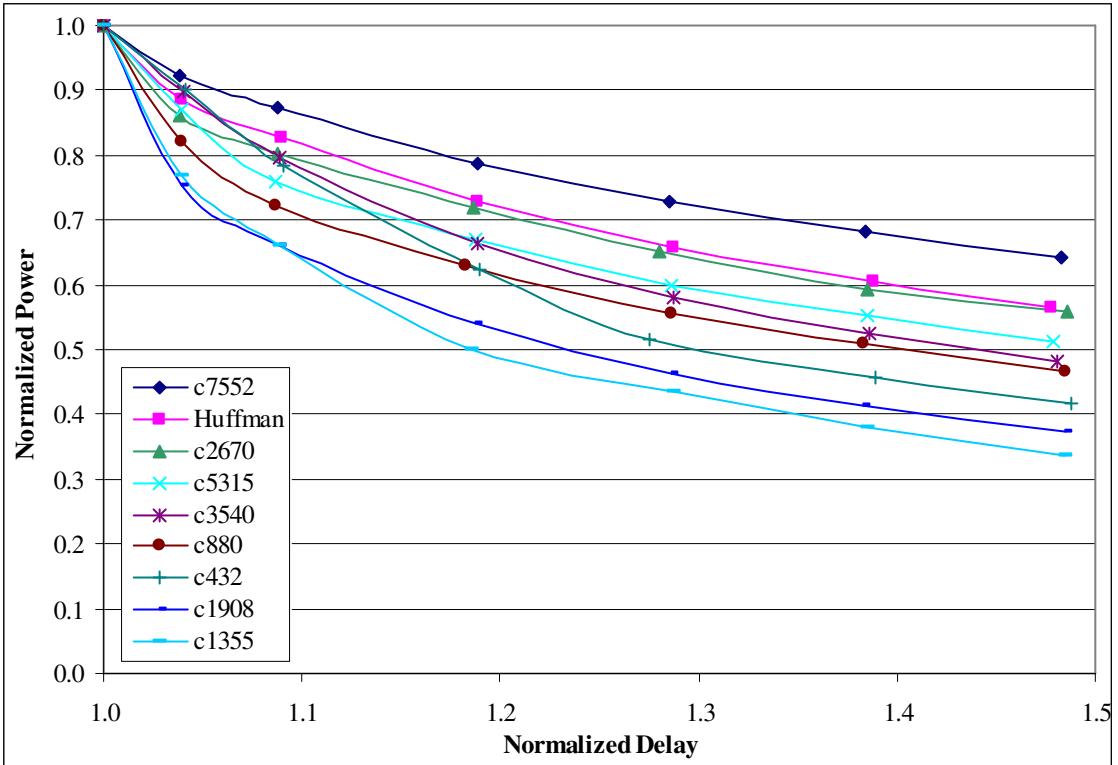


Figure G.2 This graph shows that power savings are greatest (steepest descent) when relaxing from a tight delay constraint. For example for c1355, relaxing the delay constraint from $1.0 \times T_{\min}$ to $1.1 \times T_{\min}$ allows power savings of 34%, and further relaxing from $1.1 \times T_{\min}$ to $1.2 \times T_{\min}$ provides only another 16% power savings. The data is for TILOS sizing power minimization sweeps from $1.0 \times T_{\min}$ to $1.5 \times T_{\min}$ with the PowerArc characterized $V_{dd}=1.2V/V_{th}=0.12V$ $0.13\mu m$ library, and was provided courtesy of Ashish Srivastava and Sarvesh Kulkarni. The power versus delay curves are steeper if voltage scaling is also used.

G.4.1 Accuracy of University of Michigan static timing and power analysis

To compare our optimization results to the work by Ashish Srivastava and Sarvesh Kulkarni in Dennis Sylvester's group at the University of Michigan, the first step was to ensure that we had the same libraries, gate-level netlists, SAIF files, and static timing and power analysis. As the timing and power analysis software differed, it took some time to iron out the differences. Design Compiler's analysis was used as a "golden model" for comparison. Our static timing and power analysis software is accurate to within 0.01% of analysis by Design Compiler.

There remain some issues with the accuracy of the University of Michigan tools versus Design Compiler. Analysis comparison of TILOS power minimized netlists at delay constraints of $1.0 \times T_{\min}$ and $1.1 \times T_{\min}$, where T_{\min} is the minimum delay achievable by gate sizing with the

TILOS-like sizer, are summarized in Table G.4 and Table G.5. Static timing analysis is on average low by 0.3% to 0.4%, as wire RC delays are not included in the University of Michigan static timing analysis. Leakage power numbers generally match Design Compiler, but in a couple of cases are off by 1%. Dynamic power numbers vary significantly from Design Compiler, with a standard deviation of about 5%, which introduces an error with about 4% standard deviation in the total power. Timing accuracy is more important than power accuracy, as we wish to ensure that delay constraints have been met, so the larger error in dynamic power is not a major issue if we compare average results.

I have only received TILOS-sized netlists for the $1.0 \times T_{\min}$ and $1.1 \times T_{\min}$ delay points, though there are other results to compare against. Thus it is important to examine the relative power results, to see if the percentage change tracks similarly for both the University of Michigan and Design Compiler analysis. Table G.6 examines how the results vary relative to each other as the delay constraint is relaxed, for the total power in Table G.4 and Table G.5. As can be seen from Table G.6, there is minimal systematic error (i.e. mean error) in the percentage power reduction, and the standard deviation is only 1.2%. Thus comparisons versus percentage power reduction should not be affected significantly by the absolute error in the University of Michigan dynamic power analysis.

Table G.4 Accuracy of University of Michigan static timing and power analysis versus Design Compiler on the power minimized TILOS sized netlists at $1.0 \times T_{\min}$. Leakage and dynamic are abbreviated as Leak. and Dyn. respectively.

Netlist	University of Michigan			Design Compiler & my analysis			University of Michigan error		
	Delay (ns)	Power (mW)		Delay (ns)	Power (mW)		Delay	Power	
		Leak.	Dyn.		Leak.	Dyn.		Leak.	Dyn.
c432	0.745	0.0555	0.1975	0.2530	0.747	0.0555	0.2194	0.2750	-0.4% 0.0% -10.0% -8.0%
c880	0.800	0.0888	0.3412	0.4300	0.803	0.0889	0.3540	0.4429	-0.5% 0.0% -3.6% -2.9%
c1355	0.786	0.2463	0.8101	1.0564	0.790	0.2464	0.8570	1.1035	-0.5% 0.0% -5.5% -4.3%
c1908	1.146	0.1433	0.5036	0.6470	1.151	0.1434	0.5139	0.6573	-0.5% 0.0% -2.0% -1.6%
c2670	0.742	0.1565	0.6642	0.8207	0.745	0.1580	0.6748	0.8328	-0.4% -1.0% -1.6% -1.5%
c3540	1.177	0.2007	0.7154	0.9161	1.181	0.2008	0.7053	0.9061	-0.3% -0.1% 1.4% 1.1%
c5315	1.038	0.3117	1.3121	1.6238	1.044	0.3118	1.2758	1.5876	-0.6% 0.0% 2.8% 2.3%
c7552	1.388	0.2623	1.1273	1.3896	1.393	0.2624	1.0709	1.3333	-0.3% 0.0% 5.3% 4.2%
Huffman	1.391	0.0828	0.3736	0.4565	1.395	0.0829	0.3872	0.4701	-0.3% 0.0% -3.5% -2.9%
Mean error									
Standard deviation									

Table G.5 Accuracy of University of Michigan static timing and power analysis versus Design Compiler on the power minimized TILOS sized netlists at $1.1 \times T_{\min}$. Leakage and dynamic are abbreviated as Leak. and Dyn. respectively.

Netlist	University of Michigan			Design Compiler & my analysis			University of Michigan error		
	Delay (ns)	Power (mW)		Delay (ns)	Power (mW)		Delay	Power	
		Leak.	Dyn.		Leak.	Dyn.		Leak.	Dyn.
c432	0.812	0.0434	0.1547	0.1981	0.815	0.0434	0.1733	0.2167	-0.3% 0.0% -10.7% -8.6%
c880	0.869	0.0606	0.2497	0.3103	0.871	0.0606	0.2568	0.3175	-0.3% 0.0% -2.8% -2.3%
c1355	0.856	0.1626	0.5367	0.6993	0.859	0.1627	0.5940	0.7567	-0.3% 0.0% -9.6% -7.6%
c1908	1.247	0.0890	0.3389	0.4279	1.249	0.0890	0.3516	0.4407	-0.2% 0.0% -3.6% -2.9%
c2670	0.807	0.1176	0.5407	0.6583	0.809	0.1191	0.5296	0.6486	-0.3% -1.3% 2.1% 1.5%
c3540	1.281	0.1612	0.5695	0.7307	1.285	0.1613	0.5625	0.7238	-0.3% 0.0% 1.2% 1.0%
c5315	1.128	0.2186	1.0168	1.2353	1.131	0.2186	0.9904	1.2090	-0.2% 0.0% 2.7% 2.2%
c7552	1.509	0.2324	0.9827	1.2151	1.512	0.2325	0.9327	1.1653	-0.1% 0.0% 5.4% 4.3%
Huffman	1.516	0.0681	0.3097	0.3778	1.520	0.0681	0.3215	0.3896	-0.3% 0.0% -3.7% -3.0%
Mean error									
Standard deviation									

Table G.6 Error in percentage power reduction for University of Michigan analysis of total power going from a delay constraint of $1.0 \times T_{\min}$ to $1.1 \times T_{\min}$ from the data in Table G.4 and Table G.5.

University of Michigan			Design Compiler & my analysis			University of Michigan error
Total Power (mW)		Relative Power Reduction	Total Power (mW)		Relative Power Reduction	
1.0xTmin	1.1xTmin		1.0xTmin	1.1xTmin		
0.2530	0.1981	21.7%	0.2750	0.2167	21.2%	0.5%
0.4300	0.3103	27.8%	0.4429	0.3175	28.3%	-0.5%
1.0564	0.6993	33.8%	1.1035	0.7567	31.4%	2.4%
0.6470	0.4279	33.9%	0.6573	0.4407	33.0%	0.9%
0.8207	0.6583	19.8%	0.8328	0.6486	22.1%	-2.3%
0.9161	0.7307	20.2%	0.9061	0.7238	20.1%	0.1%
1.6238	1.2353	23.9%	1.5876	1.2090	23.8%	0.1%
1.3896	1.2151	12.6%	1.3333	1.1653	12.6%	-0.1%
0.4565	0.3778	17.2%	0.4701	0.3896	17.1%	0.1%
						Mean error
						Standard deviation

Table G.7 Total power consumption as measured by the University of Michigan power analysis tools for the TILOS-like sizer netlists for delay constraints of $1.0 \times T_{\min}$, $1.1 \times T_{\min}$, $1.2 \times T_{\min}$, and $1.5 \times T_{\min}$. The fraction of power versus the $1.0 \times T_{\min}$ TILOS-like sizer results is listed. From Table G.6, error in the relative numbers may be estimated as 1%.

Netlist	University of Michigan analysis				TILOS sizer		
	Power (mW) at delay constraint of				Fraction of 1.0xTmin Power		
	1.0xTmin	1.1xTmin	1.2xTmin	1.5xTmin	1.1xTmin	1.2xTmin	1.5xTmin
c432	0.2530	0.1981	0.1581	0.1055	0.783	0.625	0.417
c880	0.4300	0.3103	0.2704	0.2004	0.722	0.629	0.466
c1355	1.0564	0.6993	0.5293	0.3545	0.662	0.501	0.336
c1908	0.6470	0.4279	0.3487	0.2426	0.661	0.539	0.375
c2670	0.8207	0.6583	0.5895	0.4580	0.802	0.718	0.558
c3540	0.9161	0.7307	0.6081	0.4413	0.798	0.664	0.482
c5315	1.6238	1.2353	1.0856	0.8326	0.761	0.669	0.513
c7552	1.3896	1.2151	1.0926	0.8922	0.874	0.786	0.642
Huffman	0.4565	0.3778	0.3324	0.2583	0.828	0.728	0.566

Table G.8 Total power consumption as measured by Design Compiler power analysis for the TILOS-like sizer netlists at a delay constraints of $1.0 \times T_{\min}$, and for the linear programming optimized netlists at $1.1 \times T_{\min}$, $1.2 \times T_{\min}$, and $1.5 \times T_{\min}$. The fraction of power versus the $1.0 \times T_{\min}$ TILOS-like sizer results is listed. The $1.0 \times T_{\min}$ TILOS netlist was the starting point for LP power minimization. The $1.1 \times T_{\min}$ delay constraints were set to be identical to the measured delay on the $1.1 \times T_{\min}$ TILOS netlists. The $1.2 \times T_{\min}$ and $1.5 \times T_{\min}$ delay constraints were set as 1.2 and 1.5 times the delay of the $1.0 \times T_{\min}$ TILOS netlists, as TILOS netlists at these larger delay constraints were not available. These LP optimization results are for the conservative slew analysis setting, as that provided better results than the less conservative “aggressive” slew analysis.

Netlist	Design Compiler analysis				Fraction of $1.0 \times T_{\min}$ Power		
	TILOS	Linear programming sizing			$1.1 \times T_{\min}$	$1.2 \times T_{\min}$	$1.5 \times T_{\min}$
		Power (mW) at delay constraint of	$1.0 \times T_{\min}$	$1.1 \times T_{\min}$	$1.2 \times T_{\min}$	$1.5 \times T_{\min}$	
c432	0.2750	0.1699	0.1374	0.0841	0.618	0.500	0.306
c880	0.4429	0.2841	0.2282	0.1608	0.641	0.515	0.363
c1355	1.1035	0.6212	0.4449	0.2780	0.563	0.403	0.252
c1908	0.6573	0.3905	0.3050	0.1966	0.594	0.464	0.299
c2670	0.8328	0.5589	0.4738	0.3588	0.671	0.569	0.431
c3540	0.9061	0.6366	0.5147	0.3529	0.703	0.568	0.390
c5315	1.5876	1.0622	0.8908	0.6579	0.669	0.561	0.414
c7552	1.3333	0.9593	0.8450	0.6691	0.719	0.634	0.502
Huffman	0.4701	0.3328	0.2798	0.2076	0.708	0.595	0.442

G.4.2 Linear programming sizing results versus TILOS-like optimizer

The linear programming optimizer consistently saves power versus the TILOS-like optimizer, with average power savings of 14.4% at $1.1 \times T_{\min}$, 17.8% at $1.2 \times T_{\min}$, and 22.1% at $1.5 \times T_{\min}$. As delay constraints are relaxed and more slack is available to distribute in the circuit to reduce power, the LP optimization approach provides more benefit. The power savings for the linear programming approach versus TILOS are listed in Table G.9 from comparison of the relative power savings in Table G.7 and Table G.8.

While these results versus the TILOS sizer are good, the real test is versus a commercial synthesis tool. The next section compares our results versus Design Compiler, which performs gate sizing based on a TILOS sizing approach, but no doubt has additional tweaks to improve results.

Table G.9 Relative power reduction for the linear programming optimizer versus the TILOS-like optimizer. Minimum, average, and maximum power savings are summarized at the bottom of the table. The LP gate sizing results are consistently better than the TILOS approach, and provide significant power savings. Circuit statistics such as the minimum delay achieved by gate sizing with TILOS (T_{\min}), the number of logic levels, the numbers of inputs and outputs, the number of gates, and the number of edges between gates in the circuit are also listed.

Netlist	# logic levels	# inputs	# outputs	# gates	# edges	Minimum Delay (ns)	LP power reduction from TILOS		
	1.1xTmin	1.2xTmin	1.5xTmin						
c432	23	36	7	166	328	0.7445	21.1%	20.0%	26.6%
c880	26	60	26	390	746	0.7996	11.1%	18.1%	22.1%
c1355	28	41	32	558	1,028	0.7862	15.0%	19.5%	24.9%
c1908	40	33	25	432	800	1.1455	10.2%	13.9%	20.2%
c2670	25	234	140	964	1,529	0.7418	16.3%	20.8%	22.8%
c3540	42	50	22	962	1,932	1.1766	11.9%	14.4%	19.1%
c5315	41	178	99	1,627	3,034	1.0380	12.0%	16.1%	19.2%
c7552	44	207	93	1,994	3,645	1.3882	17.7%	19.4%	21.8%
Huffman	47	79	42	509	865	1.3908	14.5%	18.3%	22.0%
						Minimum	10.2%	13.9%	19.1%
						Average	14.4%	17.8%	22.1%
						Maximum	21.1%	20.8%	26.6%

G.5 Linear programming sizing results with different parameter settings

In the course of developing the linear programming optimization approach, a variety of different parameter settings were tried to get the best possible results versus Design Compiler. The results in this section are with the PowerArc characterized 0.13um library with Vdd=1.2V and Vth=0.23V. Switching activity was not reduced, so leakage is only about 0.1% of total power at this high threshold voltage at 25°C. The library consisted of nine inverter sizes: XL, X1, X2, X3, X4, X8, X12, X16, X20; and four sizes for NAND2, NAND3, NOR2 and NOR3: XL, X1, X2, X4. Later comparison with additional larger cell sizes gave very similar results, as the larger cell sizes seldom appear in the power minimized netlists, particularly as the output port load capacitances are small, and to minimize delay Design Compiler had inserted buffers to reduce large fanout loading. Table G.10 and Table G.11 summarize results for the parameter settings that produced the best results.

In Table G.10 and Table G.11, “old LP settings” refers to the parameter settings used to produce the results in [39]. The settings were: doAlternates=true to consider all alternate cells when

determining β for slew analysis; `doCurrentConditionsOnly=false` to consider all possible input slew and load capacitance values when determining β for slew analysis; and `compareTotalTimingArcDeltaDelays=false` to use worst case delay change on a timing arc when picking the best alternate cell. In addition, for the delay reduction phase, τ was 0.99 and k was 0.01 for the optimization objective in Equation (4.21). If delay reduction failed to satisfy the delay constraint, τ was set to 0.98 to increase the delay reduction and k was set to 0.001 to reduce the weight on power in the delay reduction objective. Optimization with the “old LP settings” was not run on the Huffman, SOVA_EPR4, or R4_SOVA benchmarks.

A more sophisticated approach was adopted to try different parameter settings between optimization iterations if progress was insufficient. The “conservative” and “aggressive” columns in Table G.10 and Table G.11 refer to these settings for slew analysis detailed in Appendix G.1. In addition, a value for τ of 0.995 was tried with the conservative slew analysis setting, to see if power savings might improve by only just meeting the delay constraint (i.e. 0.5% below it) in the delay reduction phase – on average, this did not improve the results. As can be seen from Table G.10 and Table G.11, the “conservative” slew analysis parameter setting produced the best results on average, within 0.3% of the minimum found over the different parameter settings – as is the nature of heuristic approaches, there were some cases where it was a little suboptimal, up to 1.8% worse than the minimum found. Optimization with the “aggressive” slew analysis setting generally does reasonably well, though not as well as the conservative setting, but for c432 at the $1.1 \times T_{\min}$ delay constraint it was worse by 32%, due to the delay reduction phase failing to satisfy the delay constraint as slew analysis was not sufficiently conservative.

Note that the switching activity for the “old LP settings” results was multiplied by $1/\text{critical_path_delay}$ ($1/T$ in equations (E.12) and (E.13)), and the critical path delay of the best result from optimization is always less than or equal to the delay constraint. For the newer results, the delay constraint was used, so dynamic power can be marginally less for the same netlist. For

example, the results for c17 at $1.2 \times T_{\min}$ in Table G.11 are all for the same resulting netlist, but the switching activity was 1% less due to using a delay constraint of 0.1124ns instead of the measured critical path delay which was 0.1113ns.

For comparison, Table G.12 examines the impact of different versions of the COIN-OR CLP [66] linear programming solver. Whatever small changes occurred in the solution provided by the LP solver, from the 2004/11/27 code to the 2005/07/23 software, result in up to $\pm 1.3\%$ change in the minimum power found with the linear programming optimization flow. On average the impact is small, 0.3% worse power at $1.1 \times T_{\min}$ and 0.1% lower power at $1.2 \times T_{\min}$. This demonstrates the difficulty in choosing parameter settings that achieve the minimum power always. Getting within 0.4% and 0.3% on average with the conservative approach at $1.1 \times T_{\min}$ and $1.2 \times T_{\min}$ respectively is a good result.

Table G.10 Linear programming optimization results with different slew analysis settings. The conservative slew analysis approach produces the best results across these benchmarks. The delay constraint was $1.1 \times T_{\min}$, where T_{\min} is the delay of the Design Compiler delay minimized netlist that is the starting point for optimization.

Netlist	Power (mW)				
	old LP settings	conservative	$\tau = 0.995$	aggressive	minimum found by LP
c17	1.079	0.957	0.957	0.957	0.957
c432	2.249	2.234	2.210	2.918	2.210
c499	4.618	4.586	4.640	4.643	4.586
c880	3.491	3.491	3.451	3.477	3.451
c1355	5.535	5.461	5.596	5.421	5.421
c1908	3.113	3.084	3.222	3.112	3.084
c2670	8.628	8.423	8.530	8.504	8.423
c3540	5.789	5.844	5.892	5.848	5.789
c5315	9.508	9.485	9.577	9.506	9.485
c6288	6.066	6.095	6.114	6.094	6.066
c7552	16.652	16.663	16.809	16.757	16.652
Huffman	didn't run	4.809	5.376	4.946	4.809
SOVA_EPR4	didn't run	15.861	15.881	15.821	15.821
R4_SOVA	didn't run	21.861	21.810	22.006	21.810
<i>Power worse than minimum by</i>					
	12.8%	0.0%	0.0%	0.0%	
	1.8%	1.1%	0.0%	32.0%	
	0.7%	0.0%	1.2%	1.2%	
	1.2%	1.2%	0.0%	0.8%	
	2.1%	0.7%	3.2%	0.0%	
	1.0%	0.0%	4.5%	0.9%	
	2.4%	0.0%	1.3%	1.0%	
	0.0%	0.9%	1.8%	1.0%	
	0.2%	0.0%	1.0%	0.2%	
	0.0%	0.5%	0.8%	0.5%	
	0.0%	0.1%	0.9%	0.6%	
		0.0%	11.8%	2.9%	
		0.2%	0.4%	0.0%	
Average	2.0%	0.4%	2.1%	3.2%	
Maximum	12.8%	1.2%	11.8%	32.0%	

Table G.11 Linear programming optimization results with different slew analysis settings. The conservative slew analysis approach produces the best results across these benchmarks. The delay constraint was $1.2 \times T_{\min}$, where T_{\min} is the delay of the Design Compiler delay minimized netlist that is the starting point for optimization.

Netlist	Power (mW)				
	old LP settings	conservative	$\tau = 0.995$	aggressive	minimum found by LP
c17	0.763	0.756	0.756	0.756	0.756
c432	1.765	1.737	1.755	1.755	1.737
c499	3.761	3.734	3.749	3.737	3.734
c880	2.612	2.606	2.600	2.618	2.600
c1355	4.123	4.132	4.195	4.137	4.123
c1908	2.443	2.414	2.495	2.404	2.404
c2670	6.902	6.994	6.870	6.913	6.870
c3540	4.703	4.637	4.649	4.860	4.637
c5315	7.811	7.813	7.850	7.864	7.811
c6288	4.781	4.730	4.690	4.812	4.690
c7552	13.631	13.538	13.438	13.596	13.438
Huffman	didn't run	3.720	3.740	3.734	3.720
SOVA_EPR4	didn't run	13.905	13.894	13.894	13.894
R4_SOVA	didn't run	19.213	19.162	19.256	19.162
<i>Power worse than minimum by</i>					
Average	1.0%	0.0%	0.0%	0.0%	
Maximum	1.9%	1.8%	4.8%	3.8%	

Table G.12 Comparison of results at a delay constraint of $1.1 \times T_{\min}$ and $1.2 \times T_{\min}$, with the new and old COIN-OR CLP [66] linear programming solver software.

Netlist	Minimum Power (mW) at 1.1xTmin		1.1xTmin New code vs. old	Minimum Power (mW) at 1.2xTmin		1.2xTmin New code vs. old	
	2005/07/23 code	2004/11/27 code		2005/07/23 code	2004/11/27 code		
c17	0.957	0.957	0.0%	0.756	0.756	0.0%	
c432	2.204	2.234	1.3%	1.737	1.737	0.0%	
c499	4.589	4.586	-0.1%	3.711	3.734	0.6%	
c880	3.491	3.491	0.0%	2.607	2.606	0.0%	
c1355	5.525	5.461	-1.2%	4.117	4.132	0.4%	
c1908	3.115	3.084	-1.0%	2.415	2.414	0.0%	
c2670	8.484	8.423	-0.7%	6.946	6.994	0.7%	
c3540	5.883	5.844	-0.7%	4.620	4.637	0.4%	
c5315	9.495	9.485	-0.1%	7.819	7.813	-0.1%	
c6288	6.100	6.095	-0.1%	4.739	4.730	-0.2%	
c7552	16.778	16.663	-0.7%	13.565	13.538	-0.2%	
Huffman	4.865	4.809	-1.2%	3.707	3.720	0.4%	
SOVA_EPR4	15.857	15.861	0.0%	13.906	13.905	0.0%	
R4_SOVA	21.862	21.861	0.0%	19.201	19.213	0.1%	
		Minimum	-1.2%			Minimum	-0.2%
		Average	-0.3%			Average	0.1%
		Maximum	1.3%			Maximum	0.7%

Appendix H Survey of previous research on optimization with multi-Vdd and multi-Vth

Summaries of previous research on algorithms for optimization with multi-Vdd and multi-Vth are presented in this appendix. Where possible, theoretical runtime complexity for algorithms has been stated, but it is more useful to analyze actual runtime data where that is available.

H.1 Papers on optimization with multi-Vth

Some of the better algorithmic research on assignment of multiple threshold voltages is summarized below.

H.1.1 Wang and Vrudhula: transistor-level dual Vth assignment (1998) [229][230]

Wang and Vrudhula examined several algorithms for transistor-level threshold voltage assignment from a choice of two threshold voltages [230]. More details were provided in [229]. Transistors in a gate that are driven by the same signal were assigned the same threshold voltage (low or high Vth), and thus dual Vth assignment was posed as an edge-weighted partitioning problem. They assumed fixed gate sizes, with the circuit having been sized optimally beforehand – presumably for minimum power. The process technology had a transistor channel length of 0.46um and supply voltage of 1.0V. Circuits with the low threshold voltage had 68% of the delay of circuits with all transistors at high Vth and 22 times the leakage. Low Vth PMOS gate input capacitance was 5.5% more than high Vth, and low Vth NMOS gate input capacitance was 3.3% more than high Vth [229], compared to a capacitance increase of 7% to 20% depending on supply voltage for our libraries (as discussed in Section 5.2.2). Their delay data did include signal slew and separate rise and fall delays, but the maximum of the rise and fall delay was used on each timing arc in the algorithmic formulation. This is quite suboptimal as rise and fall delays were unbalanced – for example, 1.62:1 for the ratio of rise delay to fall delay for a high Vth inverter, but only a ratio of 1.16:1 for the low Vth inverter [229].

Three different heuristics were used for Vth assignment, and the one that produced the best results and also had the fastest runtimes will be discussed here. The starting point was a circuit sized with all gates at high Vth. The delay constraint was the critical path delay in the low Vth circuit. On subcircuits with negative slack all the transistors are changed to low Vth. Then the initial solution is found by iteratively picking the topological level cut (a level k cut has all gates of topological level more than k in one set, and the others in the second set) of the edges that can feasibly have their driven transistors assigned to high Vth which gives the greatest reduction in leakage power. This results in a set of edges driving high Vth transistors, and a set of edges driving low Vth transistors, with no more edges that feasibly (i.e. not violating a delay constraint) can be set to low Vth. Then the edge in the high Vth set with least power reduction is swapped out to low Vth, low Vth edges that can now be feasibly set to high Vth identified, and low Vth edges that can be feasibly set to high Vth are then changed in the order of descending leakage reduction until no more edges can be changed to high Vth. This algorithm gave average leakage power savings of 52% versus all transistors being low Vth.

They state that the theoretical runtime complexity of their algorithm is $O(|E|(|E|+|V|))$, and analysis of their runtimes with the best algorithm show complexity of less than $O(|V|^2)$ (see Figure H.1). Note that $|E|$ is of $O(|V|)$ for the circuits – ranging from $1.02\times$ to $3.89\times|V|$ for their benchmarks. These runtimes are fast, but failing to consider Vth assignment in conjunction with gate sizing makes this approach suboptimal.

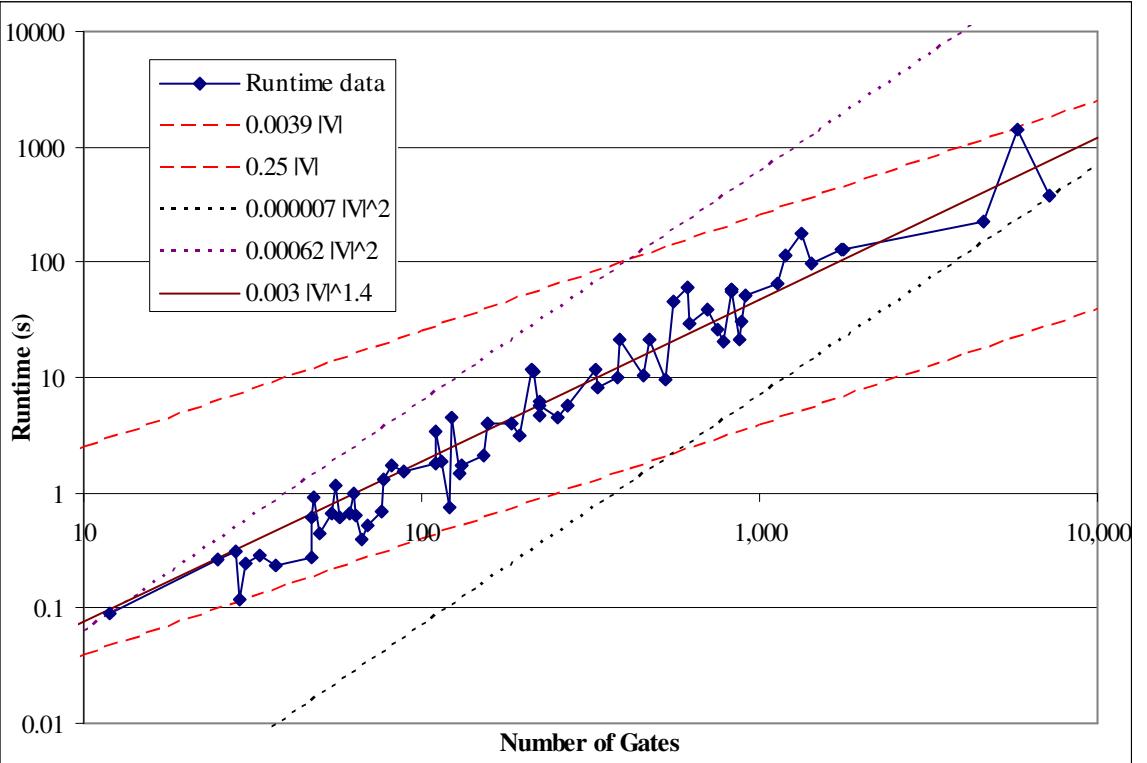


Figure H.1 Runtimes for the transistor-level dual Vth assignment algorithm in [229] on a log-log graph. This is for the algorithm that produced the best results and fastest runtimes. The logarithmic scales on both axes are used to show the wide range of circuit sizes and runtimes without obscuring the runtime behavior. This graph shows that the runtime growth is between linear and quadratic, as shown by the lines on the log-log scale. On average the runtime grows as about $|V|^{1.4}$, where $|V|$ is the number of gates in a circuit.

H.1.2 Wei et al.: gate-level dual Vth assignment and choice of second Vth (1998) [231]

The approach by Wei et al. performed gate level assignment to high Vth from an initial configuration with all gates at low Vth. It is not clear what sizing approach they used to generate the initial circuits. They used breadth first search from the primary outputs, setting gates with sufficient slack to high Vth to reduce the leakage. An outer loop swept different values of high Vth, to find the optimal value of high Vth. The process was a 0.5um MOSIS process with Vdd of 1V and low Vth of 0.2V. This greedy approach prioritized assigning gates near the output to high Vth, which is suboptimal versus considering earlier gates in the circuit as well. They achieved an average leakage power reduction of 48%. With this reduction in leakage power, they report up to 13% savings in total power for one circuit at a tight delay constraint [231].

A linear Elmore delay model was used, averaging the rise and fall delays to determine the gate delay. The runtime complexity appears to be $O(|V|)$, as slew is ignored, and the gate delay can be updated easily as the circuit is traversed in reverse topological order. Wire loads were not included. With slew analysis the complexity would be $O(|V|^2)$.

H.1.3 Sundararajan and Parhi: Dual V_{th} assignment and choice of high V_{th} (1999) [197]

In 1999, Sundararajan and Parhi proposed an integer linear program (ILP) for dual V_{th} assignment. This approach used a very simplistic delay model where gates have a small integer number of unit delays, and a retiming formulation is used to retime additional slack unit delays to minimize power. The starting point for optimization was a netlist with all low V_{th} gates. Delay constraints were encoded in the ILP, but there were no separate rise and fall delays, and delays were independent of both load capacitance and input slew [197].

In the same year, they also proposed an identical approach for Vdd assignment. The dual Vdd assignment approach just used a different power model and included level converter delays of either zero or one unit, and the starting point for optimization was a netlist with all high Vdd gates [198].

Their ILP formulation bears some similarity to our approach – there is a variable to determine if a gate is assigned to high V_{th} (or VDDL), which is essentially the same as the cell choice variable ($0 \leq \gamma_u \leq 1$) that we use. They solve the ILP using a linear program solver, using thresholds ($\gamma_u = 1$) to determine which gates are assigned to high V_{th} (or VDDL), and iterate until no more gates are assigned to high V_{th} (or VDDL). Solving the ILP directly with an integer linear programming solver would be too computationally slow.

With dual Vdd assignment, they then did a post-pass with the ECVS algorithm described in Appendix H.2.2 to see if any additional gates with $\gamma_u > 0$ can be assigned to VDDL [198]. To find

the best value for high V_{th}, they ran the optimization for different high V_{th} values and picked the best one [197]. With dual V_{th}, they achieved 34% power savings on average versus the approach in [231].

Given the completely inaccurate delay analysis, these two papers ([197] and [198]) are only noteworthy as the first attempt at a linear programming formulation for dual V_{th} or dual V_{dd} assignment. The retiming formulation with assigning units of slack is cumbersome and unnecessary. They did not consider how to encode realistic gate delays and changes in gate delay in the ILP constraints.

H.1.4 Wei et al.: transistor-level dual V_{th} assignment (1999) [232]

In 1999, Wei et al. proposed several approaches to transistor-level dual V_{th} assignment. The approach that gave the best results, the “priority selection” algorithm, works in a very similar manner to TILOS, but starts with a low V_{th} netlist that satisfies the delay constraint, proceeding to reduce leakage power subject to the delay constraint. To prioritize assigning transistors to high V_{th}, they use the metric

$$Sensitivity_{reduce\ power} = -\frac{\Delta P_{leakage}}{\Delta d} \quad (\text{H.1})$$

where $\Delta P_{leakage} < 0$ is the change in leakage if the PMOS or NMOS transistor is changed to high V_{th}, and $\Delta d > 0$ is the corresponding increase in rise or fall delay. The lower the value of this metric the better. Transistors are iteratively assigned to high V_{th} in ascending order of the metric, and then static timing analysis is updated. They use a linear model of delay versus load capacitance. Signal slew is not considered. They consider rise and fall delays separately [232].

Their technology had an effective transistor channel length of 0.32um, supply voltage of 1.0V, threshold voltages of 0.2V and 0.3V. They assumed 110°C for Hspice simulations to determine delay and leakage [232].

Wei et al. examined gate-level V_{th} assignment (all transistors are the same V_{th} – either high or low V_{th}), versus stack-level V_{th} assignment (all NMOS transistors have the same V_{th}, all PMOS transistors have the same V_{th}), versus allowing mixed V_{th} except for transistors in series (“series-level”) that must be the same V_{th} (e.g. parallel PMOS transistors in a NAND2 can have differing V_{th}). Comparing their results, stack-level V_{th} assignment provided on average 15% leakage power savings versus gate-level V_{th} assignment, and series-level V_{th} assignment gave 25% savings compared to gate-level V_{th} assignment. Series-level dual V_{th} assignment gave 61% leakage reduction versus all gates being low V_{th} [232].

They also proposed a faster approach that splits the transistors into groups based on the sensitivity. Transistors in a group are considered in reverse topological order, and those with sufficient slack are changed to high V_{th}, then delays are updated. With ten groups, the solution was 3.6% worse on average than the “priority selection” algorithm, but runtimes were two orders of magnitude faster. The runtime complexity of this approach is $O(m|V|)$, where m is the number of groups. However, this implicitly relies on the fact that slews are ignored, so the delay impact can just be propagated in reverse topological order only needing to consider the change in load capacitance. Including slew analysis would increase the complexity to $O(|V|^2)$.

H.1.5 Sirichotiyakul et al.: transistor-level dual V_{th} assignment with transistor sizing (1999) [183]

Sirichotiyakul et al. considered combining dual threshold voltages with transistor sizing to reduce leakage power. In the 0.25um process with supply voltage of 0.9V, the gates with the low threshold voltage had only 53% of the delay of gates at high V_{th}, but 33 times the leakage [183]. We compare use of multi-V_{th} at a tight delay constraint with Vdd of 1.2V, where low V_{th} of 0.08V only gives a 18% delay reduction versus a high V_{th} of 0.23V, but increases the leakage by 52x (see Table B.6). Consequently, their results give substantially better leakage power/delay trade-offs. They note that the transistor gate capacitance increases 8% to 10% as V_{th} is reduced

[183], double that reported for the process in [229] and similar to the increase of 7% to 20% depending on supply voltage for our libraries.

To calculate the leakage they consider the dominant leakage states of a gate. For example where there is a leakage current path with only one transistor off in the path. In standard cell libraries that I have examined, only the leakage states with larger leakage are listed for gates with three or more inputs (i.e. the dominant leakage states), and a default leakage is specified for other cases.

The initial circuit is all high V_{th}. They iteratively set transistors to low V_{th}, greedily choosing the transistor that gives the best delay reduction/leakage_increase trade-off, determined in the manner of Equation (4.3) by

$$Sensitivity_{reduce\ delay} = -\frac{1}{\Delta I_{leakage}} \sum_{l \in timing_arcs} \frac{\Delta d_l}{Slack_l - Slack_{min} + k} \quad (H.2)$$

where $\Delta I_{leakage} > 0$ is the increase in leakage if a transistor is changed to low V_{th}, Δd_l is the change in delay on timing arc l which has timing slack $Slack_l$. $Slack_{min}$ is the minimum timing slack in the circuit, and k is a small constant added for numerical stability. This approach is like TILOS [64], which was used for transistor sizing.

To achieve additional delay reduction, when a transistor is assigned to low V_{th}, transistors in neighboring gates over three levels of logic are resized. This neighborhood gave as good results as considering a wider range of influence when resizing – similar to the observation in Section 4.4.1 that for delay accuracy limiting analysis to the fanins, the gate itself, and the immediate fanouts (encapsulated by change in delay of the gate and transitive fanout slew impact) is sufficient. Compared to gate sizing with all gates at low V_{th}, the dual V_{th} approach reduced leakage by 3.1× to 6.2× with at most 1.3% delay penalty [183]. The circuit sizer was a TILOS-like sizer which provides a good baseline, but, as noted in Section 7.3, large savings can be

achieved versus an all low V_{th} configuration. They did not consider the impact on dynamic power and the total circuit power consumption.

Only one runtime is stated. Analysis of the algorithm indicates a theoretical complexity of $O(|V|^2)$, as iteratively the gate with maximum sensitivity is picked, the neighborhood of three logic levels are resized, and then static timing analysis must be updated.

H.1.6 Wei, Roy and Koh: gate-level dual V_{th} assignment with gate sizing (2000) [233]

Wei, Roy and Koh used a TILOS-like algorithm for gate-level power minimization with dual V_{th} and gate sizing. Initially all gates are minimum size and at high V_{th}. They computed the TILOS sensitivity metric for delay change (Δd) versus power_increase ($\Delta P > 0$) for gate upsizing and reducing the threshold voltage separately, as $-\Delta d/\Delta P$. The algorithm proceeded in the manner of TILOS, iteratively finding the gate with maximum sensitivity on the most critical path and changing that gate until delay constraints are met [233].

Power analysis included dynamic power, short-circuit power and leakage. Larger short circuit power occurs when the input transitions slowly (larger input slew) compared to the output transition time. They developed a short-circuit current model which took into account input slew, load capacitance, supply voltage, and both NMOS and PMOS threshold voltages, with error versus Hspice of 19% to 30% [233]. Delay analysis used a Sakurai-Newton alpha-power law delay model [173]. It is not clear from the paper, but they appear not to have used separate rise and fall delays. It would have been more accurate to use lookup tables for delay and short circuit power versus load capacitance and input slew. They used 0.25um MOSIS process technology with a supply voltage of 1V and threshold voltages of 0.2V and 0.3V.

They compared results for sizing with low V_{th} at the minimum achievable delay versus sizing with dual threshold voltages. At 0.1 switching activity, total power was reduced by 14% using

dual V_{th} and gate sizing. At 0.03 switching activity, total power was reduced by 24% using dual V_{th} and gate sizing [233]. This paper presented results versus a good sizing algorithm (TILOS), though power savings versus all gates being low V_{th} will be greater than if compared to an all high V_{th} circuit.

The theoretical worst case runtime complexity of this approach is the same as TILOS, $O(|V|^2)$.

H.1.7 Pant, Roy and Chatterjee: gate-level dual V_{th} assignment with gate sizing and choice of single V_{dd} (2001) [162]

Pant, Roy and Chatterjee looked at dual threshold voltage assignment in addition to gate sizing and optimal choice of a single supply voltage [162]. Gate delay versus input slew was formulated as a linear function, and from this the transitive fanout delay impact of slew was calculated. Separate rise and fall delays were not considered. The circuit was first gate sized to minimize power with all gates high V_{th}. Then an iterative algorithm assigned gates on the most critical path to low V_{th}, greedily picking the gate that gave the largest path delay reduction (i.e. including slew impact). After a user-specified number of gates were assigned to low V_{th}, gates were then sized and the optimal supply voltage chosen using the approach in [161]. The single (critical) path heuristic to choose which gate to assign to low V_{th} suffers the same problem as outlined in Figure 4.2 – this greedy approach chooses the gate with maximum sensitivity on the critical path only, rather than considering the combination of gates on other paths that could be sized instead. Their dual V_{th}/sizing/optimal single V_{dd} approach achieves on average 40% total power savings on the ISCAS'89 sequential benchmarks. However, the high V_{th}/sizing/optimal choice of V_{dd} baseline [161] that they compare is stated to be suboptimal in [162] and has not been compared versus a traditional sizing approach such as TILOS. The sizing algorithm heuristically assigns a delay budget to each gate before assigning gate size et al. to meet that delay, but signal slew is not considered [161]. They do not specify the process technology or the range of available gate sizes.

Sweeping the number of gates to set to low V_{th} introduces at least a $\log|V|$ multiplier to the sizing/supply voltage choice runtimes (e.g. assuming repeated bisection of the number of gates $|V|$ that may be assigned to low V_{th}). It would be better to consider sizing and threshold voltage simultaneously, which just increases the computation time by at most a factor of two to determine the best V_{th} (out of high V_{th} and low V_{th}) and size for each gate. Runtimes are not stated, but analysis of the algorithm indicates that it has theoretical complexity $O(|V|^2)$, as iteratively the gate with the maximum delay reduction is picked, then static timing analysis must be updated.

H.1.8 Thompson, Orshansky and Keutzer: gate-level dual V_{th} assignment with gate sizing (2002)

For an EECS244 project, Brandon Thompson looked at applying dynamic programming to gate sizing, dual threshold voltage assignment, and dual supply voltage assignment. Dynamic programming is commonly used to technology map a technology independent netlist to a standard cell library. Dynamic programming provides an optimal solution on a tree, where a tree is a directed graph with no multiple fanouts that converge to a single node. The circuit is partitioned at multiple fanouts in the directed acyclic graph into a “forest of trees” [118]. The input arrival times to a tree and capacitive loads on the tree must be assumed, so a range of discrete bins for each of these is used, and the optimal solution for each point in the resulting grid is found.

The approach was to partition the graph into trees, and assign slack to each tree using a minimum cost network flow linear programming formulation weighted by the power consumption of gates in the tree. For the 0.18um logical effort delay models that ignored slew and did not have separate rise and fall delays, the optimization results from dynamic programming [159] were comparable to those from linear programming in [154] (in some cases worse and in some cases better), and runtimes were about the same. However, it was very difficult with the partitioning required for dynamic programming to produce good results with multiple supply voltages, as supply voltage and either the need for level conversion or restriction to VDDH must also be considered across

partition boundaries. Considering slew would also have made dynamic programming less applicable due to the increased runtime for considering another binned input variable (i.e. input slews) that affects the optimal mapping. Secondly, delay budgeting for a partitioned circuit is notoriously difficult to solve well. This is why TILOS-like sizing approaches are typically used instead of dynamic programming in today's synthesis optimization tools, though dynamic programming can provide a good coarse-grained (due to the bins and budgeting across partitions) starting point with technology mapping.

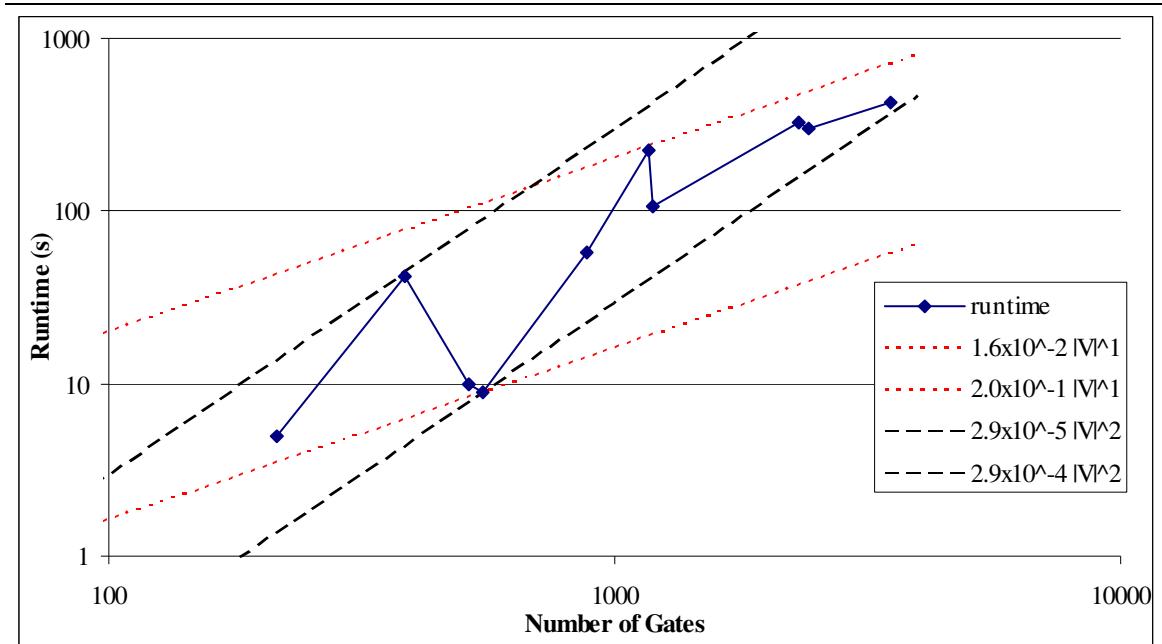


Figure H.2 Runtimes for the gate-level multi-Vth assignment algorithm in [42] on a log-log graph. This graph shows that the runtime growth is between linear and quadratic, as shown by the lines on the log-log scale. The runtime grows quadratically as $|V|^2$, where $|V|$ is the number of gates in a circuit.

H.1.9 Chou, Wang and Chen: gate-level multi-Vth assignment with gate sizing (2005) [42]

Chou, Wang and Chen formulated the gate sizing and threshold assignment problem as a geometric program with convex posynomial models [42]. The fitting error for the posynomial models for delay, dynamic power and leakage power was less than $\pm 10\%$ for 90% of the 0.1um SPICE data, with standard deviation of up to 6.7%. Posynomial models are not sufficiently accurate for optimization of real circuits without reference to more accurate static timing analysis.

They solve the Lagrangian dual of the geometric program primary problem. Solving both the primary problem and Lagrangian dual problem is a common technique used in geometric program solvers such as MOSEK [146] to converge to the solution more quickly. Chou, Wang and Chen solving the Lagrangian dual by iteratively updating the Lagrangian multiplier λ and solving the Lagrangian relaxation sub-problem associated with that λ value [42]. Their algorithm is fast – they claim linear runtime growth on benchmarks from 214 to 3,512 gates. However, analysis of their runtime data suggests that runtime grows quadratically with circuit size (see Figure H.2). Runtimes from larger benchmarks would be needed to verify this.

The allowed range of transistor widths was from 0.2um to 1.1um, and threshold voltages ranged from 0.14V to 0.26V [42]. Vdd was fixed at 1.0V and the switching activity was assumed to be 0.1. The solution with continuous V_{th} values is discretized to V_{th} values of 0.14V, 0.18V, 0.22V and 0.26V. With the threshold voltages fixed, the gates are then resized once more. We make the same assumption in the geometric programming work in Chapter 6 that gates may have a continuous range of sizes, but the range of transistor widths in [42] is much narrower.

Compared to gate sizing along with threshold voltages fixed to a nominal value of 0.18V, they found average power savings of 58% by allowing the threshold voltages to be changed [42]. The additional power saving was only about 2% with more than two threshold voltages.

The most notable contribution of this approach is the fast runtimes for a geometric programming problem. Our geometric programming runtimes with MOSEK grew cubically, and were about 50 \times slower for the largest benchmark that we considered, c499 with 580 gates, compared to problems in [42] with similar numbers of optimization variables. One reason the Lagrangian relaxation approach used by Chou, Wang and Chen scales to larger benchmarks is because they fixed the PMOS to NMOS transistor width ratio and did not consider optimizing Vdd.

H.2 Papers on optimization with multi-Vdd

Clustered voltage scaling (CVS) and extended clustered voltage scaling (ECVS) may refer either to specific algorithms for the supply voltage assignment, or to the general methodology of supply voltage assignment – without or with asynchronous level converters respectively. I will distinguish between the methodology and algorithm where appropriate.

H.2.1 Usami and Horowitz: CVS for dual Vdd assignment and choice of VDDL (1995) [222]

In 1995, Usami and Horowitz proposed clustered voltage scaling (CVS) with voltage level restoration by level converters combined with the latches [222]. Their CVS algorithm proceeded in depth-first search manner from the combinational outputs, assigning VDDH gates to VDDL if they have sufficient slack and only VDDL or level converter latch fanouts. Assigning gates to VDDL was prioritized by the gate load capacitance or the slack in descending order.

In the 0.8um technology for the first benchmark, they achieved 19% power savings with dual supply voltages of 5V and 3V with a library with drive strengths 1X and 4X for inverters, NANDs, NORs, ANDs, ORs, XORs, AOIs, and OAIs. The power savings were only 9% with dual supply voltages of 5V and 4V when using a library with finer grained gate sizes – drive strengths of 0.25X to 4X in 0.25 steps and 8X, 12X and 16X [222]. Note here the significance of having the finer grained library. They achieved 18% power savings on their other benchmark with the finer grained library. Their delay models did not include slew nor separate rise/fall delays, and short circuit power was not included. They did use a wire load model.

The complexity of CVS is $O(|V|^2)$ [125].

H.2.2 Usami et al.: ECVS for dual Vdd assignment and choice of VDDL (1997) [224][225]

Usami et al. proposed the extended clustered voltage scaling (ECVS) algorithm for assigning VDDH gates to VDDL using for level restoration both asynchronous level converters and VDDL

flip-flops that incorporate level converters if they drive VDDH gates [225]. Further details are provided in [224]. The paper also outlines interspersed standard cell row layouts, with each row being either VDDL or VDDH, which allows the additional wire length between VDDH and VDDL gates to be minimized, rather than having separate VDDH and VDDL regions with longer wires between them.

Starting from a sequential netlist with all gates at VDDH, the ECVS algorithm proceeded in reverse topological level order with breadth-first search from the flip-flops backwards. First, each flip-flop was examined and changed to a level converter flip-flop (LCFF) if there is sufficient slack. The clock tree root driver operated at VDDH and the buffers driving the clock signal to the flip-flops had VDDL supply voltage to reduce clock tree power. The delay constraint was relaxed so that all flip-flops could be set to LCFFs [224].

After the flip-flops, the combinational gates were examined in reverse topological order. If a VDDH gate had all VDDL fanouts and there was sufficient timing slack, it was set to VDDL. If the gate had some VDDH fanouts, an asynchronous level converter must be inserted. However, one gate may be insufficient to amortize the power overhead for the level converter. Thus in addition to the reduction in power for the gate changing from VDDH to VDDL, the potential power reduction for changing the gate's fanins to VDDL was estimated to check if inserting the level converter gave overall power savings – if so, the gate was changed to VDDL and the level converter inserted. This reverse topological analysis continued until all gates had been considered for assignment to VDDL.

They fabricated a dual supply voltage media processor chip in 0.3um technology with VDDH of 3.3V. In the design flow, wire loads were generated from back annotated capacitances after placement [225]. It appears from the discussion of slack in the initial circuit, “more than 60% of the paths have delays of half the cycle time” [224], that the initial circuit was not sizing power

minimized, hence larger power savings from dual Vdd would be expected. They achieved on average 47% power savings across seven random logic modules. They determined the low supply voltage, 1.9V, by sweeping VDDL to find the best power savings on the largest random logic module. Assigning flip-flops to VDDL provided the majority of the power savings, due to the high switching activity of the clock signal, which switches from 0 to 1 back to 0 every clock cycle. The clock tree power was reduced by 69%. The ECVS algorithm provided 28% power savings for the combinational logic, and the power overhead for the level converters was 8% of the total power. The area overhead in the dual Vdd modules was 15% due to level converters, additional VDDL power lines, and reduced cell density due to constrained placement on a VDDH row or VDDL row [224].

The theoretical runtime complexity is the same as that of CVS, $O(V^2)$. Runtimes would be somewhat longer due to the analysis when inserting level converters to estimate the additional potential power reduction for changing the gate's fanins to VDDL.

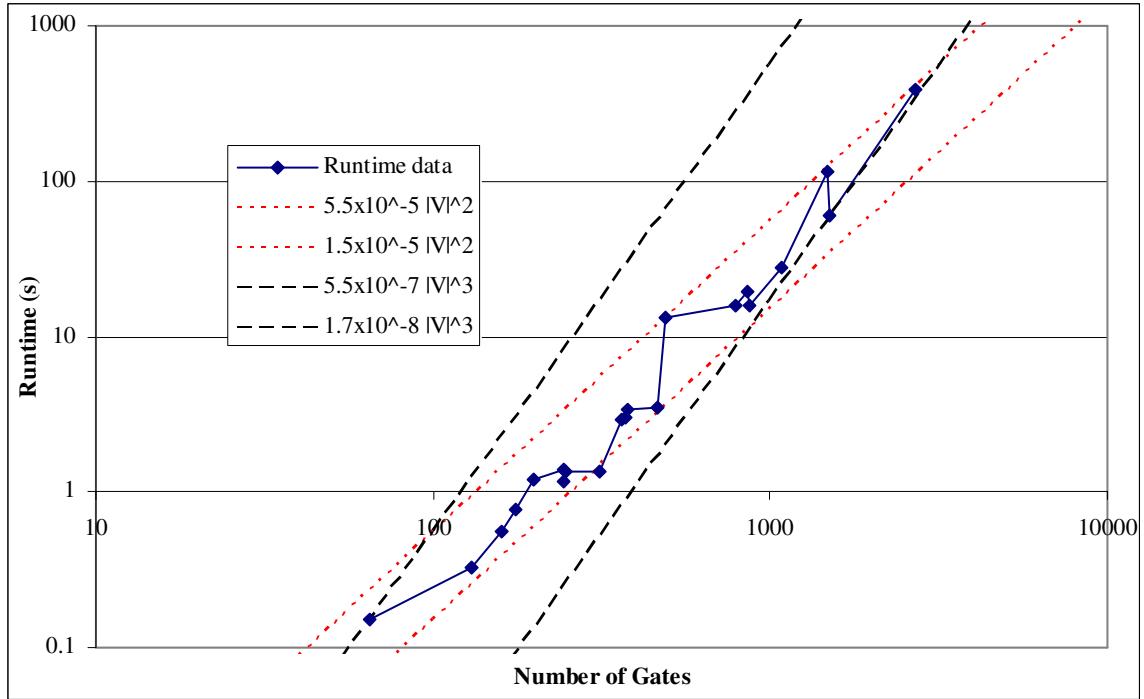


Figure H.3 Runtimes for the dual Vdd assignment algorithm in [35] on a log-log graph. This graph shows that the runtimes for their benchmarks grow quadratically with circuit size, compared to the theoretical worst case complexity of $O(|V|^3)$ from analysis in [34].

H.2.3 Chen and Sarrafzadeh: ECVS methodology for dual Vdd assignment (1999) [34][35]

Chen and Sarrafzadeh reported average power savings of 19.6% with assignment of dual supply voltages and level converters for voltage restoration, without degrading circuit performance in their 1999 paper [35]. Vdd values were 5V and 3.5V in 1um technology with 0.6V threshold voltage [34]. Level converters consumed on average 5% of the total power. This compares to 11.4% power savings achieved by their implementation of CVS [34]. Their algorithm determined a “maximally weighted independent set” of gates in the directed graph to change on each iteration, where each gate in the independent set has no path to another gate in the set. This independent set restriction is a major constraint on the optimization, such that only a single gate on a path may be optimized on each iteration, which results in more iterations required to reach a solution. Choosing the maximally weighted independent set that provides the best power reduction at each iteration may not be optimal, as it may be better to change two gates that are on the same path – that change may not be possible later as slack may be reduced by changing some other gate that

has a (directed) path to the second gate but not the first. Their delay models used only a single delay for each gate (no slew, or separate rise/fall delays). Their power models neglected internal power, and wire loads were also not included.

The theoretical complexity of this approach is $O(|V|^3)$ [34]. For their benchmark circuits, the actual runtimes grow quadratically with circuit size, as shown in Figure H.3.

H.2.4 Yeh et al.: ECVS methodology for dual Vdd assignment, and CVS methodology for dual Vdd assignment with gate sizing (1999) [242]

Yeh et al. also considered using a maximally weighted independent set approach for ECVS. Starting with all gates at VDDH, gates were first assigned to VDDL using the CVS algorithm. Then additional gates that have sufficient slack to be assigned to VDDL were identified, using the maximally weighted independent set of gates based on their power savings going from VDDH to VDDL. This step did not consider the delay impact which may limit assigning further gates to VDDL and it appears that the power overhead for voltage level converters was not considered. Level converters are then inserted. This process was repeated picking the maximally weighted independent set of gates at each step to change to VDDL [242].

Their CVS methodology for dual Vdd used gate sizing provided additional slack to change gates to VDDL. The CVS algorithm was performed on the initial circuit to assign gates to VDDL in reverse topological order from the combinational outputs. Having identified VDDH gates with all VDDL outputs that can't be changed to VDDL due to insufficient slack, the set of gates on critical paths through those VDDH gates was determined. Then those gates on critical paths may be upsized, subject to a user-specified area constraint, to give additional timing slack for further gates to be assigned to VDDL. A `delay_reduction/area_increase` metric for upsizing was used ($-\Delta d/\Delta A$) to weight gates and the minimum weight cut set was determined. Gates in the cut set were upsized, and then more gates may be assigned to VDDL using the CVS algorithm. This procedure of upsizing gates and assignment to VDDL was iterated [242]. The power savings will

be suboptimal, as a power-weighted metric was not used when considering gate upsizing. As such, this approach assumes area is a more important constraint than reducing power, which is no longer true in today's technologies where the transistor density has increased greatly.

The ECVS methodology proposed by Yeh et al. provided average power savings of 12.1% versus savings of 10.3% for CVS without gate sizing. With gate sizing, CVS provided 19.1% power savings. The area overhead for the CVS with gate sizing approach was 6%. Circuits were initially area minimized subject to a delay constraint of 1.2 times the minimum delay. They used a 0.6um library, with inaccurate Elmore delay models. Wire loads and short circuit power were not included. The dual supply voltages were 5V and 4.3V [242].

Theoretical worst case runtime complexity was $O(|V||E|\log(|V|^2/|E|))$ for their ECVS approach and $O(|V||E|^2)$ for the CVS with sizing approach [242]. As noted earlier, $|E|$ is proportional to $|V|$ in today's technologies, so these expressions may be simplified to $O(|V|^2\log|V|)$ and $O(|V|^3)$ respectively.

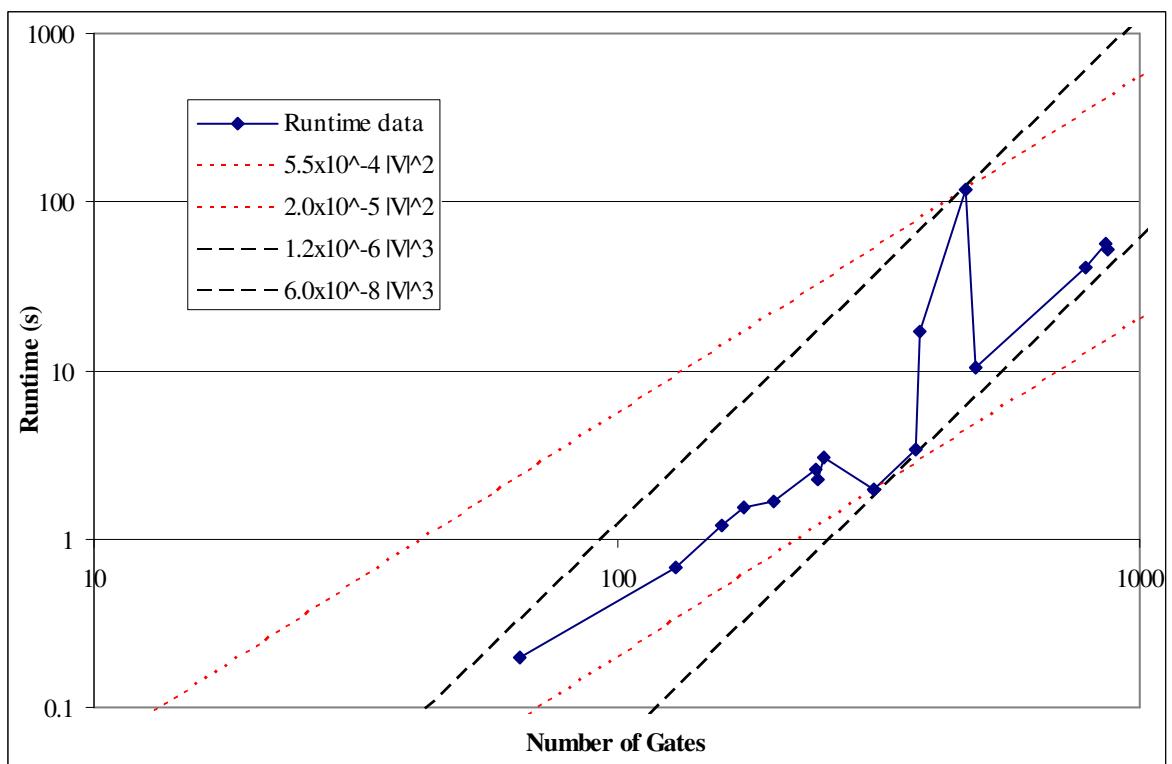


Figure H.4 Runtimes for the dual Vdd assignment and gate sizing in [36] on a log-log graph. This graph shows that the runtimes for their benchmarks grow between quadratically and cubically with circuit size.

H.2.5 Chen and Sarrafzadeh: ECVS methodology for dual Vdd assignment with gate sizing and choice of VDDL (2000) [36]

In [36], Chen and Sarrafzadeh extend the maximally weighted independent set approach to gate sizing as well as dual supply voltages. They report up to 27.4% power savings on average with gate sizing and dual Vdd versus 23.2% with dual Vdd alone, or 6.6% savings with sizing alone. The best supply voltages were 5.0V and 2.8V in 1um technology with 0.6V threshold voltage. Additionally, they report up to 51.2% power savings on average with gate sizing alone using libraries with more gate sizes, but it is not clear how good the initial gate sizing point of comparison is. Simplified logical effort delay models were used for dynamic power analysis with capacitive loads, including wire capacitances, and gate internal capacitances. Short circuit power was not included. Leakage power was not considered, but should be minimal in the 1um technology.

The theoretical worst case runtime complexity of this approach is the same as stated in [34], $O(|V|^3)$. In Figure H.4 we see that there is at least one outlier, ISCAS'85 benchmark c1908, for which this worst case complexity occurs.

H.2.6 Dhillon et al.: Partitioned (ECVS methodology) dual Vdd assignment with choice of VDDH, VDDL, and Vth (2003) [57]

Dhillon et al. proposed a “module-level” Vdd and Vth assignment approach, partitioning a netlist into N modules which may be assigned different Vdd values. They construct a matrix of the paths in the circuit A , such that if \mathbf{d} is the delay vector, then the corresponding vector of path delays (i.e. sum of module delays along each path) is $\mathbf{t} = A\mathbf{d}$. They assume that the starting point for optimization is after sizing power minimization, where there is no slack on any path to an output – this assumption is erroneous in many designs. Then, the conditions for which the gate delays \mathbf{d} can be changed without affecting \mathbf{t} are determined by the nullspace of A . To assign Vdd and Vth to each module, they iteratively follow the steepest descent of energy in the nullspace of A , meeting minimum energy conditions, and ensuring that delay constraints will still be satisfied [57].

A Sakurai-Newton alpha power law delay model [173] in terms of Vdd and Vth was used, fitted to the delay of a module in the circuit. The module-level delay analysis will be quite inaccurate, whereas it should be possible to optimize the small circuit benchmarks flat. The paper assumed that level converter overheads are negligible relative to the circuit module critical path delay and power, but this is incorrect for the small benchmarks that the paper considers. For example, a typical 0.13um level converter flip-flop delay is about 10% of the critical path delay of most of the ISCAS'85 benchmarks. They incorrectly assumed that short circuit power can be specified in the same manner as dynamic power, being proportional only to the square of the supply voltage and independent of Vth. Unfortunately, short circuit power also depends on the NMOS and PMOS threshold voltages, which determine how long the gate has a conducting path from Vdd to

ground before the transistors are turned off [233]. The process technology used was 0.25um with 2.5V nominal supply voltage and 0.5V threshold voltage.

If each module is assigned individual Vdd and Vth values, the power savings were 29%. However, this is not practical in today's technologies – dual Vdd and triple Vth is about the practical limit due to expense and other issues such as yield. They reported average power savings of 18% with switching activity of 0.05 using dual Vdd with optimal choice of voltages for VDDH, VDDL and Vth. The values for VDDH and VDDL varied widely with switching activity, and at low activity the VDDL was as low as 30% of VDDH (0.5V versus 1.6V).

The optimization is over-constrained. By using only the nullspace of the path matrix, the path delays are not allowed to decrease or increase. The starting point for optimization is after sizing power minimization with the zero-slack algorithm [151]. They assume that there is no slack on any path to an output, which is an incorrect assumption for many designs – for example, there may be short paths with minimum sized gates that have delay substantially less than the delay constraint. In which case, the delay can be increased and power minimized by reducing Vdd or increasing Vth, but the restriction that path delays are not allowed to increase may prevent this change.

Runtimes grew as $O(N \times |\text{Null}(A)|)$, which is in terms of the number of modules N and the rank of the nullspace of A , and note that $|\text{Null}(A)| \leq N$ [57]. The runtimes are quite large, up to 150 seconds for c3540 on an 800MHz Pentium III, given the module-level simplification to reduce complexity. This suggests that the approach will not be useful for circuits of significant size, particularly given that the module-level delay and power analysis is quite inaccurate and thus will produce suboptimal results.

H.2.7 Kulkarni, Srivastava and Sylvester: Greedy ECVS methodology for dual Vdd (2004) [124][125]

Unlike the ECVS algorithm that proceeds in reverse topological order, Kulkarni, Srivastava and Sylvester prioritized gates for assignment from VDDH to VDDL using this metric [125]:

$$Sensitivity_{reduce\ power} = \frac{-\Delta P \times slack}{\Delta d} \quad (H.3)$$

where ΔP is the change in power, $\Delta d > 0$ is sum of the worst changes in rise and fall delay on timing arcs through the gate, and *slack* is the sum of the worst slacks on rise and fall timing arcs through the gate. This metric includes the change in power due to adding or removing level converters. The starting point for optimization was a netlist sized with a TILOS-like sizer to meet the delay constraint. The gate with maximum sensitivity that won't violate the delay constraints is assigned to VDDL, then sensitivity analysis is updated, and this procedure is iterated. Moves which increase power are allowed, in the hope that assigning additional gates to VDDL may provide an overall power reduction. The configuration that provided the minimum power is stored and used as the solution after no further gates can be assigned to VDDL. They refer to this algorithm as greedy ECVS, or GECVS.

They used the PowerArc characterized 0.13um libraries with supply voltages of VDDH=1.2V and VDDL of 0.8V or 0.6V, and threshold voltages of 0.12V for NMOS and -0.09V for PMOS. A wire load model of $C_{wire} = 3+2\times\#fanouts$ was used. The output port load was 3fF. Analysis included the delay and power overheads for asynchronous level converters, using two sizes of the strength 5 level converter described in [123]. ECVS gave average power savings of 7.3% versus CVS, and GECVS gave average power savings of 13.4% versus CVS and 6.5% versus ECVS. These percentage savings are averaged across their results with VDDL values of 0.8V and 0.6V, and delay constraints of $1.1\times T_{min}$ and $1.2\times T_{min}$ [124].

Updated results using Equation (5.10) with α of 1.101 for scaling Vdd=0.8V delays to 0.6V will appear in [125]. For those results they used 80ps LCFF delays. Having corrected the 0.6V delays, they found that VDDL of 0.8V generally produced better results than VDDL of 0.6V. In comparison to the TILOS sizing results for a delay constraint of $1.1 \times T_{\min}$, the average power savings were 9.1% for CVS and 14.7% for GECVS. GECVS saved 6.4% power on average versus the CVS results. Asynchronous level converters consumed on average 6% of the total power. In comparison to the better gate sizing results from the linear programming approach, the average power savings were only 1.7% for CVS and 7.6% for GECVS.

Their GECVS methodology for dual Vdd has worst case theoretical runtime complexity of $O(|V|^3)$ [125], which is worse than the ECVS algorithm, because all VDDH gates are considered simultaneously for assignment to VDDL.

H.3 Papers on optimization with multi-Vth and multi-Vdd

Very few papers have proposed algorithmic approaches that consider assignment of both multiple threshold voltages and multiple supply voltages.

H.3.1 Srivastava, Sylvester and Blaauw: CVS methodology for dual Vdd and dual Vth assignment with gate sizing (2004) [125][188]

Srivastava, Sylvester and Blaauw extended the CVS algorithm to include gate sizing and dual Vth assignment. They start with a circuit that has been produced by their TILOS-like optimizer, using the delay_reduction/power_increase sensitivity metric in Equation (4.3) for gate upsizing, from a minimum size circuit to meet a given delay constraint. To aid the reader, we present the metric again here [188]

$$Sensitivity_{reduce\ delay} = -\frac{1}{\Delta P} \sum_{l \in timing_arcs} \frac{\Delta d_l}{Slack_l - Slack_{\min} + k} \quad (\text{H.4})$$

where $\Delta P > 0$ is the change in total power for upsizing a gate. This is essentially the same metric as was used by Sirichotiyakul et al. in [183] for the delay versus leakage power trade-off. The timing slack on a timing arc through a gate from input i to output j is computed as [125]

$$Slack_{arc\ ij} = (t_{required\ at\ output\ j} - t_{arrival\ at\ input\ i}) - d_{arc\ ij} \quad (H.5)$$

where $t_{required\ at\ output\ j}$ is the time the transition must occur at gate output j for the delay constraint to be met on paths that include the timing arc, $t_{arrival\ at\ input\ i}$ is the arrival time at input i , and $d_{arc\ ij}$ is the delay on the timing arc. The slack is the maximum increase in delay of the timing arc that will satisfy the delay constraints, and will be negative if a delay constraint is not met on a path through the gates with that timing arc.

The high Vdd/low Vth starting point is power-minimized with this gate sizing technique. Then the CVS algorithm is performed on this circuit. After CVS, no VDDH gates with all VDDL fanouts can be changed to VDDL without violating the timing constraints. They then perform gate upsizing on the critical paths prioritized by the metric in Equation (H.4). The CVS algorithm then continues with additional slack to assign more gates to VDDL. The lowest power circuit configuration is kept track of after each gate upsizing/VDDL assignment iteration. When the iterations finish, the lowest power configuration is picked [188].

This approach for dual Vdd assignment with gate sizing is similar to the CVS dual Vdd assignment with gate sizing approach by Yeh et al. [242], described in Section H.2.4. However, Yeh et al. used delay_reduction/area_increase as the sensitivity metric for gate upsizing. The different delay and power models prevent direct comparison, but given the use of a metric that considers power when upsizing gates, it would be expected that the approach in [188] is better. The two optimization approaches have the same worst case runtime complexity.

Having found the dual-Vdd/gate-sizing minimum power circuit, they then consider assignment to high Vth (VthH) to reduce leakage power. VDDL gates with all VDDH fanins may be assigned back to VDDH without needing level converters to be inserted. They used this sensitivity metric to prioritize gate upsizing or assignment to VDDH [125]:

$$Sensitivity_{reduce\ delay} = -\frac{1}{\Delta P} \sum_{l \in timing_arcs} \Delta d_l \quad (H.6)$$

The gate with maximum sensitivity is upsized or assigned to VDDH. This timing slack is exploited by assigning low Vth gates to high Vth to reduce the leakage. Gates are iteratively assigned to high Vth by maximum sensitivity with the metric [188]

$$Sensitivity_{reduce\ power} = -\Delta P \sum_{l \in timing_arcs} \frac{Slack_l}{\Delta d_l} \quad (H.7)$$

where $\Delta d_l > 0$. If the method for increasing slack was gate upsizing and power wasn't reduced overall, the changes are rolled back and the gate is not allowed to be upsized in future. VDDH assignments are accepted even if power increases overall after VthH assignment, because further beneficial VDDH assignments may otherwise be prevented. This process is iterated. The lowest power circuit configuration is kept track of after each gate upsizing/VDDH assignment/VthH assignment iteration. When the iterations finish, the lowest power configuration is picked [125].

The results in [188] were wrong due to the incorrect $Vdd=0.6V$ delay characterization. Updated results with VDDL of 0.8V appear in [125], with VDDL=0.6V results excluded as those were worse. They used the PowerArc characterized 0.13um libraries with VDDH of 1.2V, and threshold voltages of 0.12V and 0.23V. A wire load model of $C_{wire} = 3 + 2 \times \#fanouts$ was used. The output port load was 3fF. The LCFF delay overhead was 80ps and there was no power overhead. In comparison to the TILOS sizing results for a delay constraint of $1.1 \times T_{min}$, the average power savings were 18.9%. In comparison to the better gate sizing results from the linear programming

approach, the average power savings were only 12.3%. Their CVS approach with gate sizing and dual V_{th} achieved 10.6% power savings versus CVS.

The delay constraint in [188] was $1.2 \times T_{\min}$, where T_{\min} is the minimum delay achieved with the TILOS-like sizer with V_{dd} of 1.2V and V_{th} of 0.12V. A tighter timing constraint was used in [125], because Design Compiler, with buffer insertion et al. in addition to gate sizing, was found to achieve smaller T_{\min} than the TILOS-sizer. Thus there was concern that the TILOS sized starting point may have more slack, allowing power savings to be overstated.

The theoretical worst case runtime complexity for the CVS methodology for dual V_{dd} and dual V_{th} assignment with gate sizing is $O(|V|^3)$.

H.3.2 Hung et al.: ECVS methodology for dual V_{dd} and dual V_{th} assignment with gate sizing and stack forcing (2004) [99]

Hung et al. proposed a genetic algorithm to simultaneously perform V_{dd} assignment, V_{th} assignment, gate sizing and stack forcing [99]. Stack forcing is a technique to reduce gate leakage, by splitting a transistor into two series transistors. Stack forcing approximately doubles the delay of an inverter, but reduces the leakage by one to two orders of magnitude, and the reduction is greater as technology is scaled from 0.18um to 0.06um [152]. Increasing transistor channel length to reduce leakage is not as effective as stack forcing, because the delay penalty is higher. Stack forcing offers a cheaper leakage reduction approach than dual V_{th}, which has additional fabrication costs for the additional masks needed to implant the second threshold voltages and that also impacts yield. Stack forcing does not necessitate a different algorithmic approach and the power/delay trade-off could just as easily be considered in a gate-level TILOS-like optimizer or our linear programming approach.

The genetic algorithm encodes the gate size, V_{th}, V_{dd} and stack forcing (or not) assignment in a binary string for the circuit. For example a 0 at a particular bit position may represent VDDL and

a 1 represent VDDH. The genetic algorithm proceeds by creating new circuit configurations by mutating, i.e. flipping bits randomly, the circuit configuration from the previous generation. Crossover then occurs, where the best part of circuit configurations may be combined. The probability of mutation or crossover is determined by a metric of 1/power for the circuit, with a penalty if there are a substantial number of timing violations [99]. In a sense, the genetic algorithm approach is similar to simulated annealing, but multiple circuit configurations exist at each iteration.

Their library was characterized with the Berkeley 0.065um BSIM predictive model, with supply voltages of 1V and 0.5V and threshold voltages of 0.22V and 0.12V. They report that dual Vdd was the approach that provided the best power savings for an 8×8 multiplier and stack forcing provided about 10% additional power savings on a 32-bit adder. However, the results presented in [99] are dubious, as they only consider two gate sizes.

Using more realistic gate sizes in combination with dual Vdd and dual Vth, Sylvester, Srivastava and Kulkarni compared the genetic algorithm to gate sizing in TILOS. They found that the genetic algorithm only achieved power savings versus TILOS (15%) on the smallest ISCAS'85 benchmark, c17. The genetic algorithm was also very slow [125]. This is not surprising as the random choices in the genetic algorithm will perform poorly versus a steepest descent algorithm, randomly flipping a bit (corresponding to say changing a gate to VDDL) can easily result in a timing violation, and many more circuit configurations need to be analyzed for the genetic algorithm. The only benefit of an approach like the genetic algorithm is hill climbing out of local optima, but this is best done using a circuit that is already optimized for low power as a starting point. This example makes clear the importance of comparing novel algorithmic approaches to a standard TILOS-like optimizer.

H.3.3 Kulkarni, Srivastava and Sylvester: Greedy ECVS methodology for dual Vdd and dual Vth assignment with gate sizing (2006) [125]

This work extended the GECVS algorithm described in Section H.2.7 to include gate sizing and dual Vth assignment. The starting point is after GECVS has been performed, where any additional assignments to VDDL will cause a delay constraint violation. Then they grow the VDDL clusters by considering only VDDH gates with VDDL fanouts for assignment to VDDL, avoiding the need for additional level converters. The same metric for prioritizing assignment to VDDL is used, Equation (H.3), and the gate with maximum sensitivity is assigned to VDDL, violating a delay constraint. Gates are then upsized to reduce the delay, per the TILOS-like sizing metric in Equation (H.4). This process of assigning a gate to VDDL and upsizing gates to meet the delay constraints is iterated until no more gates are candidates for assignment to VDDL. Again, moves which increase power are allowed subject to a user-specified maximum tolerance for hill climbing, in the hope that assigning additional gates to VDDL may provide an overall power reduction [125].

Remaining slack in the circuit is used to iteratively assign gates from low Vth to high Vth to reduce leakage power, prioritized by the metric in Equation (H.3). Then gates are upsized or assigned to VDDH (only allowed if all fanins are VDDH to avoid needing level converters to be inserted) to get additional slack to assign more gates to high Vth, prioritized by the metric

$$Sensitivity_{reduce\ delay} = -\frac{\Delta d}{\Delta P} \quad (H.8)$$

where $\Delta P > 0$ is the change in power, and Δd is the sum of the worst change in rise and fall delays on timing arcs through the gate. Then more gates are assigned to high Vth, and this process is iterated until no more gates can be assigned to high Vth. The minimum power configuration found in the whole procedure is picked as the solution [125].

Results with VDDL of 0.6V, using the Equation (5.10) with α of 1.101 delay scaling from Vdd=0.8V, were not reported, as those results were worse than using VDDL of 0.8V. They used the PowerArc characterized 0.13um libraries with VDDH of 1.2V, and threshold voltages of 0.23V and 0.12V. A wire load model of $C_{wire} = 3+2\times\#fanouts$ was used. The output port load was 3fF. The delay constraint was 1.1 times the minimum delay (T_{min}) achieved with the TILOS sizer at Vdd=1.2V/Vth=0.12V. The GECVS approach with gate sizing and dual Vth achieved average power savings of 21.6% versus the TILOS-like sizer, or 15.0% versus our LP sizing results. The GECVS approach with gate sizing and dual Vth achieved 7.9% power savings versus using GECVS alone.

Theoretical worst case runtime complexity was $O(|V|^3)$, the same as the CVS approach with gate sizing and dual Vth assignment described in [188].

Appendix I Multi-Vdd and multi-Vth with gate sizing results vs. University of Michigan

We used the same libraries and conditions as the University of Michigan work [124][125][188]. The port loads were $3fF$, excluding the additional wire load. The wire loads were $3+2\times\text{num_fanout} fF$, and slews were 0.1ns for the 1.2V input drive ramps. Switching activities were multiplied by a fraction such that leakage was about 20% of total power at $Vdd=1.2V/Vth=0.12V$. We used an 80ps delay overhead for level converter flip-flops, and used the two characterized sizes of the strength 5 asynchronous level converter described in [123]. The ISCAS'85 and Huffman benchmarks for comparison were discussed in Section 4.6.1. Switching activity was multiplied by a fraction such that leakage was about 20% of total power at $Vdd=1.2V/Vth=0.12V$. VDDH was 1.2V and VDDL was 0.8V or 0.6V. Vth was 0.23V or 0.12V.

For results in this appendix, the PowerArc characterized 0.13um library at 25°C with Vdd of 1.2V and Vth of 0.12V was used. There were thirteen inverter cell sizes: XL, X1, X2, X3, X4, X6, X8, X10, X12, X14, X16, X18, X20; and eight sizes for NAND2, NAND3, NOR2, NOR3: XL, X1, X2, X4, X5, X6, X7, X8. The University of Michigan TILOS-like sizer started from a netlist of size X1 gates and then only considered upsizing gates. Consequently, they did not use size XL cells¹. In Section I.1, we disallowed the XL cell size for some of the LP results, to ensure the comparison has identical conditions. We did allow the XL cell size with our LP approach for all the results in Section I.2. For multi-Vdd with a low supply voltage of 0.8V or 0.6V, we encountered slews that were outside the cell input slew characterization range. These cell sizes were characterized with input slew of up to 1.02ns (for sizes X6, X10, X14, X18 for the inverter, and X5, X6, X7 and X8 for the other gates) or 1.8ns (for the other cell sizes). To avoid input slews exceeding 1.02ns, the maximum cell capacitance was set to prevent a gate having an output slew of more than 1.02ns.

¹ Private communication with Ashish Srivastava.

There were some accuracy issues with the University of Michigan power analysis, as discussed in Appendix G.4.1. To account for this, their power results are calculated from the relative power saving percentages that they provided and the Design Compiler analysis of their $1.0 \times T_{\min}$ TILOS sized netlists. For example for c432 at $1.5 \times T_{\min}$, the relative power saving with Sarvesh Kulkarni's sizing/multi-Vth/multi-Vdd ECVS approach was 17.1% versus the $1.5 \times T_{\min}$ TILOS-sized netlist, and the $1.5 \times T_{\min}$ TILOS-sized netlist had 0.417 of the $1.0 \times T_{\min}$ TILOS-sized netlist, which has power of 0.275mW measured in Design Compiler – so the sizing/multi-Vth/multi-Vdd ECVS power reported is $(1 - 17.1\% / 100\%) \times 0.417 \times 0.275 = 0.095\text{mW}$.

The linear programming results presented here are for the conservative slew analysis setting, described in Appendix G.1, as that mostly produced better results than the aggressive slew analysis setting.

I.1 Results at $1.1 \times T_{\min}$, delay scaling by Equation (5.10) and α of 1.101

The power results presented here are at a delay constraint of $1.1 \times T_{\min}$, where T_{\min} is the minimum delay found by the TILOS-like sizer. The results with VDDL of 0.6V in this section use the delay scaling from 0.8V to 0.6V with Equation (5.10) and α of 1.101.

Table I.1 Comparison of the linear programming (LP) results versus the University of Michigan (UM) results for a delay constraint of $1.1 \times T_{\min}$. Cases where CVS multi-Vdd/multi-Vth/sizing performs worse than Vdd=1.2V/multi-Vth/sizing and where ECVS performs worse than CVS are highlighted in red. At the bottom, percentage power savings versus the linear programming sizing results at Vdd=1.2V/Vth=0.12V are shown. Cells of the smallest size XL were not allowed in these results. Results highlighted in blue had only VDDH=1.2V cells.

Netlist	1.1xTmin (ns)	LP								UM	
		TILOS	CVS				ECVS		CVS	ECVS	
			Vth (V)	0.12	0.23	0.23	0.23	0.23			
Netlist	1.1xTmin (ns)	TILOS	Vth (V)	0.12	0.23	0.23	0.23	0.23	CVS	ECVS	
c432	0.815	0.217	0.176	0.165	0.168	0.160	0.165	0.161	0.189	0.196	
c880	0.871	0.317	0.313	0.264	0.243	0.245	0.239	0.242	0.242	0.231	
c1355	0.859	0.757	0.623	0.588	0.586	0.582	0.572	0.573	0.690	0.685	
c1908	1.249	0.441	0.405	0.365	0.360	0.359	0.346	0.349	0.360	0.375	
c2670	0.809	0.649	0.637	0.524	0.453	0.435	0.431	0.426	0.468	0.459	
c3540	1.285	0.724	0.696	0.592	0.581	0.578	0.545	0.551	0.605	0.615	
c5315	1.131	1.209	1.188	1.008	0.974	0.956	0.871	0.854	0.987	0.888	
c7552	1.512	1.165	1.142	0.892	0.796	0.822	0.712	0.719	0.890	0.764	
Huffman	1.520	0.390	0.373	0.314	0.304	0.301	0.284	0.284	0.313	0.283	
Power (mW)											
Power savings versus LP sizing											
c432	-23.0%	0.0%	6.6%	4.5%	9.2%	6.6%	8.7%	-7.4%	-11.2%		
c880	-1.4%	0.0%	15.9%	22.4%	21.9%	23.7%	22.6%	22.9%	26.3%		
c1355	-21.4%	0.0%	5.7%	6.0%	6.7%	8.2%	8.0%	-10.8%	-9.9%		
c1908	-8.9%	0.0%	9.8%	11.1%	11.2%	14.4%	13.8%	11.1%	7.4%		
c2670	-1.8%	0.0%	17.7%	29.0%	31.8%	32.3%	33.1%	26.5%	27.9%		
c3540	-3.9%	0.0%	15.1%	16.6%	17.0%	21.8%	20.9%	13.1%	11.7%		
c5315	-1.7%	0.0%	15.2%	18.0%	19.5%	26.7%	28.1%	16.9%	25.3%		
c7552	-2.1%	0.0%	21.8%	30.3%	28.0%	37.6%	37.1%	22.0%	33.1%		
Huffman	-4.4%	0.0%	15.8%	18.5%	19.3%	23.7%	23.9%	16.2%	24.1%		
Average	-7.6%	0.0%	13.7%	17.4%	18.3%	21.7%	21.8%	12.3%	15.0%		

Table I.2 Linear programming (LP) results for a delay constraint of $1.1 \times T_{\min}$ with XL cells allowed. In the middle, percentage power savings versus the LP sizing results at $Vdd=1.2V/Vth=0.12V$ with XL cells are shown. Results highlighted in blue had only VDDH=1.2V cells. At the bottom, percentage power savings versus LP results for the same Vdd and Vth values without XL cells are shown. Cases where the LP results with XL cells were worse than the LP results without XL cells are highlighted in red.

Netlist	$1.1 \times T_{\min}$ (ns)	LP			
		CVS	CVS	ECVS	ECVS
		Vth (V)	0.12	0.23, 0.12	0.23, 0.12
Vdd (V)	1.2	1.2	1.2, 0.8	1.2, 0.6	1.2, 0.8
		1.2	1.2, 0.8	1.2, 0.6	1.2, 0.6
		Power (mW)			
c432	0.815	0.170	0.168	0.165	0.165
c880	0.871	0.278	0.248	0.249	0.245
c1355	0.859	0.632	0.575	0.576	0.581
c1908	1.249	0.392	0.356	0.353	0.354
c2670	0.809	0.550	0.476	0.426	0.411
c3540	1.285	0.634	0.561	0.556	0.550
c5315	1.131	1.061	0.953	0.909	0.907
c7552	1.512	0.959	0.793	0.770	0.768
Huffman	1.520	0.333	0.302	0.301	0.298
		Power savings versus LP sizing with XL cells			
c432	0.0%	1.2%	2.7%	3.0%	4.1%
c880	0.0%	11.0%	10.4%	12.0%	17.3%
c1355	0.0%	9.1%	8.9%	8.2%	10.1%
c1908	0.0%	9.2%	10.0%	9.8%	11.7%
c2670	0.0%	13.5%	22.6%	25.4%	23.5%
c3540	0.0%	11.5%	12.2%	13.2%	13.7%
c5315	0.0%	10.2%	14.4%	14.6%	19.7%
c7552	0.0%	17.4%	19.7%	19.9%	28.4%
Huffman	0.0%	9.4%	9.7%	10.5%	12.6%
Average	0.0%	10.3%	12.3%	13.0%	15.7%
		Power savings versus no XL cells			
c432	3.6%	-2.0%	1.8%	-2.9%	1.0%
c880	11.1%	6.0%	-2.6%	-0.2%	3.6%
c1355	-1.5%	2.2%	1.7%	0.2%	0.6%
c1908	3.1%	2.4%	1.9%	1.5%	0.0%
c2670	13.6%	9.2%	5.8%	5.5%	2.3%
c3540	9.0%	5.1%	4.2%	4.8%	-0.5%
c5315	10.7%	5.5%	6.7%	5.2%	2.1%
c7552	16.0%	11.2%	3.3%	6.6%	3.6%
Huffman	10.8%	4.0%	1.1%	1.1%	-2.2%
Average	8.5%	4.8%	2.6%	2.4%	1.2%

I.2 Results at $1.1 \times$, $1.2 \times$ and $1.5 \times T_{\min}$, delay scaling by Equation (5.9) and α of 1.3

The power results presented here are at a delay constraint of $1.1 \times T_{\min}$, $1.2 \times T_{\min}$ and $1.5 \times T_{\min}$, where T_{\min} is the minimum delay found by the TILOS-like sizer. The minimum XL cell size was

allowed with the linear programming approach for these results. The VDDL=0.6V results in this section use the delay scaling from 0.8V to 0.6V with Equation (5.9) and α of 1.3, which is optimistic for the delays at 0.6V. Results for the TILOS-like sizing optimizer used by Sarvesh Kulkarni and Ashish Srivastava are reported in the second column of the tables. The third column reports the linear programming sizing only results, followed by LP sizing/multi-Vth results in the fourth column, LP sizing/multi-Vth/CSV multi-Vdd in column five, and LP sizing/multi-Vth/ECVS multi-Vdd in column six. Columns seven and eight list the sizing/multi-Vth/multi-Vdd results from University of Michigan for their CVS and ECVS approaches that incorporated gate sizing and multi-Vth.

Dual Vth with sizing in the LP approach achieves 10.6%, 13.2%, and 19.4% lower power than LP sizing alone at $1.1 \times T_{\min}$, $1.2 \times T_{\min}$, and $1.5 \times T_{\min}$ respectively, as shown in the middle tables where the power savings versus the linear programming sizing results are reported. The University of Michigan sizing/multi-Vth/multi-Vdd results for netlists c432 and c1355 are worse the LP sizing results, because multi-Vdd is not particularly helpful for these netlists and their TILOS-like sizer is worse than the LP approach.

Despite the overly optimistic $\alpha=1.3$ delay scaling from 0.8V delays to 0.6V delays, multi-Vdd does not provide that much additional power savings beyond sizing with multi-Vth. Dual Vdd of 1.2V and 0.6V with dual Vth and sizing with the ECVS LP approach (setting asynchronous level converter power to zero in the first LP run, then correcting it) achieves 18.0%, 22.9%, and 32.0% lower power than LP sizing alone. This is on average 8.5%, 11.4%, and 17.3% power savings versus LP sizing/dual Vth at $1.1 \times T_{\min}$, $1.2 \times T_{\min}$, and $1.5 \times T_{\min}$ respectively. However, we see up to 41.6% savings with ECVS dual Vdd/dual Vth/sizing vs. dual Vth/sizing for c7552 at $1.5 \times T_{\min}$.

The linear programming CVS and ECVS results are better than the University of Michigan CVS and ECVS results, as detailed at the bottom of Table I.3, Table I.4 and Table I.5, except for three

cases for CVS highlighted in red (c7552 at $1.2 \times T_{\min}$ and $1.5 \times T_{\min}$, Huffman at $1.1 \times T_{\min}$). Average power savings for the LP approach are around 10%, varying from 8.2% on average for CVS at $1.1 \times T_{\min}$ to 12.7% on average for ECVS at $1.2 \times T_{\min}$. Note that the average savings for sizing with the LP approach are more than we achieved with multi-Vdd, which suggests the University of Michigan multi-Vdd approach provides good results given that their TILOS-like sizer is not as good.

ECVS provides from 5.3% to 9.8% average power savings vs. CVS from $1.1 \times T_{\min}$ and $1.5 \times T_{\min}$, as reported at the bottom right of these tables. As more timing slack is available at a larger delay constraint, there are more opportunities to use asynchronous level converters. For ISCAS'85 benchmark c7552 with ECVS, we see as much as 17.5% power savings at $1.1 \times T_{\min}$ to 29.9% power savings vs. CVS at $1.5 \times T_{\min}$. However, as noted below, the linear programming CVS approach is 18.7% worse than the University of Michigan CVS approach at $1.5 \times T_{\min}$. This indicates that ECVS is at best about 20% better than CVS for the results on these benchmarks (e.g. c5315 at $1.5 \times T_{\min}$).

Note that the $\alpha=1.3$ delay scaling that was used in this subsection is an underestimate by 26% at $V_{dd}=0.6V/V_{th}=0.23V$ and 19% at $V_{dd}=0.6V/V_{th}=0.12V$, which causes multi-Vdd results with $V_{DDL}=0.6V$ in this section to be significantly better than they should be.

Table I.3 Comparison of the linear programming (LP) results versus the University of Michigan (UM) results for a delay constraint of $1.1 \times T_{\min}$. Cases where CVS multi-Vdd/multi-Vth/sizing performs worse than Vdd=1.2V/multi-Vth/sizing and where ECVS performs worse than CVS are highlighted in red. In the middle, percentage power savings versus the linear programming sizing results at Vdd=1.2V/Vth=0.12V are shown. At the bottom, percentage power savings for CVS and ECVS versus the University of Michigan CVS and ECVS results are shown, then linear programming ECVS power savings are reported versus CVS.

Netlist	TILOS	LP				UM		
				CVS	ECVS	CVS	ECVS	
		Vdd (V)	0.12	0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	
		Vth (V)	1.2	1.2	1.2	1.2, 0.6	1.2, 0.6	
Netlist	1.1xTmin (ns)	Power (mW)						
	c432	0.815	0.217	0.170	0.165	0.163	0.190	0.195
	c880	0.871	0.317	0.284	0.246	0.243	0.223	0.240
	c1355	0.859	0.757	0.621	0.578	0.584	0.575	0.690
	c1908	1.249	0.441	0.391	0.356	0.350	0.333	0.358
	c2670	0.809	0.649	0.559	0.480	0.411	0.392	0.455
	c3540	1.285	0.724	0.637	0.560	0.550	0.548	0.586
	c5315	1.131	1.209	1.062	0.950	0.899	0.853	0.954
	c7552	1.512	1.165	0.959	0.792	0.762	0.628	0.822
	Huffman	1.520	0.390	0.333	0.302	0.296	0.280	0.340
Power savings versus LP sizing with XL cells								
c432	-27.6%	0.0%	3.1%	3.7%	4.1%	-11.8%	-15.0%	
c880	-11.8%	0.0%	13.3%	14.5%	21.5%	14.3%	15.6%	
c1355	-21.8%	0.0%	6.9%	6.1%	7.5%	-11.1%	-10.3%	
c1908	-12.8%	0.0%	9.0%	10.3%	14.7%	8.4%	7.8%	
c2670	-16.1%	0.0%	14.2%	26.5%	29.8%	18.5%	22.2%	
c3540	-13.7%	0.0%	12.0%	13.6%	13.9%	7.9%	5.6%	
c5315	-13.8%	0.0%	10.5%	15.3%	19.7%	10.2%	16.0%	
c7552	-21.5%	0.0%	17.5%	20.6%	34.5%	14.3%	25.8%	
Huffman	-17.1%	0.0%	9.3%	11.1%	16.0%	-2.2%	11.6%	
Average	-17.3%	0.0%	10.6%	13.5%	18.0%	5.4%	8.8%	
LP savings vs. UM								
Netlist	CVS		LP ECVS savings versus LP CVS					
	c432	13.9%	16.6%			0.4%		
	c880	0.2%	6.9%			8.2%		
	c1355	15.5%	16.1%			1.5%		
	c1908	2.1%	7.5%			4.9%		
	c2670	9.8%	9.8%			4.5%		
	c3540	6.2%	8.8%			0.3%		
	c5315	5.8%	4.4%			5.1%		
	c7552	7.3%	11.7%			17.5%		
	Huffman	13.0%	5.0%			5.5%		
	Average	8.2%	9.6%			5.3%		

Table I.4 Comparison of the linear programming (LP) results versus the University of Michigan (UM) results for a delay constraint of $1.2xT_{\min}$. Cases where CVS multi-Vdd/multi-Vth/sizing performs worse than Vdd=1.2V/multi-Vth/sizing and where ECVS performs worse than CVS are highlighted in red. In the middle, percentage power savings versus the linear programming sizing results at Vdd=1.2V/Vth=0.12V are shown. At the bottom, percentage power savings for CVS and ECVS versus the University of Michigan CVS and ECVS results are shown, then linear programming ECVS power savings are reported versus CVS. Cases where the LP approach is worse than the University of Michigan results are shown in red in the bottom table.

Netlist	TILOS	LP				UM		
				CVS	ECVS	CVS	ECVS	
		Vdd (V)	0.12	0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12	
Vth (V)	1.2	1.2	1.2	1.2, 0.6	1.2, 0.6	1.2, 0.6	1.2, 0.6	
1.2xTmin (ns)	Power (mW)							
c432	0.893	0.172	0.137	0.124	0.124	0.121	0.144	0.148
c880	0.959	0.279	0.228	0.194	0.180	0.170	0.191	0.188
c1355	0.943	0.553	0.445	0.421	0.423	0.417	0.477	0.496
c1908	1.375	0.354	0.305	0.268	0.260	0.252	0.280	0.284
c2670	0.890	0.598	0.474	0.398	0.332	0.323	0.395	0.396
c3540	1.412	0.601	0.515	0.443	0.436	0.411	0.485	0.485
c5315	1.246	1.061	0.891	0.764	0.722	0.633	0.767	0.719
c7552	1.666	1.048	0.845	0.675	0.642	0.516	0.639	0.548
Huffman	1.669	0.342	0.280	0.245	0.234	0.210	0.249	0.226

Netlist	Power savings versus LP sizing with XL cells						
c432	-25.0%	0.0%	10.0%	10.0%	12.2%	-4.8%	-7.6%
c880	-22.1%	0.0%	15.1%	21.0%	25.6%	16.2%	17.7%
c1355	-24.3%	0.0%	5.3%	5.0%	6.2%	-7.3%	-11.4%
c1908	-16.2%	0.0%	12.0%	14.8%	17.4%	8.3%	6.8%
c2670	-26.2%	0.0%	16.1%	29.9%	31.8%	16.6%	16.5%
c3540	-16.8%	0.0%	14.0%	15.3%	20.2%	5.7%	5.7%
c5315	-19.2%	0.0%	14.2%	18.9%	29.0%	13.9%	19.2%
c7552	-24.1%	0.0%	20.1%	24.0%	38.9%	24.4%	35.1%
Huffman	-22.3%	0.0%	12.4%	16.3%	25.1%	10.9%	19.1%
Average	-21.8%	0.0%	13.2%	17.3%	22.9%	9.3%	11.2%

Netlist	LP savings vs. UM		LP ECVS savings versus LP CVS
	CVS	ECVS	
c432	14.2%	18.4%	2.4%
c880	5.8%	9.6%	5.8%
c1355	11.4%	15.8%	1.3%
c1908	7.1%	11.3%	3.1%
c2670	16.0%	18.3%	2.8%
c3540	10.2%	15.3%	5.8%
c5315	5.8%	12.0%	12.4%
c7552	-0.5%	5.9%	19.6%
Huffman	6.0%	7.3%	10.5%
Average	8.4%	12.7%	7.1%

Table I.5 Comparison of the linear programming (LP) results versus the University of Michigan (UM) results for a delay constraint of $1.5 \times T_{\min}$. Cases where CVS multi-Vdd/multi-Vth/sizing performs worse than Vdd=1.2V/multi-Vth/sizing and where ECVS performs worse than CVS are highlighted in red. In the middle, percentage power savings versus the linear programming sizing results at Vdd=1.2V/Vth=0.12V are shown. At the bottom, percentage power savings for CVS and ECVS versus the University of Michigan CVS and ECVS results are shown, then linear programming ECVS power savings are reported versus CVS. Cases where the LP approach is worse than the University of Michigan results are shown in red in the bottom table.

Netlist	TILOS	LP				UM	
				CVS	ECVS	CVS	ECVS
		Vdd (V)	0.12	0.12	0.23, 0.12	0.23, 0.12	0.23, 0.12
		Vth (V)	1.2	1.2	1.2	1.2, 0.6	1.2, 0.6
	1.5xTmin (ns)	Power (mW)					
c432		1.117	0.115	0.084	0.072	0.071	0.072
c880		1.199	0.206	0.161	0.129	0.109	0.106
c1355		1.179	0.370	0.278	0.236	0.236	0.240
c1908		1.718	0.246	0.197	0.165	0.157	0.158
c2670		1.113	0.465	0.359	0.281	0.229	0.203
c3540		1.765	0.436	0.353	0.277	0.270	0.247
c5315		1.557	0.814	0.658	0.523	0.477	0.385
c7552		2.082	0.856	0.669	0.499	0.416	0.291
Huffman		2.086	0.266	0.208	0.167	0.149	0.119
						0.135	0.126

Netlist	Power savings versus LP sizing with XL cells						
c432	-36.3%	0.0%	14.7%	15.5%	14.0%	-24.7%	-13.0%
c880	-28.4%	0.0%	19.5%	32.2%	34.3%	12.7%	33.9%
c1355	-33.2%	0.0%	15.1%	15.1%	13.5%	-4.7%	-7.3%
c1908	-25.3%	0.0%	16.2%	20.2%	19.8%	2.0%	10.6%
c2670	-29.5%	0.0%	21.6%	36.3%	43.3%	22.5%	35.6%
c3540	-23.7%	0.0%	21.6%	23.6%	30.0%	1.0%	17.1%
c5315	-23.7%	0.0%	20.5%	27.5%	41.4%	24.5%	35.1%
c7552	-28.0%	0.0%	25.4%	37.8%	56.5%	47.6%	52.6%
Huffman	-28.1%	0.0%	19.6%	28.3%	42.9%	34.7%	39.5%
Average	-28.5%	0.0%	19.4%	26.3%	32.9%	12.9%	22.7%

Netlist	LP savings vs. UM		LP ECVS savings versus LP CVS
	CVS	ECVS	
c432	32.2%	23.8%	-1.8%
c880	22.3%	0.5%	3.1%
c1355	18.9%	19.4%	-1.8%
c1908	18.6%	10.3%	-0.4%
c2670	17.8%	12.0%	11.1%
c3540	22.8%	15.6%	8.4%
c5315	4.0%	9.8%	19.2%
c7552	-18.7%	8.1%	29.9%
Huffman	-10.0%	5.6%	20.4%
Average	12.0%	11.7%	9.8%

Appendix J Results for Multi-Vth and Multi-Vdd Analysis

The optimization results in this appendix start from the netlists that were delay minimized in Design Compiler with Vdd of 1.2V and Vth of 0.23V. The Design Compiler delay minimized netlists average 22% faster than the TILOS delay minimized netlists, comparing delay data in Table G.9 and Table 4.2 (varying between 1% and 65% faster). Thus, there may be significantly less timing slack at $1.2 \times T_{\min}$ for the Design Compiler netlists than for the TILOS netlists. However, it is difficult to compare the netlists as they differ because delay minimization in Design Compiler used both technology mapping and gate sizing, whereas TILOS was limited to gate sizing.

J.1 Multi-Vth/Vdd=1.2V with 1% leakage at $1.0 \times T_{\min}$

Table J.1 This table compares multi-Vth sizing power minimization results versus the best single Vth results. Leakage was normalized to 1% of total power at Vdd=1.2V/Vth=0.23V for the delay constraint of $1.0 \times T_{\min}$.

Vth (V)	Single			Dual			Triple	
	0.23	0.14	0.08	0.23	0.23	0.14	0.23	0.23
	Power (mW)							
Netlist	1.0xTmin (ns)	DC	LP					
c17	0.094	0.022	0.019	0.023	0.018	0.020	0.019	0.018
c432	0.733	0.430	0.360	0.424	0.351	0.380	0.357	0.348
c499	0.701	1.004	0.831	0.972	0.773	0.832	0.808	0.764
c880	0.700	0.531	0.509	0.615	0.483	0.486	0.495	0.459
c1355	0.778	1.258	0.997	1.169	0.951	1.015	0.972	0.938
c1908	0.999	0.833	0.750	0.899	0.704	0.762	0.742	0.704
c2670	0.649	1.268	1.117	1.379	1.058	1.111	1.101	1.038
c3540	1.054	1.716	1.461	1.767	1.395	1.486	1.449	1.375
c5315	0.946	1.968	1.850	2.296	1.735	1.849	1.824	1.723
c6288	3.305	5.245	4.654	5.441	4.540	4.818	4.580	4.475
c7552	0.847	3.346	3.100	3.665	2.907	3.076	2.999	2.864

Netlist	Power savings vs. Vdd=1.2V/Vth=0.14V						
	-13.7%	0.0%	-20.1%	7.0%	-8.0%	0.0%	7.0%
c17	-13.7%	0.0%	-20.1%	7.0%	-8.0%	0.0%	7.0%
c432	-19.6%	0.0%	-17.9%	2.5%	-5.6%	0.8%	3.3%
c499	-20.9%	0.0%	-17.1%	7.0%	-0.2%	2.8%	8.1%
c880	-4.2%	0.0%	-20.8%	5.1%	4.6%	2.8%	10.0%
c1355	-26.1%	0.0%	-17.2%	4.6%	-1.8%	2.6%	5.9%
c1908	-11.1%	0.0%	-19.9%	6.1%	-1.6%	1.0%	6.1%
c2670	-13.5%	0.0%	-23.5%	5.2%	0.5%	1.4%	7.1%
c3540	-17.4%	0.0%	-20.9%	4.5%	-1.7%	0.8%	5.9%
c5315	-6.4%	0.0%	-24.1%	6.2%	0.0%	1.4%	6.8%
c6288	-12.7%	0.0%	-16.9%	2.4%	-3.5%	1.6%	3.8%
c7552	-8.0%	0.0%	-18.3%	6.2%	0.8%	3.2%	7.6%
Average	-14.0%	0.0%	-19.7%	5.2%	-1.5%	1.7%	6.5%

J.2 Multi-Vth/Vdd=1.2V with 8% leakage at $1.0 \times T_{\min}$

To account for deeper submicron technologies where leakage is an increasing problem, we also examine a scenario where leakage is 8% of total power at Vdd=1.2V/Vth=0.23V at a delay constraint of $1.0 \times T_{\min}$. In this scenario, the penalty for using low Vth is very large, as leakage is about 75% of total power at Vth=0.08V.

For the $1.0 \times T_{\min}$ scenario where leakage was normalized to 8% of total power at Vdd=1.2V/Vth=0.23V, the leakage power is too large a portion of total power to justify reducing

Vth below 0.23V. The total power is 33.6% higher on average with Vth of 0.14V, and 197.1% larger at Vth of 0.08V, as shown in Table J.2.

Table J.2 This table compares multi-Vth sizing power minimization results versus the best single Vth results. Leakage was normalized to 8% of total power at Vdd=1.2V/Vth=0.23V for the delay constraint of $1.0 \times T_{min}$.

Vth (V)	Single			Dual			Triple	
	0.23	0.14	0.08	0.23	0.23	0.14	0.14	0.08
	Power (mW)							
Netlist	1.0xTmin (ns)	DC	LP					
c17	0.094	0.003	0.004	0.008	0.003	0.003	0.004	0.003
c432	0.733	0.054	0.071	0.150	0.051	0.054	0.071	0.051
c499	0.701	0.126	0.153	0.341	0.111	0.121	0.153	0.111
c880	0.700	0.066	0.096	0.220	0.066	0.066	0.096	0.066
c1355	0.778	0.157	0.191	0.410	0.140	0.151	0.191	0.140
c1908	0.999	0.104	0.148	0.323	0.101	0.104	0.146	0.101
c2670	0.649	0.158	0.210	0.479	0.146	0.152	0.210	0.146
c3540	1.054	0.214	0.278	0.626	0.196	0.214	0.278	0.196
c5315	0.946	0.246	0.348	0.805	0.243	0.246	0.348	0.243
c6288	3.305	0.656	0.868	1.889	0.656	0.656	0.868	0.656
c7552	0.847	0.418	0.579	1.279	0.410	0.418	0.579	0.406

Netlist	Power savings vs. Vdd=1.2V/Vth=0.23V						
c17	0.0%	-33.1%	-200.0%	0.0%	0.0%	-33.1%	0.0%
c432	0.0%	-31.4%	-178.8%	5.6%	0.0%	-31.4%	5.6%
c499	0.0%	-22.2%	-172.0%	11.4%	3.3%	-22.2%	11.4%
c880	0.0%	-44.5%	-231.9%	0.0%	0.0%	-44.5%	0.0%
c1355	0.0%	-21.6%	-161.0%	11.0%	4.1%	-21.6%	11.0%
c1908	0.0%	-42.2%	-209.9%	3.2%	0.0%	-40.5%	3.2%
c2670	0.0%	-32.4%	-202.1%	7.8%	4.1%	-32.4%	7.8%
c3540	0.0%	-29.7%	-191.8%	8.8%	0.1%	-29.7%	8.8%
c5315	0.0%	-41.6%	-227.2%	1.3%	0.0%	-41.6%	1.3%
c6288	0.0%	-32.4%	-188.2%	0.0%	0.0%	-32.4%	0.0%
c7552	0.0%	-38.4%	-205.7%	2.1%	0.0%	-38.4%	3.0%
Average	0.0%	-33.6%	-197.1%	4.6%	1.1%	-33.4%	4.7%

J.3 Multi-Vth/single Vdd with 1% leakage at $1.2 \times T_{min}$ with 80ps LCFFs

When the delay constraint is relaxed to $1.2 \times T_{min}$, the reduction in delay by reducing Vth to 0.08V allows use Vdd of 0.8V in some cases. The 80ps LCFF delay overhead for restoring to 1.2V at the output prevents the delay constraint being met in many cases. In other cases with single Vth, the delay reduction phase of the LP approach was unable to satisfy the delay constraint, so at best

only one useful power minimization iteration was performed. Later power minimization iterations resulted in the delay constraint being violated. The Vdd=0.8V/Vth=0.08V result for c6288 was only 12.5% worse than with the result with Vdd=1.2V, as 80ps is only 2% of the larger delay constraint of 3.96ns versus being 6% or more of the delay constraints for the other benchmarks.

For the 0.8V c6288 result, 65% of the gates are at high Vth and leakage is 25.6% of total power.

For the 0.8V c3540 result, 61% of the gates are at high Vth and leakage is 27.1% of total power.

In both cases a majority of the gates are at high Vth, but for c3540 this leaves substantially less slack to reduce dynamic power. In general, the highest Vth of 0.23V is not helpful at Vdd=0.8V, as those gates are much slower. 8.3% power savings are seen with Vdd=0.8V/triple Vth for c1908 versus the Vdd=0.8V/dual Vth results, but these results were worse than those with Vdd of 1.2V.

Hence triple Vth sees no additional benefit at Vdd of 0.8V.

Table J.3 This table compares multi-Vth sizing power minimization results versus the best single Vth results. Leakage was normalized to 1% of total power at Vdd=1.2V/Vth=0.23V for the delay constraint of $1.2 \times T_{\min}$.

	Single			Dual			Triple	Single	Dual	Triple
Vth (V)	0.23	0.14	0.08	0.23	0.23	0.14	0.23	0.14	0.23	0.14
Vdd (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2	0.8	0.8	0.8
Netlist	Power (mW)									
c17	0.112	0.013	0.012	0.014	0.011	0.012	0.012	0.011	failed delay constraint	
c432	0.879	0.233	0.225	0.280	0.210	0.223	0.225	0.210	failed delay constraint	
c499	0.842	0.528	0.492	0.640	0.483	0.520	0.511	0.483	failed delay constraint	
c880	0.840	0.322	0.332	0.424	0.306	0.316	0.330	0.306	failed delay constraint	
c1355	0.934	0.625	0.614	0.775	0.586	0.619	0.614	0.585	failed delay constraint	
c1908	1.199	0.479	0.480	0.620	0.445	0.464	0.483	0.443	0.853	0.698
c2670	0.779	0.750	0.771	0.983	0.718	0.731	0.771	0.715	failed delay constraint	
c3540	1.265	0.963	0.960	1.227	0.894	0.938	0.952	0.888	1.615	1.615
c5315	1.135	1.233	1.293	1.677	1.187	1.215	1.291	1.185	1.891	1.891
c6288	3.965	2.983	2.955	3.689	2.839	2.910	2.932	2.802	3.323	3.019
c7552	1.016	2.008	2.035	2.590	1.896	1.962	2.026	1.896	failed delay constraint	

J.4 Multi-Vdd/single Vth with 1% leakage at $1.2 \times T_{\min}$ with 80ps LCFFs

The largest power saving with CVS dual Vdd was 12.2% for benchmark c2670, using VDDL of 0.6V and Vth of 0.23V. For other benchmarks, the power savings with CVS dual Vdd were 3.3% or less. In some cases, average power savings were negative for dual Vdd with a particular Vth, as we used the best single Vdd/single Vth result for the baseline.

The power results in Table J.4 include the multi-Vdd results with single Vth of 0.08V, which were excluded in Table 7.5 as they are suboptimal versus the other results. As for the earlier results, Vth of 0.08V was too leaky and gave poor results.

Table J.4 This table compares CVS and ECVS dual Vdd/single Vth/sizing power minimization results versus the sizing only baseline results in Table 7.4. At the bottom are shown the ECVS power savings versus CVS.

Vth (V)	Single											
	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08
Vdd (V)	CVS dual Vdd						ECVS dual Vdd					
	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6
Netlist	Power (mW)											
c17	0.013	0.012	0.014	0.013	0.012	0.014	0.013	0.012	0.014	0.013	0.012	0.014
c432	0.231	0.224	0.280	0.232	0.225	0.280	0.231	0.224	0.273	0.232	0.225	0.277
c499	0.527	0.492	0.640	0.527	0.492	0.640	0.527	0.492	0.630	0.527	0.492	0.638
c880	0.315	0.322	0.405	0.313	0.318	0.385	0.315	0.302	0.368	0.313	0.309	0.366
c1355	0.625	0.614	0.775	0.625	0.614	0.762	0.625	0.609	0.742	0.623	0.614	0.755
c1908	0.473	0.469	0.606	0.477	0.467	0.603	0.472	0.463	0.581	0.470	0.462	0.568
c2670	0.676	0.691	0.876	0.658	0.665	0.844	0.675	0.671	0.830	0.658	0.645	0.791
c3540	0.942	0.940	1.207	0.939	0.942	1.208	0.931	0.887	1.104	0.939	0.907	1.092
c5315	1.192	1.234	1.604	1.193	1.227	1.577	1.152	1.146	1.423	1.185	1.138	1.400
c6288	2.944	2.916	3.655	2.958	2.919	3.625	2.944	2.916	3.645	2.958	2.919	3.547
c7552	1.963	1.995	2.542	1.974	1.993	2.547	1.963	1.934	2.365	1.974	1.944	2.346

Netlist	Power savings vs. single Vdd/single Vth baseline											
c17	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%	-8.8%	0.0%	-20.4%
c432	-2.4%	0.6%	-24.3%	-2.9%	0.0%	-24.3%	-2.4%	0.6%	-21.1%	-2.9%	0.0%	-22.9%
c499	-7.1%	0.0%	-30.0%	-7.1%	0.0%	-30.0%	-7.1%	0.0%	-28.1%	-7.1%	0.0%	-29.7%
c880	2.3%	0.0%	-25.7%	3.0%	1.4%	-19.4%	2.3%	6.2%	-14.3%	3.0%	4.2%	-13.5%
c1355	-1.8%	0.0%	-26.2%	-1.8%	0.0%	-24.1%	-1.8%	0.8%	-20.8%	-1.5%	0.0%	-23.0%
c1908	1.4%	2.2%	-26.3%	0.4%	2.6%	-25.7%	1.6%	3.4%	-21.1%	1.9%	3.6%	-18.5%
c2670	9.8%	7.8%	-16.8%	12.2%	11.3%	-12.6%	10.0%	10.4%	-10.7%	12.2%	13.9%	-5.5%
c3540	1.8%	2.1%	-25.7%	2.2%	1.9%	-25.8%	3.0%	7.6%	-15.0%	2.2%	5.5%	-13.7%
c5315	3.3%	-0.1%	-30.1%	3.2%	0.4%	-27.9%	6.6%	7.1%	-15.4%	3.9%	7.7%	-13.6%
c6288	0.4%	1.3%	-23.7%	-0.1%	1.2%	-22.7%	0.4%	1.3%	-23.4%	-0.1%	1.2%	-20.0%
c7552	2.3%	0.6%	-26.6%	1.7%	0.7%	-26.8%	2.3%	3.7%	-17.8%	1.7%	3.2%	-16.8%
Average	0.1%	1.3%	-25.1%	0.2%	1.8%	-23.6%	0.5%	3.7%	-18.9%	0.4%	3.6%	-18.0%

Netlist	ECVS power savings vs. CVS						
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.0%	0.0%	2.5%	0.0%	0.0%	0.0%	1.1%
c499	0.0%	0.0%	1.5%	0.0%	0.0%	0.0%	0.2%
c880	0.0%	6.2%	9.0%	0.0%	2.8%	0.0%	4.9%
c1355	0.0%	0.8%	4.3%	0.3%	0.0%	0.0%	0.9%
c1908	0.1%	1.2%	4.1%	1.4%	1.0%	0.0%	5.7%
c2670	0.2%	2.8%	5.3%	0.0%	3.0%	0.0%	6.3%
c3540	1.1%	5.6%	8.5%	0.0%	3.7%	0.0%	9.6%
c5315	3.4%	7.2%	11.3%	0.7%	7.3%	0.0%	11.2%
c6288	0.0%	0.0%	0.3%	0.0%	0.0%	0.0%	2.2%
c7552	0.0%	3.1%	7.0%	0.0%	2.5%	0.0%	7.9%
Average	0.4%	2.5%	4.9%	0.2%	1.8%	0.0%	4.5%

J.5 Multi-Vdd/multi-Vth with 1% leakage at $1.2 \times T_{min}$ with 80ps LCFFs

Table J.5 This table compares CVS dual Vdd/single Vth/sizing power minimization results versus the sizing only baseline results in Table 7.4.

Vth (V)	Dual		Triple		Dual		Triple	
	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
	0.14	0.08	0.14	0.14	0.14	0.08	0.14	0.14
CVS dual Vdd								
Vdd (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
	0.8	0.8	0.8	0.8	0.6	0.6	0.6	0.6
Netlist	Power (mW)							
c17	0.011	0.012	0.012	0.011	0.011	0.012	0.012	0.011
c432	0.210	0.223	0.225	0.208	0.208	0.223	0.223	0.208
c499	0.483	0.520	0.492	0.483	0.483	0.518	0.492	0.483
c880	0.300	0.313	0.319	0.300	0.300	0.312	0.316	0.300
c1355	0.585	0.619	0.614	0.585	0.585	0.619	0.614	0.584
c1908	0.439	0.458	0.466	0.438	0.431	0.462	0.473	0.431
c2670	0.646	0.665	0.692	0.645	0.624	0.636	0.665	0.623
c3540	0.882	0.924	0.939	0.875	0.878	0.922	0.940	0.873
c5315	1.152	1.185	1.232	1.149	1.150	1.184	1.227	1.150
c6288	2.816	2.906	2.921	2.802	2.803	2.903	2.932	2.802
c7552	1.884	1.947	1.994	1.875	1.872	1.944	1.991	1.868

Netlist	Power savings vs. single Vdd/single Vth baseline							
c17	5.0%	-2.3%	0.0%	5.0%	5.0%	-1.8%	0.0%	5.0%
c432	6.9%	1.1%	0.0%	7.7%	7.5%	1.0%	0.9%	7.5%
c499	1.8%	-5.7%	0.0%	1.9%	1.8%	-5.4%	0.0%	1.8%
c880	6.9%	3.0%	0.9%	6.9%	7.0%	3.2%	1.8%	7.0%
c1355	4.7%	-0.8%	0.0%	4.7%	4.8%	-0.8%	0.0%	4.9%
c1908	8.4%	4.6%	2.8%	8.6%	10.2%	3.5%	1.4%	10.2%
c2670	13.8%	11.2%	7.7%	14.0%	16.7%	15.2%	11.3%	16.8%
c3540	8.1%	3.7%	2.1%	8.8%	8.5%	3.9%	2.1%	9.0%
c5315	6.6%	3.9%	0.1%	6.8%	6.7%	4.0%	0.5%	6.7%
c6288	4.7%	1.6%	1.1%	5.2%	5.1%	1.7%	0.7%	5.2%
c7552	6.2%	3.0%	0.7%	6.6%	6.8%	3.2%	0.9%	7.0%
Average	6.6%	2.1%	1.4%	6.9%	7.3%	2.5%	1.8%	7.4%

Table J.6 This table compares ECVS dual Vdd/single Vth/sizing power minimization results versus the sizing only baseline results in Table 7.4. At the bottom are shown the ECVS power savings versus CVS.

	Dual		Triple		Dual		Triple	
Vth (V)	0.23 0.14 0.08							
ECVS dual Vdd								
Vdd (V)	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.6
Netlist								
Netlist	Power (mW)							
c17	0.011	0.012	0.012	0.011	0.011	0.012	0.012	0.011
c432	0.210	0.223	0.225	0.208	0.208	0.223	0.223	0.208
c499	0.483	0.518	0.492	0.483	0.483	0.518	0.492	0.483
c880	0.289	0.307	0.298	0.289	0.299	0.310	0.306	0.292
c1355	0.585	0.615	0.607	0.584	0.585	0.618	0.612	0.584
c1908	0.434	0.458	0.457	0.434	0.431	0.462	0.457	0.431
c2670	0.637	0.660	0.669	0.627	0.610	0.636	0.641	0.610
c3540	0.858	0.903	0.886	0.856	0.860	0.915	0.897	0.855
c5315	1.097	1.140	1.144	1.091	1.086	1.127	1.120	1.062
c6288	2.816	2.906	2.920	2.802	2.803	2.903	2.885	2.803
c7552	1.870	1.947	1.913	1.853	1.846	1.934	1.936	1.845

Netlist	Power savings vs. single Vdd/single Vth baseline							
c17	5.0%	-2.3%	0.0%	5.0%	5.0%	-1.8%	0.0%	5.0%
c432	7.0%	1.1%	0.1%	7.7%	7.5%	1.0%	0.9%	7.5%
c499	1.8%	-5.2%	0.0%	1.9%	1.8%	-5.3%	0.0%	1.8%
c880	10.4%	4.7%	7.5%	10.4%	7.3%	3.9%	5.0%	9.3%
c1355	4.7%	-0.1%	1.2%	4.9%	4.8%	-0.6%	0.4%	4.9%
c1908	9.4%	4.6%	4.8%	9.4%	10.2%	3.6%	4.7%	10.2%
c2670	15.1%	12.0%	10.7%	16.4%	18.6%	15.2%	14.5%	18.6%
c3540	10.6%	5.9%	7.7%	10.8%	10.4%	4.6%	6.5%	10.9%
c5315	11.0%	7.5%	7.2%	11.5%	11.9%	8.6%	9.2%	13.9%
c6288	4.7%	1.6%	1.2%	5.2%	5.1%	1.7%	2.4%	5.1%
c7552	6.9%	3.0%	4.7%	7.7%	8.1%	3.7%	3.6%	8.1%
Average	7.9%	3.0%	4.1%	8.3%	8.2%	3.2%	4.3%	8.7%

Netlist	ECVS power savings vs. CVS							
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.1%	0.0%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
c499	0.0%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c880	3.8%	1.7%	6.7%	3.8%	0.3%	0.7%	3.3%	2.5%
c1355	0.0%	0.7%	1.2%	0.2%	0.0%	0.1%	0.4%	0.0%
c1908	1.0%	0.0%	2.0%	0.8%	0.0%	0.1%	3.3%	0.0%
c2670	1.5%	0.8%	3.2%	2.8%	2.3%	0.0%	3.6%	2.2%
c3540	2.7%	2.3%	5.7%	2.1%	2.1%	0.7%	4.5%	2.1%
c5315	4.8%	3.8%	7.1%	5.0%	5.5%	4.8%	8.7%	7.7%
c6288	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.6%	0.0%
c7552	0.8%	0.0%	4.1%	1.2%	1.4%	0.5%	2.7%	1.2%
Average	1.3%	0.9%	2.7%	1.5%	1.1%	0.6%	2.6%	1.4%

J.6 Vdd=0.8V/multi-Vth results with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs

The results with single Vdd of 0.8V assuming 0ps LCFF delay overhead are shown in Table J.7.

Table J.7 This table compares Vdd=0.8V results versus the best sizing only results with 1.2V input drivers from Table 7.4 and the second column here. The input drivers for these Vdd=0.8V results had voltage swing of 0.8V or 1.2V.

Vth (V)	with 1.2V drivers			with 0.8V drivers		
	Single	Dual	Triple	Single	Dual	Triple
	0.08	0.08	0.08	0.08	0.08	0.08
Netlist	Power (mW)					
c17	failed delay constraint			0.011	0.011	0.009
c432	0.245	0.245	0.220	0.218	0.213	0.212
c499	0.506	0.489	0.447	0.442	0.454	0.443
c880	0.341	0.294	0.268	0.268	0.294	0.259
c1355	0.709	0.621	0.573	0.573	0.568	0.557
c1908	0.536	0.442	0.405	0.400	0.436	0.407
c2670	0.719	0.651	0.601	0.599	0.695	0.559
c3540	0.957	0.870	0.795	0.791	0.874	0.787
c5315	1.185	1.019	0.970	0.964	1.028	0.930
c6288	2.858	2.785	2.586	2.586	2.690	2.449
c7552	1.937	1.733	1.588	1.579	1.706	1.587

Netlist	Power savings vs. single Vdd/single Vth baseline with 1.2V drivers						
c17	failed delay constraint			3.3%	3.3%	24.4%	24.4%
c432	-8.6%	-8.7%	2.3%	3.4%	5.5%	6.0%	14.0%
c499	-2.7%	0.7%	9.2%	10.1%	7.8%	10.0%	19.7%
c880	-5.7%	8.9%	16.7%	16.7%	8.8%	19.5%	27.7%
c1355	-15.5%	-1.1%	6.7%	6.7%	7.5%	9.4%	18.2%
c1908	-11.8%	7.8%	15.5%	16.5%	9.0%	15.1%	24.1%
c2670	0.0%	9.4%	16.3%	16.6%	3.2%	22.2%	28.6%
c3540	0.0%	9.2%	16.9%	17.3%	8.7%	17.8%	24.3%
c5315	0.0%	14.0%	18.1%	18.6%	13.2%	21.4%	27.7%
c6288	0.0%	2.6%	9.5%	9.5%	5.9%	5.9%	14.3%
c7552	0.0%	10.5%	18.0%	18.5%	11.9%	18.1%	25.1%
Average	-4.4%	5.3%	12.9%	13.4%	7.7%	13.5%	22.5%

J.7 Multi-Vdd/single Vth with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs

On average there is only a 0.9% reduction in power for the multi-Vdd/single Vth results with 0ps LCFF delay overhead versus assuming 80ps delay overhead, and the maximum reduction in power is 8.8% as listed in Table J.8.

Table J.8 This table compares CVS and ECVS dual Vdd/single Vth/sizing power minimization results with 0ps LCFF delay overhead versus the results 80ps overhead in Table J.4. At the bottom are shown the ECVS power savings versus CVS with 0ps LCFF delay overhead.

Vth (V)	Single											
	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08	0.23	0.14	0.08
Vdd (V)	CVS dual Vdd with 1.2V drivers						ECVS dual Vdd with 1.2V drivers					
	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6
Netlist	Power (mW)											
c17	0.013	0.012	0.014	0.013	0.012	0.014	0.013	0.012	0.014	0.013	0.012	0.014
c432	0.230	0.224	0.280	0.232	0.225	0.278	0.230	0.224	0.273	0.232	0.225	0.276
c499	0.527	0.492	0.615	0.527	0.492	0.640	0.527	0.492	0.615	0.527	0.492	0.638
c880	0.309	0.298	0.369	0.313	0.305	0.371	0.309	0.298	0.353	0.313	0.305	0.355
c1355	0.625	0.613	0.768	0.625	0.614	0.762	0.625	0.608	0.724	0.623	0.614	0.747
c1908	0.469	0.465	0.591	0.468	0.463	0.591	0.466	0.456	0.580	0.463	0.452	0.562
c2670	0.676	0.685	0.867	0.656	0.665	0.836	0.675	0.671	0.830	0.656	0.645	0.788
c3540	0.942	0.938	1.192	0.939	0.932	1.197	0.931	0.887	1.101	0.938	0.907	1.092
c5315	1.156	1.169	1.475	1.178	1.209	1.504	1.152	1.146	1.409	1.175	1.115	1.350
c6288	2.937	2.908	3.655	2.952	2.919	3.625	2.937	2.908	3.645	2.952	2.919	3.547
c7552	1.954	1.958	2.510	1.974	1.986	2.504	1.954	1.895	2.342	1.974	1.934	2.305

Netlist	Saved vs. CVS with 80ps LCFFs						Saved vs. ECVS with 80ps LCFFs					
	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.1%	0.0%	0.0%	0.0%	0.0%	0.6%	0.1%	0.0%	0.0%	0.0%	0.0%	0.4%
c499	0.0%	0.0%	3.9%	0.0%	0.0%	0.0%	0.0%	0.0%	2.5%	0.0%	0.0%	0.0%
c880	2.0%	7.6%	8.8%	0.0%	4.1%	3.6%	2.0%	1.4%	4.3%	0.0%	1.4%	3.0%
c1355	0.0%	0.2%	0.9%	0.0%	0.0%	0.0%	0.0%	0.1%	2.4%	0.0%	0.0%	1.1%
c1908	0.8%	0.9%	2.4%	2.0%	0.9%	1.9%	1.2%	1.5%	0.0%	1.6%	2.2%	1.0%
c2670	0.0%	0.9%	1.1%	0.2%	0.0%	1.0%	0.0%	0.0%	0.0%	0.2%	0.0%	0.3%
c3540	0.0%	0.1%	1.2%	0.0%	1.0%	0.9%	0.0%	0.0%	0.2%	0.1%	0.0%	0.0%
c5315	3.0%	5.3%	8.0%	1.2%	1.5%	4.7%	0.0%	0.0%	1.0%	0.8%	2.0%	3.6%
c6288	0.2%	0.3%	0.0%	0.2%	0.0%	0.0%	0.2%	0.3%	0.0%	0.2%	0.0%	0.0%
c7552	0.4%	1.9%	1.3%	0.0%	0.3%	1.7%	0.4%	2.0%	1.0%	0.0%	0.5%	1.7%
Average	0.6%	1.6%	2.5%	0.3%	0.7%	1.3%	0.4%	0.5%	1.0%	0.3%	0.6%	1.0%

Saved vs. CVS with 0ps LCFFs						
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	2.5%	0.0%	0.0%	0.0%	0.9%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
0.0%	0.0%	4.5%	0.0%	0.1%	4.2%	
0.0%	0.7%	5.7%	0.3%	0.0%	1.9%	
0.5%	1.9%	1.8%	1.1%	2.3%	4.8%	
0.2%	1.9%	4.3%	0.0%	3.0%	5.7%	
1.1%	5.5%	7.7%	0.1%	2.7%	8.8%	
0.3%	2.0%	4.5%	0.2%	7.8%	10.2%	
0.0%	0.0%	0.3%	0.0%	0.0%	2.2%	
0.0%	3.2%	6.7%	0.0%	2.6%	7.9%	
0.2%	1.4%	3.5%	0.2%	1.7%	4.3%	

J.8 Multi-Vdd/multi-Vth with 1% leakage at $1.2 \times T_{\min}$ with 0ps LCFFs

These results are with 0ps LCFF delay penalty and 1.2V input driver voltage swing. Switching activity was multiplied by a fraction such that leakage is 1.0% of total power at $V_{dd}=1.2V/V_{th}=0.23V$ for the Design Compiler delay minimized netlists.

Table J.9 CVS dual Vdd/multi-Vth/sizing power minimization results with 0ps LCFF delay overhead. At the bottom are shown the power savings versus the corresponding results with 80ps LCFF delay overhead.

	Dual		Triple		Dual		Triple	
Vth (V)	0.23 0.14 0.08							
CVS dual Vdd with 1.2V drivers								
Vdd (V)	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.6
Netlist								
Power (mW)								
c17	0.011	0.012	0.012	0.011	0.011	0.012	0.012	0.011
c432	0.209	0.223	0.224	0.208	0.208	0.223	0.223	0.208
c499	0.473	0.507	0.492	0.471	0.483	0.512	0.492	0.475
c880	0.283	0.291	0.297	0.283	0.292	0.296	0.298	0.283
c1355	0.581	0.615	0.603	0.573	0.584	0.619	0.609	0.582
c1908	0.436	0.458	0.462	0.433	0.431	0.458	0.465	0.431
c2670	0.643	0.661	0.685	0.642	0.610	0.636	0.663	0.610
c3540	0.877	0.924	0.937	0.873	0.860	0.920	0.937	0.855
c5315	1.082	1.140	1.149	1.082	1.086	1.141	1.177	1.062
c6288	2.799	2.903	2.910	2.800	2.800	2.903	2.933	2.795
c7552	1.850	1.937	1.963	1.832	1.846	1.922	1.966	1.835
Netlist								
Saved vs. CVS with 80ps LCFFs								
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.2%	0.0%	0.7%	0.0%	0.4%	0.0%	0.0%	0.4%
c499	2.2%	2.5%	0.0%	2.3%	0.0%	1.3%	0.0%	1.8%
c880	5.7%	6.8%	7.0%	5.6%	2.1%	5.1%	5.8%	3.2%
c1355	0.7%	0.6%	1.8%	2.0%	0.0%	0.0%	0.8%	0.4%
c1908	0.8%	0.0%	0.8%	1.2%	0.0%	0.9%	1.6%	0.0%
c2670	0.5%	0.6%	1.0%	0.3%	0.0%	0.0%	0.4%	0.0%
c3540	0.5%	0.0%	0.2%	0.2%	0.0%	0.3%	0.3%	0.0%
c5315	6.1%	3.8%	6.8%	5.8%	0.0%	3.6%	4.1%	0.0%
c6288	0.6%	0.1%	0.4%	0.0%	0.1%	0.0%	0.0%	0.3%
c7552	1.8%	0.5%	1.5%	2.3%	0.0%	1.1%	1.2%	0.5%
Average	1.7%	1.4%	1.8%	1.8%	0.2%	1.1%	1.3%	0.6%

Table J.10 ECVS dual Vdd/multi-Vth/sizing power minimization results with 0ps LCFF delay overhead. In the middle are shown the power savings versus the corresponding results with 80ps LCFF delay overhead. At the bottom are shown the power savings versus the corresponding CVS results with 0ps LCFF delay overhead.

Vth (V)	Dual			Triple	Dual			Triple
	0.23	0.23		0.23	0.23	0.23		0.23
	0.14	0.14	0.08	0.14	0.14	0.08	0.08	0.14
ECVS dual Vdd with 1.2V drivers								
Vdd (V)	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.8	1.2 0.6	1.2 0.6	1.2 0.6	1.2 0.6
Netlist	Power (mW)							
c17	0.011	0.012	0.012	0.011	0.011	0.012	0.012	0.011
c432	0.209	0.223	0.224	0.208	0.208	0.223	0.223	0.208
c499	0.473	0.507	0.492	0.471	0.483	0.512	0.492	0.475
c880	0.275	0.291	0.277	0.276	0.292	0.296	0.295	0.283
c1355	0.574	0.615	0.594	0.571	0.584	0.618	0.609	0.582
c1908	0.434	0.458	0.451	0.433	0.431	0.458	0.448	0.428
c2670	0.637	0.655	0.663	0.627	0.610	0.636	0.638	0.606
c3540	0.858	0.903	0.883	0.846	0.855	0.912	0.893	0.855
c5315	1.082	1.106	1.113	1.076	1.085	1.113	1.103	1.062
c6288	2.799	2.903	2.908	2.800	2.800	2.903	2.885	2.795
c7552	1.844	1.937	1.903	1.826	1.845	1.922	1.905	1.827
Netlist	Saved vs. ECVS with 80ps LCFFs							
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.1%	0.0%	0.6%	0.0%	0.4%	0.0%	0.0%	0.4%
c499	2.2%	2.1%	0.0%	2.3%	0.0%	1.2%	0.0%	1.8%
c880	4.7%	5.2%	6.9%	4.5%	2.1%	4.4%	3.5%	3.2%
c1355	1.9%	0.0%	2.1%	2.3%	0.0%	0.1%	0.4%	0.4%
c1908	0.0%	0.0%	1.2%	0.3%	0.0%	0.8%	1.9%	0.5%
c2670	0.0%	0.7%	0.9%	0.0%	0.0%	0.0%	0.5%	0.6%
c3540	0.0%	0.0%	0.3%	1.2%	0.6%	0.4%	0.5%	0.0%
c5315	1.4%	3.0%	2.7%	1.4%	0.2%	1.2%	1.4%	0.0%
c6288	0.6%	0.1%	0.4%	0.0%	0.1%	0.0%	0.0%	0.3%
c7552	1.4%	0.5%	0.5%	1.5%	0.0%	0.6%	1.6%	1.0%
Average	1.1%	1.1%	1.4%	1.2%	0.3%	0.8%	0.9%	0.7%
Netlist	Saved vs. CVS with 0ps LCFFs							
c17	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c432	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c499	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
c880	2.7%	0.0%	6.6%	2.6%	0.0%	0.0%	0.8%	0.0%
c1355	1.2%	0.0%	1.5%	0.5%	0.0%	0.2%	0.0%	0.0%
c1908	0.2%	0.0%	2.5%	0.0%	0.0%	0.0%	3.6%	0.5%
c2670	1.0%	1.0%	3.2%	2.5%	0.0%	0.0%	3.7%	0.6%
c3540	2.2%	2.3%	5.8%	3.1%	0.6%	0.8%	4.7%	0.0%
c5315	0.0%	2.9%	3.1%	0.6%	0.2%	2.4%	6.2%	0.0%
c6288	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%	1.6%	0.0%
c7552	0.3%	0.0%	3.1%	0.3%	0.0%	0.0%	3.1%	0.4%
Average	0.7%	0.6%	2.3%	0.9%	0.1%	0.3%	2.2%	0.1%

Let's examine the results in detail with a single V_{th} of 0.08V. The single $V_{dd}=0.8V$ /single $V_{th}=0.08V$ with 1.2V input driver results were on average only 4.4% worse than the best single $V_{dd}/single\ V_{th}$ results with 0ps LCFF delay overhead. Whereas the CVS dual $V_{dd}=1.2V\&0.8V$ /single $V_{th}=0.08V$ results in Table 7.8 were on average 23.6% worse. Table J.11 compares the single $V_{dd}=0.8V$ results with the dual $V_{dd}=1.2V\&0.8V$ ECVS results. In green are shown the ECVS dual V_{dd} results after the LP approach has run assuming no power consumption for the level converters. The only cases where those results are lower power than the single $V_{dd}=0.8V$ results are c880 and c1908 (highlighted in blue text). On average these results were 7.9% worse. After using the correct level converter power and running further iterations of the LP approach, the power is 6.5% worse on average, and none of the results are better than for single $V_{dd}=0.8V$. With the power overhead for the level converters included, the later optimization runs on average halve the number of level converters. However for c5315, the delay reduction phase of the LP approach failed after the first power minimization run, and so the result given is simply the correct power after running optimization with zero level converter power overhead. Correcting the level converter power for c5315 increases the total power by 9.9%. As observed previously, ECVS provides more power savings than CVS. The ECVS results with correct level converter power average 3.2% lower power than the corresponding CVS results with $V_{dd}=1.2V\&0.8V/V_{th}=0.08V$ in Table 7.8, except for c6288, where they are 0.8% worse.

The results show that power minimization with the linear programming approach for multi- V_{dd} can be quite suboptimal. The ECVS multi- $V_{dd}/V_{th}=0.08V$ results average 14.8% worse than the $V_{dd}=0.8V/V_{th}=0.08V$ results. Restricting to $V_{dd}=0.8V$ only is a subspace of the space of possible multi- V_{dd} solutions. However, the multi- V_{dd} optimization gets stuck in a local minimum with only 34% of gates at $V_{dd}=0.8V$ on average, and fails to traverse to the solution with all gates at $V_{dd}=0.8V$. Note that for all the dual V_{dd} netlists, except c17 which uses $V_{dd}=1.2V$ gates exclusively, the majority of the outputs are driven by $V_{dd}=0.8V$ gates with level

converter flip-flops (% outputs with LCFFs). Despite having set the level converter power to zero and having no delay overhead for the level converter flip-flops, there is still an optimization barrier. More gates changing to 0.8V is impeded by the level converter delay and because VDDL gates have substantially larger rise delay when the input voltage swing is VDDH. This problem was discussed in detail in Section 7.4.4. The optimization space is very “hilly” and it is very difficult to surmount the delay barrier. This is a significant limitation for our optimization approach and it suggests that our other multi-Vdd results may be suboptimal, despite the fact that we outperform other multi-Vdd approaches. Setting the level converter delay overhead to zero probably won’t be beneficial, given the problems reducing delay after the delay constraint is violated substantially, as occurred for c5315. A stronger delay minimization approach, such as the TILOS-like delay minimization in Design Compiler may be able to provide sufficient delay reduction, but we have not had the opportunity to experiment with that.

We will now examine several more sets of results, and look at a few different parameter choices in optimization to see how they influence the results.

Table J.11 This table compares ECVS dual Vdd=1.2V&0.8V/single Vth=0.08V/sizing results versus single Vdd=0.8V/single Vth=0.08V. In green are shown the ECVS results with zero power for the level converters. The ECVS results after correcting the level converter power and running further optimization are shown on the right. In most cases, the power was worse than the single Vdd=0.8V results. For c5315, the LP delay reduction phase failed to reduce delay below the delay constraint after the first power minimization iteration with correct LC power. Consequently, there was no reduction in the number of level converters (# LCs) for c5315. The two cases where ECVS with zero power for the level converters has lower power than the single Vdd=0.8V results are highlighted in blue.

Circuit	# outputs	# gates	Vdd=0.8V	ECVS (negated LC power)				ECVS (correct LC power)			
			Power (mW)	Power (mW)	# LCs	% outputs with LCFFs	% gates at VDDL	Power (mW)	# LCs	% outputs with LCFFs	% gates at VDDL
c17	2	10	failed	0.014	0	0.0%	0.0%	0.014	0	0.0%	0.0%
c432	7	259	0.245	0.259	42	57.1%	20.5%	0.276	17	57.1%	13.5%
c499	32	644	0.506	0.611	29	100.0%	29.0%	0.623	14	100.0%	27.3%
c880	26	484	0.341	0.333	44	84.6%	57.2%	0.353	12	92.3%	61.4%
c1355	32	764	0.709	0.731	37	96.9%	45.3%	0.724	21	96.9%	44.2%
c1908	25	635	0.536	0.524	97	68.0%	34.2%	0.580	47	72.0%	30.1%
c2670	139	1164	0.719	0.802	106	89.2%	51.9%	0.830	44	88.5%	47.9%
c3540	22	1283	0.957	1.027	166	72.7%	41.5%	1.101	69	72.7%	37.5%
c5315	123	1956	1.185	1.282	208	90.2%	57.3%	1.409	208	90.2%	57.3%
c6288	32	3544	2.858	3.314	698	90.6%	25.3%	3.653	203	90.6%	15.0%
c7552	86	2779	1.937	2.134	365	87.2%	44.8%	2.342	235	87.2%	41.4%

Circuit	Power worse than Vdd=0.8V by		After correcting LC power (comparing ECVS negated LC power vs. correct LC power)		
	ECVS (negated LC power)	ECVS (correct LC power)	% increase in power		% decrease in # LCs
			% increase in power	% decrease in # LCs	
c17	Vdd = 0.8V failed delay constraint		0.0%		0%
c432	5.8%	12.9%	6.7%		60%
c499	21.0%	23.1%	1.8%		52%
c880	-2.1%	3.5%	5.8%		73%
c1355	3.1%	2.1%	-0.9%		43%
c1908	-2.3%	8.3%	10.8%		52%
c2670	11.6%	15.5%	3.5%		58%
c3540	7.3%	15.0%	7.2%		58%
c5315	8.2%	19.0%	9.9%		0%
c6288	15.9%	27.8%	10.2%		71%
c7552	10.2%	20.9%	9.7%		36%
Average	7.9%	14.8%	6.5%		50%

The results presented thus far were mostly for the conservative slew analysis setting. The initial optimization setting is to use the worst timing arc Δ delay when determining the best alternate cell choices. These optimization settings are described in Appendix G.1. Also, optimization has always started with the fastest initial circuit with high Vdd and low Vth. These choices generally provided the best results.

For the multi-Vth and multi-Vdd results, the conservative slew analysis setting gave results on average within 1% to 2% of the best solution found, excluding the single Vdd=0.8V results, whereas there were cases where the aggressive slew analysis setting gave results on average 4% worse. For the ECVS results with Vdd=1.2V&0.8V/Vth=0.14V&0.08V in Table J.12, the aggressive slew analysis setting gave results on average 1% worse than the conservative slew analysis setting (comparing columns seven and nine), though for c5315 the power was 2.1% lower. For c17, which has only ten gates, some results were suboptimal by up to 20%, as one or two gates changing or not (depending on optimization parameters and available Vth values) substantially affects the results. For the other benchmarks at $1.2 \times T_{\min}$, the most suboptimal conservative slew result seen was 3.7% worse, but other results were within 2% of the best result found. For the other benchmarks at $1.0 \times T_{\min}$, there were a couple of cases where either slew analysis setting was suboptimal by 20%. The worse suboptimal results were due to the delay reduction phase of the LP approach having problems meeting the delay constraint.

In Table J.12, the only cases where the ECVS results are better is 2.4% power saving at Vdd=1.2V&0.6V/triple Vth for c432, and for c17 where single Vdd=0.8V could not satisfy the delay constraint. On average, the multi-Vdd results are 6% to 11% worse than the single Vdd=0.8V results. We focused on trying different optimization parameters with the Vdd=1.2V&0.8V/Vth=0.14V&0.08V, to try and find the good single Vdd=0.8V results that exist within the solution space.

As discussed in Section 7.4.4, when the input voltage to a gate is higher than the supply voltage, the rise delay of the gate is substantially worse. When comparing $\Delta P/\Delta d$ for alternate cells to determine the best alternate cell to encode in the linear program, the penalty for changing Vdd is less if we compare the total of timing arc Δ delays, than if we use the worst Δ delay on a timing arc. The “total Δ delays” columns of Table J.12 give results for starting with the optimization parameter compareTotalTimingArcDeltaDelays set true, and the “worst Δ delay” corresponds to

compareTotalTimingArcDeltaDelays set false. If we start at Vdd=1.2V, using “total Δdelays” does help more gates change to Vdd=0.8V: 9.9% more gates end up at VDDL, but the results are 0.9% worse on average. If we start at Vdd=0.8V, using “total Δdelays” does result in several more gates changing to Vdd=1.2V, but still the vast majority of gates remain at Vdd=0.8V and again the results are 1% worse.

Starting multi-Vdd optimization with all gates at Vdd=0.8V/Vth=0.08V provides the closest results to the single Vdd=0.8V results. Excluding the outlier for c1908 (shown in red) where the LP delay reduction phase failed to satisfy the delay constraint after two power minimization iterations, the multi-Vdd results comparing the worst Δdelay average only 2.8% worse than the single Vdd=0.8V results. Indeed, at most two gates are changed to Vdd=1.2V. This strongly suggests that Vdd=1.2V is a poor choice for the high supply voltage at a delay constraint of $1.2 \times T_{\min}$ assuming we have 0.14V and 0.08V threshold voltages. This is contrary to the results in [188], which failed to consider using a lower threshold voltage that would enable reducing Vdd to 0.8V, though the University of Michigan researchers had the same libraries available to them. Finding such a solution can require trying a range of values for a single Vdd in conjunction with a variety of dual Vth values, as we have done for the results in this chapter.

Table J.12 This table compares ECVS dual Vdd/multi-Vth versus single Vdd=0.8V/multi-Vth results with 0ps LCFF delay overhead from Table 7.7. Using the worst timing arc Δ delay and total Δ delay across timing arcs was tried, along with starting all gates at Vdd=0.8V. Only one result in blue was better than the single Vdd=0.8V results. Two results in red were substantially worse.

Slew analysis setting			conservative			aggressive	conservative		
			start at Vdd=1.2V				start at Vdd=0.8V		
Baseline	Dual Vth	Triple Vth	worst Δ delay	worst Δ delay	total Δ delays	worst Δ delay	worst Δ delay	total Δ delays	
Vth (V)	varies		0.23	0.23					
		0.14	0.14	0.14	0.14	0.14	0.14	0.14	
		0.08	0.08	0.08	0.08	0.08	0.08	0.08	
Single Vdd			Dual Vdd						
Vdd (V)	1.2	0.8	0.8	1.2	1.2	1.2	1.2	1.2	
Vdd (V)	0.8		0.6	0.6	0.8	0.8	0.8	0.8	
Netlist	1.2xTmin (ns)	Power (mW)							
c17	0.112	0.012	failed	failed	0.011	0.012	0.017	0.012	
c432	0.879	0.225	0.220	0.218	0.213	0.225	0.228	0.229	
c499	0.842	0.492	0.447	0.442	0.492	0.505	0.501	0.501	
c880	0.840	0.322	0.268	0.268	0.293	0.277	0.277	0.286	
c1355	0.934	0.614	0.573	0.573	0.584	0.594	0.590	0.601	
c1908	1.199	0.479	0.405	0.400	0.428	0.451	0.466	0.459	
c2670	0.779	0.719	0.601	0.599	0.606	0.663	0.682	0.669	
c3540	1.265	0.957	0.795	0.791	0.857	0.883	0.888	0.887	
c5315	1.135	1.185	0.970	0.964	1.066	1.113	1.122	1.090	
c6288	3.965	2.858	2.586	2.586	2.796	2.908	2.943	3.002	
c7552	1.016	1.937	1.588	1.579	1.827	1.903	1.895	1.914	
Power worse than Vdd=0.8V/triple Vth by			Power worse than Vdd=0.8V/Vth=0.14V&0.08V by						
c17				single Vdd=0.8V failed delay constraint					
c432				-2.4%	2.4%	3.5%	4.1%	3.6%	8.2%
c499				11.2%	13.0%	12.1%	12.1%	2.4%	3.0%
c880				9.0%	3.3%	3.3%	6.5%	7.3%	8.6%
c1355				1.9%	3.8%	3.0%	4.9%	1.2%	1.2%
c1908				7.1%	11.4%	15.0%	13.4%	38.6%	27.5%
c2670				1.1%	10.2%	13.3%	11.3%	3.4%	10.3%
c3540				8.3%	11.1%	11.6%	11.5%	2.5%	4.6%
c5315				10.6%	14.7%	15.7%	12.3%	3.5%	8.1%
c6288				8.1%	12.4%	13.8%	16.1%	0.4%	0.5%
c7552				15.7%	19.8%	19.3%	20.5%	1.0%	2.1%
Average				7.1%	10.2%	11.1%	11.3%	6.4%	7.4%
Netlist Percentage of gates at VDDL									
c17	0.0%	0.0%	40.0%	0.0%	failed	50.0%			
c432	7.3%	7.7%	8.9%	8.5%	99.6%	99.6%			
c499	6.7%	21.0%	32.5%	24.1%	100.0%	99.7%			
c880	42.1%	62.2%	67.8%	55.2%	100.0%	99.2%			
c1355	9.0%	22.4%	37.3%	28.0%	100.0%	99.9%			
c1908	21.7%	28.3%	38.6%	35.3%	100.0%	99.4%			
c2670	42.4%	47.6%	51.0%	46.0%	100.0%	99.1%			
c3540	29.7%	39.2%	38.7%	36.6%	99.9%	99.8%			
c5315	46.3%	53.2%	60.9%	58.6%	100.0%	99.9%			
c6288	13.5%	15.7%	18.8%	15.1%	99.9%	99.9%			
c7552	32.0%	35.9%	47.6%	37.2%	100.0%	99.9%			
Average	22.8%	30.3%	40.2%	31.3%	99.9%	95.1%			

J.9 Computation runtimes with multi-Vdd and multi-Vth

Computation runtimes for 40 iterations of optimization with single and multiple Vdd and Vth values are listed in Table J.13. These runtimes listed are representative of the range of runtimes typically seen. Benchmark c17 was omitted due to its small size (10 gates) and small runtimes (0.9 to 3.3 seconds). Runtimes for the Huffman, and larger SOVA_EPR4 and R4_SOVA benchmarks are included in though leakage was not normalized to 1% of total power at Vdd=1.2V/Vth=0.23V for these runs.

Table J.13 This table shows the computation runtimes for 40 iterations of optimization with the linear programming approach on the ISCAS'85 benchmarks. A variety of runs with single and multiple Vdd and Vth values are shown, and are representative of the typical runtimes seen. In the middle are listed the runtimes with the linear program solver, and at the bottom are listed the LP solver runtimes. Runtimes in blue were for runs with 0.8V input drivers; the other runs had 1.2V drivers.

Circuit	# logic levels	# gates	Single Vdd 0ps LCFF delay			CVS Dual Vdd 0ps LCFF delay			ECVS Dual Vdd 80ps LCFF delay				
			Vdd (V)	1.2	0.8	0.8	Vdd (V)	1.2	1.2	1.2	Vdd (V)	1.2	1.2
			Vth (V)	0.23	0.14	0.23	0.14	0.08	0.08	0.08	Vth (V)	0.23	0.14
Total Runtime (s)													
c432	24	259	85	82	119	76	74	108	71	120	200		
c880	23	484	143	117	161	145	139	216	144	215	368		
c1908	33	635	205	178	253	173	215	284	217	336	497		
c499	25	644	188	149	182	141	155	247	161	231	373		
c1355	27	764	205	198	256	204	224	322	219	326	539		
Huffman	29	774	125	207	268	125	193	276	193	346	320		
c2670	23	1164	291	254	340	256	249	377	266	425	676		
c3540	36	1283	541	588	777	567	512	761	651	953	1432		
c5315	34	1956	568	561	803	629	614	879	655	1079	1480		
c7552	31	2779	947	1062	1374	997	916	1279	1128	1682	2374		
c6288	113	3544	2593	3610	4091	3796	2725	3367	5111	4451	6381		
SOVA_EPR4	110	15686	6458	8201	9899	5792	6297	8306	9063	11298	12492		
R4_SOVA	144	33344	36758	146024	270904	25457	28639	64683	73148	168055	300132		
Circuit			Runtime for everything except the LP solver (s)										
c432	66	64	101	58	57	91	52	102	181				
c880	110	84	127	111	106	184	109	180	333				
c1908	153	120	194	112	162	231	154	271	435				
c499	141	97	130	89	107	200	107	177	321				
c1355	144	130	191	137	164	261	144	254	474				
Huffman	64	145	206	65	134	217	130	285	258				
c2670	209	166	251	171	166	293	177	334	589				
c3540	377	315	502	333	312	546	378	653	1148				
c5315	402	348	589	434	440	702	431	840	1248				
c7552	661	544	868	632	606	955	678	1163	1929				
c6288	1032	911	1296	778	878	1449	1000	1622	2643				
SOVA_EPR4	2030	3215	4643	2079	2803	4403	3749	6156	7010				
R4_SOVA	6159	10927	16693	6179	9557	14144	12764	20063	31240				
Circuit			Runtime for the linear program solver (s)										
c432	18	21	21	21	17	17	22	21	21				
c880	34	39	39	39	32	32	39	40	40				
c1908	51	65	66	67	52	54	69	72	69				
c499	48	57	57	57	48	47	60	60	58				
c1355	62	75	73	75	60	61	82	80	72				
Huffman	61	62	62	60	59	59	64	62	62				
c2670	82	100	99	95	83	84	98	102	97				
c3540	165	289	292	251	200	215	288	316	299				
c5315	166	236	237	217	174	176	245	261	255				
c7552	286	552	542	397	309	324	483	554	479				
c6288	1,561	2,746	2,844	3,062	1,847	1,918	4,155	2,874	3,784				
SOVA_EPR4	4,428	4,986	5,256	3,713	3,494	3,903	5,314	5,141	5,482				
R4_SOVA	30,599	135,098	254,211	19,277	19,082	50,539	60,383	147,991	268,892				

References

- [1] Aho, A., Johnson, S., and Ullman, J, “Code Generation for Expressions with Common Subexpressions,” *Journal of the ACM*, vol. 24., no. 1, 1976, pp. 146-160.
- [2] AMD, AMD Athlon Processor – Technical Brief, December 1999,
<http://www.amd.com/products/cpg/athlon/techdocs/pdf/22054.pdf>
- [3] AMD, AMD Processors for Desktops: AMD Athlon 64 Processor and AMD Sempron Processor, September 2006. <http://www.amdcompare.com/us-en/desktop/>
- [4] AMD, AMD Turion 64 Mobile Technology Model Number, Thermal Design Power, Frequency, and L2 Cache Comparison, September 2006.
http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_12651_12658,00.html
- [5] AMD, Processor Pricing, September 2006.
http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_609,00.html?redir=CPPR01
- [6] Anderson, F., Wells, J., and Berta, E., “The Core Clock System on the Next Generation Itanium™ Microprocessor,” *Digest of Technical Papers of the IEEE International Solid-State Circuits Conference*, 2002, pp. 146-147, 453.
- [7] ARM, ARM926EJ-S, 2006. <http://www.arm.com/products/CPUs/ARM926EJ-S.html>
- [8] ARM, ARM1026EJ-S, 2006. <http://www.arm.com/products/CPUs/ARM1026EJS.html>
- [9] ARM, ARM1136J(F)S, 2006. <http://www.arm.com/products/CPUs/ARM1136JF-S.html>
- [10] ARM, ARM Cortex-A8, 2006. http://www.arm.com/products/CPUs/ARM_Cortex-A8.html
- [11] ARM, ARM Cortex-R4, 2006. http://www.arm.com/products/CPUs/ARM_Cortex-R4.html
- [12] ARM, ARM Processor Cores.
[http://www.armdevzone.com/open.nsf/htmlall/A944EB65693A4EB180256A440051457A/\\$File/ARM+cores+111-1.pdf](http://www.armdevzone.com/open.nsf/htmlall/A944EB65693A4EB180256A440051457A/$File/ARM+cores+111-1.pdf)
- [13] ARM, ARM Cortex-A8, 2005.
http://www.arm.com/products/CPUs/ARM_Cortex-A8.html
- [14] Avant!, *Star-Hspice Manual*, 1998, 1714 pp.
- [15] Bai, M., and Sylvester, D., “Analysis and Design of Level-Converting Flip-Flops for Dual-V_{dd}/V_{th} Integrated Circuits,” *IEEE International Symposium on System-on-Chip*, 2003, pp. 151-154.
- [16] Benschneider, B.J., et al., “A 300-MHz 64-b Quad-Issue CMOS RISC Microprocessor,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, November 1995, pp. 1203-1214.
- [17] Bhavnagarwala, A., et al., “A Minimum Total Power Methodology for Projecting Limits on CMOS GSI,” *IEEE Transactions on VLSI Systems*, vol. 8, no. 3, June 2000, pp. 235-251.
- [18] Bisdounis, L., et al., “A comparative study of CMOS circuit design styles for low-power high-speed VLSI circuits,” *International Journal of Electronics*, vol. 84, no. 6, 1998, pp. 599-613.
- [19] Bohr, M., “Staying Ahead of the Power Curve: Q&A with Intel Senior Fellow Mark T. Bohr,” *Technology@Intel Magazine*, August 2006.
- [20] Bohr, M., et al., “A high performance 0.25um logic technology optimized for 1.8 V operation,” *Technical Digest of the International Electron Devices Meeting*, 1996, pp. 847-850.
- [21] Borkar, S. et al., “Parameter variation and impact on Circuits and Microarchitecture,” in *Proceedings of the Design Automation Conference*, 2003, pp. 338-342.
- [22] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004, 716 pp.
<http://www.stanford.edu/~boyd/cvxbook/>
- [23] Brand, A., et al., “Intel’s 0.25 Micron, 2.0 Volts Logic Process Technology,” *Intel Technology Journal*, Q3 1998, 8 pp. <http://developer.intel.com/technology/itj/q31998/pdf/p856.pdf>
- [24] Brooks, D., et al., “Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors,” *IEEE Micro*, vol. 20, no. 6, 2000, pp. 26-44.
- [25] Brglez, F., and Fujiwara, H., “A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran,” in *Proceedings of the International Symposium Circuits and Systems*, 1985, pp. 695-698.
- [26] Brodersen, R., et al., “Methods for True Power Minimization,” in *Proceedings of the International Conference on Computer-Aided Design*, 2002, pp. 35-42.
- [27] Burd, T., et al., “A Dynamic Voltage Scaled Microprocessor System,” *IEEE Journal of Solid State Circuits*, vol. 35, no. 11, 2000, pp. 1571-80.

- [28] Burd, T., *Energy-Efficient Processor System Design*, Ph.D. dissertation, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 2001, 301 pp.
- [29] Burd, T. “Low-Power CMOS Library Design Methodology,” M.S. Report, University of California, Berkeley, UCB/ERL M94/89, 1994, 78 pp.
<http://bwrc.eecs.berkeley.edu/Publications/theses/low.power.CMOS.library.MS/masters.ps.gz>
- [30] Callaway, T., and Swartzlander, E., “Optimizing Arithmetic Elements for Signal Processing,” *VLSI Signal Processing*, 1992, pp. 91-100.
- [31] Chakraborty, S., and Murgai, R., “Complexity of Minimum-delay Gate Resizing,” in *Proceedings of the International Conference on VLSI Design*, 2001, pp. 425-430.
- [32] Chandrakasan, A., and Brodersen, R., “Minimizing Power Consumption in Digital CMOS Circuits,” in *Proceedings of the IEEE*, vol. 83, no. 4, April 1995, pp. 498-523.
- [33] Chang, A., “VLSI Datapath Choices: Cell-Based Versus Full-Custom,” S.M. Thesis, Massachusetts Institute of Technology, February 1998, 146 pp.
http://cva.stanford.edu/publications/1998/achang_sm_thesis.pdf
- [34] Chen, C., Srivastava, A., and Sarrafzadeh, M., “On Gate Level Power Optimization Using Dual-Supply Voltages,” *IEEE Transactions on VLSI Systems*, vol. 9, no. 5, 2001, pp. 616-629.
- [35] Chen, C., and Sarrafzadeh, M., “An Effective Algorithm for Gate-Level Power-Delay Tradeoff Using Two Voltages,” in *Proceedings of the International Conference on Computer Design*, 1999, pp. 222 – 227.
- [36] Chen, C., and Sarrafzadeh, M., “Power Reduction by Simultaneous Voltage Scaling and Gate Sizing,” in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2000, pp. 333-338.
- [37] Chinnery, D., et al., “Automatic Replacement of Flip-Flops by Latches in ASICs,” chapter 7 in *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002, pp. 187-208.
- [38] Chinnery, D., and Keutzer, K., *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002, 432 pp.
- [39] Chinnery, D., and Keutzer, K., “Linear Programming for Sizing, V_{th} and V_{dd} Assignment,” in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2005, pp. 149-154.
- [40] Chinnery, D., Nikolić, B., and Keutzer, K., “Achieving 550 MHz in an ASIC Methodology,” in *Proceedings of the Design Automation Conference*, 2001, pp. 420-425.
- [41] Chinnery, D., Thompson, B., Orshansky, M., and Keutzer, K., “Power Minimization with Multiple Supply Voltages and Multiple Threshold Voltages,” presented at the Semiconductor Research Corporation Technical Conference, Dallas, TX, August 25-27, 2003.
- [42] Chou, H., Wang, Y., and Chen, C., “Fast and Effective Gate-Sizing with Multiple-V_t Assignment using Generalized Lagrangian Relaxation,” in *Proceedings of the Asia South Pacific Design Automation Conference*, 2005, pp. 381-386.
- [43] Chu, K., and Pulfrey, D., “A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic,” *Journal of Solid-State Circuits*, vol. sc-22, no. 4, August 1987, pp. 528-32.
- [44] Clark, L., “The XScale Experience: Combining High Performance with Low Power from 0.18um through 90nm Technologies,” presented at the Electrical Engineering and Computer Science Department of the University of Michigan, September 30, 2005.
http://www.eecs.umich.edu/vlsi_seminar/f05/Slides/VLSI_LClark.pdf
- [45] Clark, L., et al., “An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications,” *Journal of Solid-State Circuits*, vol. 36, no. 11, November 2001, pp. 1599-1608.
- [46] Clark, L., et al., “Standby Power Management for a 0.18um Microprocessor,” in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2002, pp. 7-12.
- [47] Clark, L., Morrow, M., and Brown, W., “Reverse-Body Bias and Supply Collapse for Low Effective Standby Power,” *IEEE Transactions on VLSI Systems*, vol. 12, no. 9, 2004, pp. 947-956.
- [48] Contreras, G., et al., “XTREM: A Power Simulator for the Intel XScale Core,” in *Proceedings of the ACM Conference on Languages, Compilers, and Tools for Embedded Systems*, 2004, 11 pp.
- [49] Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms*, MIT Press, 1989, 1028 pp.
- [50] Cote, M., and Hurat, P. “Faster and Lower Power Cell-Based Designs with Transistor-Level Cell Sizing,” chapter 9 in *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002, pp. 225-240.

- [51] Coudert, O., "Gate Sizing for Constrained Delay/Power/Area Optimization," *IEEE Transactions on VLSI Systems*, vol. 5, no. 4, 1997, pp. 465-472.
- [52] Dai, W., and Staepelaere, D., "Useful-Skew Clock Synthesis Boosts ASIC Performance," chapter 8 in *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002, pp. 209-223.
- [53] Davies, B., et al., "iPART: An Automated Phase Analysis and Recognition Tool," Intel Research Tech Report IR-TR-2004-1, 2004, pp. 12.
- [54] De, V., et al., "Techniques for Leakage Power Reduction," chapter in *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2001, pp. 48.52.
- [55] De Gelas, J. AMD's Roadmap. February 28, 2000.
http://www.aceshardware.com/Spades/read.php?article_id=119
- [56] Dharchoudhury, A., Blaauw, D., Norton, J., Pullela, S., and Dunning, J., "Transistor-level sizing and timing verification of domino circuits in the Power PC microprocessor," in *Proceedings of the International Conference on Computer Design*, 1997, pp. 143-148.
- [57] Dhillon, Y., et al., "Algorithm for Achieving Minimum Energy Consumption in CMOS Circuits Using Multiple Supply and Threshold Voltages at the Module Level," in *Proceedings of the International Conference on Computer-Aided Design*, 2003, pp. 693-700.
- [58] Duarte, D., et al., "Evaluating run-time techniques for Leakage Power Reduction," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2002, pp. 31-38.
- [59] Duarte, D., et al., "Evaluating the Impact of Architectural-Level Optimizations on Clock Power," in *Proceedings of the Annual IEEE International ASIC/SOC Conference*, 2001, pp. 447-51.
- [60] Elmore, E., "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, January 1948, pp. 55-63.
- [61] Fanucci, L., and Saponara, S., "Data driven VLSI computation for low power DCT-based video coding," *International Conference on Electronics, Circuits and Systems*, vol.2, 2002, pp. 541-4.
- [62] Fanucci, L., and Saponara, S., "Low-Power VLSI Architectures for 3D Discrete Cosine Transform (DCT)," in *Proceedings of the International Midwest Symposium on Circuits and Systems*, 2003, pp. 1567-1570.
- [63] Ferguson, D., "Fibonacci Searching," *Communications of the ACM*, vol. 3, no. 12, 1960, pp. 648.
- [64] Fishburn, J., and Dunlop, A., "TILOS: A Posynomial Programming Approach to Transistor Sizing," in *Proceedings of the International Conference on Computer-Aided Design*, 1985, pp. 326-328.
- [65] Flynn, D., and Keating, M., "Creating Synthesizable ARM Processors with Near Custom Performance," chapter 17 in *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002, pp. 383-407.
- [66] Forrest, J., Nuez, D., and Lougee-Heimer, R., CLP User Guide,
<http://www.coin-or.org/Clp/userguide/>
- [67] Frenkil, J., and Venkatraman, S., "Power Gating Design Automation," chapter in *Closing the Power Gap between ASIC and Custom*, Springer, 2007.
- [68] Furber, S., *ARM System-on-Chip Architecture*. 2nd Ed. Addison-Wesley, 2000.
- [69] Ganswijk, J., Chip Directory: ARM Processor family.
<http://www.xs4all.nl/~ganswijk/chipdir/fam/arm/>
- [70] Garg, M., "High Performance Pipelining Method for Static Circuits using Heterogeneous Pipelining Elements," in *Proceedings of the European Solid-State Circuits Conference*, 2003, pp. 185-188.
- [71] Ghani, T., et al., "100 nm Gate Length High Performance / Low Power CMOS Transistor Structure," *Technical digest of the International Electron Devices Meeting*, 1999, pp. 415-418.
- [72] Goering, R., "The battle for logic synthesis," EE Times, September 9, 2004.
http://www.eetimes.com/news/design/columns/tool_talk/showArticle.jhtml?articleID=46200731
- [73] Golden, M., et al., "A Seventh-Generation x86 Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 11, November 1999, pp. 1466-1477.
- [74] Gong, J., et al., "Simultaneous buffer and wire sizing for performance and power optimization," *International Symposium on Low Power Electronics and Design*, 1996, pp. 271-6.
- [75] Gonzalez, D., "Micro-RISC architecture for the wireless market," *IEEE Micro*, vol. 19, no. 4, 1999, pp. 30-37.
- [76] Gowan, M., Biro, L., and Jackson, D., "Power Considerations in the Design of the Alpha 21264 Microprocessor," in *Proceedings of the Design Automation Conference*, 1998, pp. 726-731.

- [77] Greenhalgh, P., "Power Management Techniques for Soft IP," *Synopsys Users Group European Conference*, May 6, 2004, 12 pp.
- [78] Greenlaw, D., et al., "Taking SOI Substrates and Low-k Dielectrics into High-Volume Microprocessor Production," *Technical Digest of the International Electron Devices Meeting*, 2003, 4 pp.
- [79] Grochowski, E., and Annavaram, M., "Energy per Instruction Trends in Intel Microprocessors," *Technology@Intel Magazine*, March 2006, 8 pp.
- [80] Gronowski, P., et al., "High-Performance Microprocessor Design," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, May 1998, pp. 676-686.
- [81] Hamoui, A., and Rumin, N., "An Analytical Model for Current, Delay, and Power Analysis of CMOS Logic Circuits," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 10, 2000, pp. 999-1007.
- [82] Hansen, M., Yalcin, H., Hayes, J., ISCAS High-Level Models.
<http://www.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html>
- [83] Hansen, M., Yalcin, H., Hayes, J., "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE Design & Test of Computers*, vol. 16, no. 3, 1999, pp. 72-80.
- [84] Hare, C. 586/686 Processors Chart. <http://users.erols.com/chare/586.htm>
- [85] Hare, C. 786 Processors Chart. <http://users.erols.com/chare/786.htm>
- [86] Harris, D., "High Speed CMOS VLSI Design – Lecture 2: Logical Effort & Sizing," November 4, 1997.
- [87] Harris, D., and Horowitz, M., "Skew-Tolerant Domino Circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, November 1997, pp. 1702-1711.
- [88] Harris, D., et al. "The Fanout-of-4 Inverter Delay Metric," unpublished manuscript, 1997, 2 pp.
<http://odin.ac.hmc.edu/~harris/research/FO4.pdf>
- [89] Harstein, A., and Puzak, T., "Optimum Power/Performance Pipeline Depth," in *Proceedings of the 36th International Symposium on Microarchitecture*, 2003, pp. 117-126.
- [90] Harstein, A., and Puzak, T., "The Optimum Pipeline Depth for a Microprocessor," in *Proceedings of the International Symposium on Computer Architectures*, 2002, pp. 7-13.
- [91] Hauck, C., and Cheng, C. "VLSI Implementation of a Portable 266MHz 32-Bit RISC Core," *Microprocessor Report*, November 2001, 5 pp.
- [92] Heo, S., and Asanović, K., "Power-Optimal Pipelining in Deep Submicron Technology," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2004, pp. 218-223.
- [93] Hinton, G., et al., "A 0.18-um CMOS IA-32 Processor With a 4-GHz Integer Execution Unit," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, November 2001, pp. 1617-1627.
- [94] Hinton, G., et al., "The Microarchitecture of the Pentium 4 Processor," *Intel Technical Journal*, Q1 2001, pp. 13.
- [95] Ho, R., Mai, K.W., and Horowitz, M., "The Future of Wires," in *Proceedings of the IEEE*, vol. 89, no. 4, April 2001, pp. 490-504.
- [96] Hofstee, H., "Power Efficient Processor Architecture and the Cell Processor," in *Proceedings of the Symposium on High-Performance Computer Architecture*, 2005, pp. 258-262.
- [97] Horan, B., "Intel Architecture Update," presented at the IBM EMEA HPC Conference, May 17, 2006. www-5.ibm.com/fr/partenaires/forum/hpc/intel.pdf
- [98] Hrishikesh, M., et al., "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," in *Proceedings of the Annual International Symposium on Computer Architecture*, May 2002, pp. 14-24.
- [99] Hung, W., et al., "Total Power Optimization through Simultaneously Multiple-V_{DD} Multiple-V_{TH} Assignment and Device Sizing with Stack Forcing," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2004, pp. 144-149.
- [100] Hurat, P., "Beyond Physical Synthesis," *Synopsys Users Group Conference*, Europe, 2001, 11 pp.
- [101] ILOG, *ILOG CPLEX 10.1 User's Manual*, July 2006, 476 pp.
- [102] Intel, Dual-Core Intel Xeon Processor 5100 Series, Features, September 2006.
<http://www.intel.com/cd/channel/reseller/asmo-na/eng/products/server/processors/5100/feature/index.htm>
- [103] Intel, Intel Core Duo Processor Specifications, September 2006.
<http://www.intel.com/products/processor/coreduo/specs.htm>

- [104]Intel, Intel Pentium M Processor – Voltage Requirements, September 28, 2005.
<http://www.intel.com/support/processors/mobile/pm/sb/cs-007983.htm>
- [105]Intel, Intel Unveils World's Best Processor, July 27, 2006.
<http://www.intel.com/pressroom/archive/releases/20060727comp.htm>
- [106]Intel, Intel XScale Microarchitecture: Benchmarks.
<http://developer.intel.com/design/intelxscale/benchmarks.htm>
- [107]Intel, Inside the NetBurst Micro-Architecture of the Intel Pentium 4 Processor, Revision 1.0, 2000.
<http://developer.intel.com/pentium4/download/netburst.pdf>
- [108]Intel, Processor Number Feature Table, September 2006.
http://www.intel.com/products/processor_number/proc_info_table.pdf
- [109]Ishii, A.T., Leiserson, C.E., and Papaefthymiou, M.C. “Optimizing Two-Phase, Level-Clocked Circuitry,” *Journal of the Association for Computing Machinery*, January 1997, pp. 148-199.
- [110]Ishihara, F., Sheikh, F., and Nikolić, B., “Level Conversion for Dual-Supply Systems,” *IEEE Transactions on VLSI Systems*, vol. 12, no. 2, 2004, pp. 185-195.
- [111]Ito, M., Chinnery, D., and Keutzer, K., “Low Power Multiplication Algorithm for Switching Activity Reduction through Operand Decomposition,” in *Proceedings of the International Conference on Computer Design*, 2003, pp. 21-26.
- [112]Jan, C., et al., “A 65nm Ultra Low Power Logic Platform Technology using Uni-axial Strained Silicon Transistors,” *Technical Digest of the International Electron Devices Meeting*, 2005, pp. 60-63.
- [113]Kasamsetty, K., Ketkar, M. and Sapatnekar, S., “A New Class of Convex Functions for Delay Modeling and their Application to the Transistor Sizing Problem,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 7, 2000, pp. 779-788.
- [114]Kao, S., Zlatanovici, R., and Nikolić, B., “A 250ps 64-bit Carry-Lookahead Adder in 90nm CMOS,” *Digest of Technical Papers of the International Solid State Circuits Conference*, 2006, pp. 438-439.
- [115]Keltcher, C., et al., “The AMD Opteron Processor for Multiprocessor Servers,” *IEEE Micro*, vol. 23, no. 2, 2003, pp. 66-76.
- [116]Keshavarzi, A., et al., “Effectiveness of Reverse Body Bias for Leakage Control in Scaled Dual Vt CMOS ICs,” *International Symposium on Low-Power Electronics Design*, 2001, pp. 207-212.
- [117]Ketkar, M., Kasamsetty, K., and Sapatnekar, S., “Convex Delay Models for Transistor Sizing,” in *Proceedings of the Design Automation Conference*, 2000, pp. 655-660.
- [118]Keutzer, K., “DAGON: Technology Binding and Local Optimization by DAG Matching,” in *Proceedings of the Design Automation Conference*, 1987, pp. 341-347.
- [119]Keutzer, K. et al., “System-level Design: Orthogonalization of Concerns and Platform-Based Design,” *IEEE Transactions on Computer-Aided Design*, vol. 19, no. 12, December 2000, pp. 1523-1543.
- [120]Kim, J., “GHz ARM Processor Design,” tutorial at the International System-on-Chip Conference, October 23, 2002.
- [121]Kitahara, T., et al., “A clock-gating method for low-power LSI design,” in *Proceedings of the Asia and South Pacific Design Automation Conference*, 1998, pp.307-12.
- [122]Kosonocky, S., et al., “Low-Power Circuits and Technology for Wireless Data Systems,” *IBM Journal of Research and Development*, vol. 47, no. 2/3, March/May 2003, pp. 283-298.
- [123]Kulkarni, S., and Sylvester, D., “Fast and Energy-Efficient Asynchronous Level Converters for Multi-VDD Design,” *IEEE Transactions on VLSI Systems*, September 2004, pp. 926-936.
- [124]Kulkarni, S., Srivastava, A., and Sylvester, D., “A New Algorithm for Improved VDD Assignment in Low Power Dual VDD Systems,” *International Symposium on Low-Power Electronics Design*, 2004, pp. 200-205.
- [125]Kulkarni, S., et al., “Power Optimization Techniques Using Multiple Supply Voltages,” chapter in *Closing the Power Gap between ASIC and Custom*, Springer, 2007.
- [126]Kuon, I., and Rose, J., “Measuring the Gap Between FPGAs and ASICs,” *International Symposium on Field Programmable Gate Arrays*, 2006, pp 21-30
- [127]Kurd, N.A, et al., “A Multigigahertz Clocking Scheme for the Pentium® 4 Microprocessor,” *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, November 2001, pp. 1647-1653.
- [128]Kuroda, T., “Low-power CMOS design in the era of ubiquitous computers,” *OYO BUTURI*, vol. 73, no. 9, 2004, pp. 1184-1187.
- [129]Larri, G., “ARM810: Dancing to the Beat of a Different Drum,” presented at Hot Chips, 1996.

- [130]Lee, D., et al., "Analysis and Minimization Techniques for Total Leakage Considering Gate Oxide Leakage," *proceedings of the Design Automation Conference*, 2003, pp. 175-80.
- [131]Lee, D., et al., "Simultaneous Subthreshold and Gate-Oxide Tunneling Leakage Current Analysis in Nanometer CMOS Design," in *Proceedings of the International Symposium on Quality Electronic Design*, 2003, pp. 287-292.
- [132]Lee, D., and Blaauw, D., "Static Leakage Reduction through Simultaneous Threshold Voltage and State Assignment," in *Proceedings of the Design Automation Conference*, 2003, pp. 191-194.
- [133]Lee, D., Blaauw, D., and Sylvester, D., "Gate Oxide Leakage Current Analysis and Reduction for VLSI Circuits," *IEEE Transactions on VLSI Systems*, vol. 12, no. 2, 2004, pp. 155-166.
- [134]Lee, J., et al., "On the signal bounding problem in timing analysis," in *Proceedings of the International Conference on Computer-Aided Design*, 2001, pp. 507-514.
- [135]Leitjen, J., Meerbergen, J., and Jess, J., "Analysis and Reduction of Glitches in Synchronous Networks," in *Proceedings of the European Design and Test Conference*, 1995, pp. 398-403.
- [136]Levy, M., "Samsung Twists ARM Past 1GHz," *Microprocessor Report*, October 16, 2002.
- [137]Lexra, Lexra LX4380 Product Brief, 2002, http://www.lexra.com/LX4380_PB.pdf
- [138]Library Technologies, ASIC Libraries Made Easy: SolutionWare User Manual, 2002.
- [139]Mahnke, T., "Low Power ASIC Design Using Voltage Scaling at the Logic Level," Ph.D. dissertation, Department of Electronics and Information Technology, Technical University of Munich, May 2003, pp. 204.
- [140]Mani, M., Devgan, A., and Orshansky, M., "An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints," in *Proceedings of the Design Automation Conference*, 2005, pp. 309-314.
- [141]Mathworks, *MATLAB: The Language of Technical Computing – MATLAB Function Reference*, version 6, The MathWorks, 2002.
- [142]Matson, M., and Glasser, L., "Macromodeling and Optimization of Digital MOS VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 5, no. 4, 1986, pp. 659-678.
- [143]McDonald, C., "The Evolution of Intel's Copy Exactly! Technology Transfer Method," *Intel Technology Journal*, Q4 1998. <http://developer.intel.com/technology/itj/q41998/pdf/copyexactly.pdf>
- [144]Montanaro, J., et al., "A 160MHz, 32-b, 0.5W, CMOS RISC Microprocessor," *Journal of Solid-State Circuits*, vol. 31, no. 11, 1996, pp. 1703-14.
- [145]Moore, G., "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, 1965, pp. 114-117.
- [146]MOSEK ApS, *The MOSEK optimization tools version 3.2 (Revision 8): User's manual and reference*, MOSEK ApS, 2002.
- [147]Moyer, B., "Low-Power Design for Embedded Processors," in *Proceedings of the IEEE*, vol. 89, no. 11, November 2001.
- [148]MTEK Computer Consulting, AMD CPU Roster, January 2002.
http://www.cpuscorecard.com/cpuprices/head_amd.htm
- [149]MTEK Computer Consulting, Intel CPU Roster, January 2002.
http://www.cpuscorecard.com/cpuprices/head_intel.htm
- [150]Mutoh, S., et al., "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *Journal of Solid-State Circuits*, vol. 30, no. 8, 1995, pp. 847-854.
- [151]Nair, R., et al., "Generation of Performance Constraints for Layout," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 8, 1989, pp. 860-874.
- [152]Narendra, S., et al., "Comparative Performance, Leakage Power and Switching Power of Circuits in 150 nm PD-SOI and Bulk Technologies Including Impact of SOI History Effect," *Symposium on VLSI Circuits*, 2001, pp. 217-8.
- [153]Nassif, S., "Delay Variability: Sources, Impact and Trends," *International Solid-State Circuits Conference*, 2000.
- [154]Nguyen, D., et al., "Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization," *International Symposium on Low Power Electronics and Design*, 2003, pp. 158-163.
- [155]Nikolić, B., et al., "Layout Decompression Chip for Maskless Lithography," in *Emerging Lithographic Technologies VIII, Proceedings of SPIE*, vol. 5374, 2004, 8 pp.

- [156] Nose, K., and Sakurai, T., "Optimization of V_{DD} and V_{TH} for Low-Power and High-Speed Applications," in *Proceedings of the Asia South Pacific Design Automation Conference*, 2000, pp. 469-474.
- [157] Nowka, K., and Galambos, T., "Circuit Design Techniques for a Gigahertz Integer Microprocessor," in *Proceedings of the International Conference on Computer Design*, 1998, pp. 11-16.
- [158] Oliveira, A., and Murgai, R., "On the Problem of Gate Assignment Under Different Rise and Fall Delays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, 2003, pp. 807-814.
- [159] Orshansky, M., Thompson, B., and Keutzer, K., "A Fast Algorithm for Total Power Reduction by Simultaneous Assignment of Dual Threshold Voltages and Sizing," unpublished paper, December 2002.
- [160] Ostrander, D., "Logic Technology and Manufacturing," slides presented at AMD's Technology Analyst Day, 2006. http://www.amd.com/us-en/assets/content_type/DownloadableAssets/DarylOstranderAMDAналystDay.pdf
- [161] Pant, P., De, V., and Chatterjee, A., "Simultaneous Power Supply, Threshold Voltage, and Transistor Size Optimization for Low-Power Operation of CMOS Circuits," *IEEE Transactions on VLSI Systems*, vol. 6, no. 4, 1998, pp. 538-545.
- [162] Pant, P., Roy, R., and Chatterjee, A., "Dual-Threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits," *IEEE Transactions on VLSI Systems*, vol. 9, no. 2, 2001, pp. 390-394.
- [163] Perera, A.H., et al., "A versatile 0.13um CMOS Platform Technology supporting High Performance and Low Power Applications," *Technical Digest of the International Electron Devices Meeting*, 2000, pp. 571-574.
- [164] Pradhan, D., et al., "Gate-Level Synthesis for Low-Power Using New Transformations," *International Symposium on Low Power Electronics and Design*, 1996, pp. 297-300.
- [165] Puri, R., et al., "Pushing ASIC Performance in a Power Envelope," in *Proceedings of the Design Automation Conference*, 2003, pp. 788-793.
- [166] Quinn, J., *Processor98: A Study of the MPU, CPU and DSP Markets*, Micrologic Research, 1998.
- [167] Rabaey, J.M., *Digital Integrated Circuits*. Prentice-Hall, 1996.
- [168] Raghunathan, A., Dey, S., and Jha, N., "Glitch Analysis and Reduction in Register Transfer Level Power Optimization," in *Proceedings of the Design Automation Conference*, 1996, pp. 331-336.
- [169] Ricci, F., "A 1.5 GHz 90 nm Embedded Microprocessor Core," *Digest of Technical Papers of the Symposium on VLSI Circuits*, 2005, pp. 12-15.
- [170] Richardson, N., et al., "The iCORE™ 520MHz Synthesizable CPU Core," Chapter 16 of *Closing the Gap Between ASIC and Custom*, 2002, pp. 361-381.
- [171] Rollins, N., and Wirthlin, M., "Reducing Energy in FPGA Multipliers Through Glitch Reduction," presented at the International Conference on Military and Aerospace Programmable Logic Devices, September 2005, 10 pp.
- [172] Roy, K., Mukhopadhyay, S., and Mahmoodi-Meimand, H., "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," in *Proceedings of the IEEE*, vol. 91, no. 2, 2003, pp. 305-327.
- [173] Sakurai, T., and Newton, R., "Delay Analysis of Series-Connected MOSFET Circuits," *Journal of Solid-State Circuits*, vol. 26, no. 2, February 1991, pp. 122-131.
- [174] Satish, N., et al., "Evaluating the Effectiveness of Statistical Gate Sizing for Power Optimization," Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, ERL Memorandum M05/28, August 2005.
- [175] Schultz, J., "The Future of Computing: Tera-scale Computing Research Overview," slides presented at the Intel Developer Forum, San Francisco, September 2006.
- [176] Segars, S., "The ARM9 Family – High Performance Microprocessors for Embedded Applications," in *Proceedings of the International Conference on Computer Design*, 1998, pp. 230-235.
- [177] Semiconductor Industry Association, International Technology Roadmap for Semiconductors 2000 Update: Overall Roadmap Technology Characteristics. December 2000.
<http://public.itrs.net/Files/2000UpdateFinal/ORTC2000final.pdf>
- [178] Shen, W., Lin, J., and Wang, F., "Transistor Reordering Rules for Power Reduction in CMOS Gates," in *Proceedings of the Asia South Pacific Design Automation Conference*, 1995, pp. 1-6.
- [179] Shenoy, N., "Retiming Theory and Practice," *Integration, The VLSI Journal*, vol.22, no. 1-2, August 1997, pp. 1-21.

- [180] Silberman, J., et al., "A 1.0-GHz Single-Issue 64-Bit PowerPC Integer Processor," *IEEE Journal of Solid-State Circuits*, vol.33, no.11, November 1998, pp. 1600-1608.
- [181] Simonen, P., et al., "Comparison of bulk and SOI CMOS Technologies in a DSP Processor Circuit Implementation," *International Conference on Microelectronics*, 2001.
- [182] Singh, D., et al., "Power Conscious CAD Tools and Methodologies: a Perspective," in *Proceedings of the IEEE*, vol. 83, no. 4, April 1995, pp. 570-94.
- [183] Sirichotiyakul, S., et al., "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing," in *Proceedings of the Design Automation Conference*, 1999, pp. 436-41.
- [184] Smith, D., *HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs Using VHDL or Verilog*, Doone Publications, 1998, 464 pp.
- [185] Smith, M., *Application-specific Integrated Circuits*, Addison-Wesley, Berkeley, CA, 1997.
- [186] Srinivasan, V., et al., "Optimizing pipelines for power and performance," in *Proceedings of the International Symposium on Microarchitecture*, 2002, pp. 333-344.
- [187] Srivastava, A., "Simultaneous V_t Selection and Assignment for Leakage Optimization," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, 2003, pp. 146-151.
- [188] Srivastava, A., Sylvester, D., and Blaauw, D., "Power Minimization using Simultaneous Gate Sizing Dual-Vdd and Dual-V_t Assignment," in *Proceedings of the Design Automation Conference*, 2004, pp. 783-787.
- [189] Standard Performance Evaluation Corporation, SPEC's Benchmarks and Published Results, 2006. <http://www.spec.org/benchmarks.html>
- [190] Staszewski, R., Muhammad, K., and Balsara, P., "A 550-MSample/s 8-Tap FIR Digital Filter for Magnetic Recording Read Channels," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 8, 2000, pp. 1205-1210.
- [191] STMicroelectronics, *CORE9GPHS HCMOS9 Databook*, version 3.2a, December 14, 2001.
- [192] STMicroelectronics, "STMicroelectronics 0.25μ, 0.18μ & 0.12 CMOS," slides presented at the annual Circuits Multi-Projets users meeting, January 9, 2002. http://cmp.imag.fr/Forms/Slides2002/061_STM_Process.pdf
- [193] Stojanovic, V., et al., "Energy-Delay Tradeoffs in Combinational Logic Using Gate Sizing and Supply Voltage Optimization," in *Proceedings of the European Solid-State Circuits Conference*, 2002, pp. 211-214.
- [194] Stojanovic, V., and Oklobdzija, V., "Comparative Analysis of Master-Slave Latches and Flip-Flops for High-Performance and Low-Power Systems," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 4, April 1999, pp. 536-548.
- [195] Stok, L., et al., "Design Flows," chapter in the *CRC Handbook of EDA for IC Design*, CRC Press, 2006.
- [196] Stok, L., et al., "Pushing ASIC Performance in a Power Envelope," chapter in *Closing the Power Gap between ASIC and Custom*, Springer, 2007.
- [197] Sundararajan, V., and Parhi, K., "Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 139-144.
- [198] Sundararajan, V., and Parhi, K., "Synthesis of Low Power CMOS VLSI Circuits Using Dual Supply Voltages," in *Proceedings of the Design Automation Conference*, 1999, pp. 72-75.
- [199] Sutherland, I., Sproull, R., and Harris, D., *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann, 1999.
- [200] Sylvester, D. and Keutzer, K., "Getting to the Bottom of Deep Sub-micron," in *Proceedings of the International Conference on Computer Aided Design*, November 1998, pp. 203-211.
- [201] Synopsys, *Design Compiler User Guide*, version U-2003.06, June 2003, 427 pp.
- [202] Synopsys. *Synopsys Design Compiler – Whitepaper*. February 1998. http://www.synopsys.com/products/logic/dc98_bckgr.html
- [203] Synopsys, *Power Compiler User Guide*, version 2003.06, 2003.
- [204] Synopsys, *VCS/VCSI User Guide*, version 6.0, January 2001, 508 pp.
- [205] Takahashi, M., et al., "A 60-mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," *Journal of Solid-State Circuits*, vol. 33, no. 11, 1998, pp. 1772-1780.

- [206]techPowerUp! CPU Database, August 2006.
<http://www.techpowerup.com/cpudb/>
- [207]Tennakoon, H., and Sechen, C., "Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-Processing Step," in *Proceedings of the International Conference on Computer-Aided Design*, 2002, pp. 395-402.
- [208]Tensilica, Xtensa 6 Product Brief, 2005.
- [209]Tensilica, Xtensa Microprocessor – Overview Handbook – A Summary of the Xtensa Data Sheet for Xtensa T1020 Processor Cores. August 2000.
- [210]Tensilica, Xtensa Microprocessor – Overview Handbook – A Summary of the Xtensa Data Sheet for Xtensa V(T1050) Processor Cores. August 2002.
- [211]Thompson, S., et al., "A 90nm Logic Technology Featuring 50nm Strained Silicon Channel Transistors, 7 Layers of Cu Interconnects, Low k ILD, and 1 μm^2 SRAM Cell," *Technical Digest of the International Electron Devices Meeting*, 2002, pp. 61-64.
- [212]Thompson, S., et al., "An Enhanced 130 nm Generation Logic Technology Featuring 60 nm Transistors Optimized for High Performance and Low Power at 0.7 – 1.4 V," *Technical Digest of the International Electron Devices Meeting*, 2001, 4 pp.
- [213]Tschanz, J. et al., "Adaptive Body Bias for Reducing the Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage," *IEEE International Solid-State Circuits Conference*, 2002.
- [214]Tschanz, J., et al., "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, 2003, pp. 1838-1845.
- [215]TSMC, 0.13 Micron CMOS Process Technology, March 2002
- [216]TSMC, 0.18 Micron CMOS Process Technology, March 2002.
- [217]TSMC, TSMC Unveils Nexsys 90-Nanometer Process Technology, August 2006.
<http://www.tsmc.com/english/technology/t0113.htm>
- [218]Tsui, C., et al., "Low Power State Assignment Targeting Two- And Multi-level Logic Implementations," in *Proceedings of the International Conference on Computer-Aided Design*, 1994, pp. 82-87.
- [219]Tyagi, S., et al., "A 130 nm generation logic technology featuring 70 nm transistors, dual V_t transistors and 6 layers of Cu interconnects," *Technical Digest of the International Electron Devices Meeting*, 2000, pp. 567-570.
- [220]Tyagi, S., et al., "An advanced low power, high performance, strained channel 65nm technology," *Technical Digest of the International Electron Devices Meeting*, 2005, pp. 245-247.
- [221]Usami, K., et al., "Automated Selective Multi-Threshold Design For Ultra-Low Standby Applications," in *Proceedings of the International Symposium on Low Power Design*, 2002, pp. 202-206.
- [222]Usami, K., and Horowitz, M., "Clustered voltage scaling technique for low power design," in *Proceedings of the International Symposium on Low Power Design*, 1995, pp. 3–8.
- [223]Usami, K., and Igarashi, M., "Low-Power Design Methodology and Applications Utilizing Dual Supply Voltages," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2000, pp. 123-8.
- [224]Usami, K., et al., "Automated Low-Power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, 1998, pp. 463-472.
- [225]Usami, K., et al., "Automated Low-power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor," in *Proceedings of the Custom Integrated Circuits Conference*, 1997, pp.131-134.
- [226]Vangal, S., et al., "5GHz 32b Integer-Execution Core in 130nm Dual- V_t CMOS," *Digest of Technical Papers of the IEEE International Solid-State Circuits Conference*, 2002, pp. 334-335, 535.
- [227]Veendrick, H., "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *Journal of Solid-State Circuits*, vol. SC-19, August 1984, pp. 468-73.
- [228]Virtual Silicon. <http://www.virtual-silicon.com/>
- [229]Wang, Q., and Vrudhula, S., "Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual- V_t CMOS Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, 2002, pp. 306-318.
- [230]Wang, Q., and Vrudhula, S., "Static Power Optimization of Deep Submicron CMOS Circuits for Dual V_t Technology," in *Proceedings of the International Conference on Computer-Aided Design*, 1998, pp. 490-496.

- [231]Wei, L., et al., "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits," in *Proceedings of the Design Automation Conference*, 1998, pp. 489-494.
- [232]Wei, L., et al., "Mixed-V_{th} (MVT) CMOS Circuit Design Methodology for Low Power Applications," in *Proceedings of the Design Automation Conference*, 1999, pp. 430-435.
- [233]Wei, L., Roy, K., and Koh, C., "Power Minimization by Simultaneous Dual-V_{th} Assignment and Gate-Sizing," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2000, pp. 413-416.
- [234]Weicker, R., "Dhrystone: A Synthetic Systems Programming Benchmark," *Communications of the ACM*, vol. 27, no. 10, 1984, pp. 1013-1030.
- [235]Weisstein, E., "Wheel Graph," from MathWorld, 2005.
<http://mathworld.wolfram.com/WheelGraph.html>
- [236]Weste, N., and Eshraghian, K., *Principles of CMOS VLSI Design*, Addison-Wesley, 1992.
- [237]Wilton, S., Ang, S., and Luk, W., "The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, 2004, pp. 719-728.
- [238]Wolfe, A., "Intel Clears Up Post-Tejas Confusion," VARBusiness magazine, May 17, 2004.
<http://www.varbusiness.com/sections/news/breakingnews.jhtml?articleId=18842588>
- [239]Xanthopoulos, T., and Chandrakasan, A., "A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization," *Journal of Solid State Circuits*, vol. 35, no. 5, May 2000, pp. 740-50.
- [240]Xanthopoulos, T., and Chandrakasan, A., "A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity," *Journal of Solid State Circuits*, vol. 34, May 1999, pp. 693-703.
- [241]Yang, F., et al., "A 65nm Node Strained SOI Technology with Slim Spacer," *Technical Digest of the International Electron Devices Meeting*, 2003, pp. 627-630.
- [242]Yeh, C., et al., "Gate-Level Design Exploiting Dual Supply Voltages for Power-Driven Applications," in *Proceedings of the Design Automation Conference*, 1999, pp. 68-71.
- [243]Yeo, E., et al., "A 500-Mb/s Soft-Output Viterbi Decoder," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 7, 2003, pp. 1234-1241.
- [244]Zhuang, X., Zhang, T., and Pande, S., "Hardware-managed Register Allocation for Embedded Processors," in *Proceedings of the ACM Conference on Languages, Compilers, and Tools for Embedded Systems*, 2004, pp. 10.
- [245]Zimmerman, R., and Fichtner, W., "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," *Journal of Solid-State Circuits*, vol. 32, no. 7, July 1997, 1079-1090.
- [246]Zlatanovici, R., and Nikolić, B., "Power-Performance Optimal 64-bit Carry-Lookahead Adders," *European Solid-State Circuits Conference*, 2003, pp. 321-324.
- [247]Zlatanovici, R., Nikolić, B. "Power - Performance Optimization for Custom Digital Circuits," *International Workshop on Power And Timing Modeling Optimization and Simulation*, September 2005, pp. 404-414.