

Copyright © 1994, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

LOW POWER DIGITAL CMOS DESIGN

by

Anantha P. Chandrakasan

Memorandum No. UCB/ERL M94/65

30 August 1994

LOW POWER DIGITAL CMOS DESIGN

by

Anantha P. Chandrakasan

Memorandum No. UCB/ERL M94/65

30 August 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

Low Power Digital CMOS Design

by

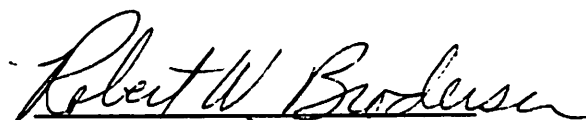
Anantha P. Chandrakasan

Doctor of Philosophy in Engineering-Electrical
Engineering and Computer Sciences

University of California at Berkeley

Professor Robert W. Brodersen, Chair

Portable operation is becoming increasingly important for many of the most exciting new electronic products. The strict limitation on power dissipation which this imposes must be met by the designer while still meeting ever higher computational requirements. A system level approach is presented to minimize power consumption which involves optimizing technology, circuit and logic design, architectures, and algorithms. An architecture driven voltage scaling strategy has been developed which trades silicon area for lower power consumption without sacrificing system performance. This strategy enables an order of magnitude reduction in power consumption over current-day "low-voltage" 3.3V standards. Approaches to minimize the switched capacitance are also developed which involve data coding and representation, minimizing resource sharing, minimizing spurious transitions, optimizing transistor sizes, activity based logic level shut down, aggressive predictive shut down for programmable computation, dynamic range optimization, and the utilization of algorithmic transformations. A CAD tool has been developed to automatically find computational structures that have minimal power requirements to implement a given function. The various techniques have been applied to the design of a chipset for a portable multimedia terminal. Six chips provide protocol conversion, synchronization, error correction, packetization, buffering, video decompression, and D/A conversion at a total power consumption of less than 5 mW, while operating from a 1.1V supply.



Robert W. Brodersen

Chairman of Committee

**To my dear parents,
Gowri and Nagappa Chandrakasan
my wife Karthiyayani,
and Mrs. Charlotte Coe**

TABLE OF CONTENTS

Introduction.....	1
1.1 Overview of Thesis	4
1.1.1 Low-power Design Methodologies [Chapters 2,3, and 4].....	4
1.1.2 CAD Tools for Low-power Design [Chapter 5].....	9
1.1.3 System Design Example: A Portable Multimedia Terminal [Chapters 6 and 7].....	9
1.1.4 Energy Efficient Programmable Computation [Chapter 8]	10
Power Consumption in CMOS Circuits	12
2.1 Switching Component of Power	13
2.1.1 Computation of Switching Energy Per Transition.....	14
2.1.2 Definition of Switching Activity Factor, α	19
2.1.3 Components of Physical Load Capacitance	22
2.1.4 Influence of Logic Level Statistics and Circuit Topologies on Switching Activity, α	29
2.1.5 Word Level Signal Statistics Influencing Activity, α	44
2.1.6 Influence of Voltage Scaling	47
2.2 Short-circuit Component of Power	52
2.3 Leakage Component of Power	57
2.3.1 Diode Leakage	58
2.3.2 Sub-threshold Leakage.....	59
2.4 Static Power.....	61
2.4.1 Reduced Voltage Levels Feeding CMOS Gates.....	61
2.4.2 Pseudo-NMOS Logic Style.....	62
2.5 Summary	62
Approaches to Voltage Scaling and Issues for Low Voltage Operation.....	65
3.1 Approaches to Supply Voltage Scaling	66
3.1.1 Reliability-Driven Voltage Scaling.....	66
3.1.2 Technology-Driven Voltage Scaling	68
3.1.3 Energy x Delay Minimum Based Voltage Scaling.....	72
3.1.4 Voltage Scaling Through Optimal Transistor Sizing.....	73
3.1.5 Architecture-Driven Voltage Scaling	77
3.1.6 Voltage Scaling using Threshold Reduction.....	105
3.2 Support Circuitry for Low-voltage Operation.....	107

3.2.1 High Efficiency Power Supply Generation [Stratakos94]	107
3.2.2 Voltage Level-conversion Circuitry	114
3.3 Noise Considerations at Reduced Supply Voltages	115
3.4 Summary	117
Approaches to Minimizing Switched Capacitance	119
4.1 Algorithmic Optimization	122
4.1.1 Minimizing the Number of Operations	122
4.1.2 Minimizing Shift-Add Operations for Multiplications with Multiplications by Constants	126
4.1.3 Minimizing Temporal Bit Transition Activity by Choice of Data Representation	130
4.2 Architecture Optimization	144
4.2.1 Optimizing Data Representation for Arithmetic Computation	144
4.2.2 Ordering of Input Signals	149
4.2.3 Reducing Glitching Activity	151
4.2.4 Degree of Resource Sharing	152
4.3 Logic Optimization	156
4.3.1 Logic Minimization and Technology Mapping	156
4.3.2 Activity Trade-off for Various Logic Structures	157
4.3.3 Logic Level Power Down	158
4.4 Circuit Optimization	160
4.4.1 Dynamic Logic vs. Static Logic	160
4.4.2 Pass Transistor Logic vs. Conventional CMOS Logic	162
4.4.3 Synchronous vs. Asynchronous	164
4.5 Physical Design	165
4.5.1 Silicon-On-Insulator Technology	165
4.5.2 Layout Optimization	166
4.5.3 Place and Route	169
4.6 Summary	169
Computer Aided Design Tools	171
5.1 Previous and Concurrent Work	172
5.1.1 Power Estimation	172
5.1.2 High-level Transformations	178
5.2 Application of Transformations to Minimize Power	179
5.2.1 Speed-up transformations	181
5.2.2 Operation Reduction	181

5.2.3	Operation Substitution	183
5.2.4	Resource Utilization	184
5.2.5	Wordlength Reduction	187
5.3	Cost Function	189
5.3.1	Capacitance estimate	189
5.3.2	Supply Voltage Estimation.....	202
5.4	Optimization Algorithm	205
5.5	Examples and Results.....	208
5.5.1	Example #1: Wavelet Filter	209
5.5.2	Example #2: IIR Filter	211
5.5.3	Example #3: Volterra Filter.....	213
5.6	Summary	215
A Low-power Chipset for a Portable Multimedia Terminal		216
6.1	Technology Trends for Low Power.....	220
6.1.1	Battery Technology	220
6.1.2	Display Technology.....	221
6.1.3	Packaging Technology	222
6.2	System Partitioning for Low-power	223
6.2.1	Impact of the Low-power Partitioning Approach on the Interactive Latency	227
6.2.2	Impact of the Low-power Partitioning Approach on the Tolerance to High BER	228
6.3	Architecture of the Portable Multimedia Terminal	229
6.4	Protocol Chip.....	231
6.4.1	Radio Receiver Module [Burd93]	232
6.4.2	Text/Graphics Module	237
6.4.3	Speech I/O Module	251
6.4.4	Pen Input Module [Narayanaswamy93]	254
6.4.5	Radio Transmitter Module.....	254
6.4.6	Low-Power Circuits Common to all Modules	255
6.4.7	Chip Layout and Statistics.....	259
6.5	Text/Graphics Frame-buffer Module [Burstein93]	261
6.5.1	Memory Architecture	263
6.5.2	Circuit Level Optimization	265
6.5.3	Chip Statistics	267
6.6	System Implementation.....	268

6.7 Summary	271
A Low-power Video Decompression Chipset.....	273
7.1 Algorithm Selection for Video Decompression	273
7.1.1 Computational Complexity Trade-off	274
7.1.2 Robustness to Channel Errors.....	276
7.2 Video Decompression Module Implementation.....	277
7.2.1 Luminance Video Decompression.....	278
7.2.2 Chrominance Decompression Chip	283
7.2.3 Video Controller	285
7.2.4 Color Space Converter and Digital to Analog Converter.....	289
7.2.5 Chipset Summary.....	296
7.2.6 Video Decompression Test Board	297
7.3 Summary	299
Low-power	
Programmable Computation.....	300
8.1 Architectural Approaches to Low-power	301
8.2 Shutdown Techniques.....	303
8.2.1 Conventional Shutdown Approaches.....	305
8.2.2 Predictive Shutdown Approaches	307
8.3 Architecture-Driven Voltage Reduction	319
8.3.1 Voltage-Concurrency Trade-Off and Architectural Bottlenecks.....	321
8.3.2 Exploiting Concurrency in Programmable Computation	323
8.3.3 A Power Consumption Model for MIMD.....	326
8.3.4 Example: CORDIC on a MIMD using Thread Level Parallelism	328
8.3.5 Low Power Computers - Multiple Slow CPUs More Energy Efficient than a Single Fast CPU?.....	330
8.4 Summary	331
Conclusions and Future Directions	332
9.1 Future Directions	337
9.1.1 Design Methodologies	337
9.1.2 CAD Tools for Portable System Design	338
9.1.3 New Applications	339
Bibliography	341
Appendix A: Proof of NP-Completeness.....	346

LIST OF FIGURES

Figure 1-1 : Power consumption of micro-processors reported at ISSCC over the last 20 years [Rabaey94]	2
Figure 1-2 : A system level approach to supply voltage scaling	6
Figure 1-3 : Dependence of activity on statistics: correlated vs. random input	7
Figure 1-4 : A system level approach to minimizing the switched capacitance.....	8
Figure 2-1 : Circuit model for computing the dynamic power of a CMOS gate.....	14
Figure 2-2 : Switching power for reduced swing logic.	16
Figure 2-3 : Charge sharing in dynamic circuits result in “extra” switching power.....	17
Figure 2-4 : Stepwise driver for a capacitive load CL.....	18
Figure 2-5 : Switching energy analysis must include internal node switching.	21
Figure 2-6 : Circuit model for computing parasitic load capacitance of a CMOS gate.	22
Figure 2-7 : Overlap capacitance for a MOS device.	23
Figure 2-8 : Junction capacitance for a MOS device.....	24
Figure 2-9 : Parallel plate capacitance model of interconnect capacitance.	25
Figure 2-10 : Fringe field contribution to the total interconnect capacitance [Glasser85].	26
Figure 2-11 : Interconnect capacitance including fringing effect as a function of W/tox [Schaper83]	26
Figure 2-12 : Interconnect capacitance including wire-to-wire capacitance [Schaper83]	27
Figure 2-13 : State transition diagram for a 2 input NOR gate	31
Figure 2-14 : Transition activity for different gates as function of the number of inputs	32
Figure 2-15 : Static NOR vs. N-tree based dynamic NOR.....	33
Figure 2-16 : A generalized DCVSL gate	34
Figure 2-17 : Transition probability for a 2 input NOR as a function of input statistics.....	36
Figure 2-18 : Transition probability for a 2 input XOR gate.....	37
Figure 2-19 : Examples illustrating the effect of signal correlations.....	38
Figure 2-20 : Dynamic component of switching power	41
Figure 2-21 : Simple example to demonstrate the influence of circuit topology on activity	42
Figure 2-22 : Circuit imbalances result in spurious transitions	43
Figure 2-23 : Waveforms for a 16-bit adder demonstrating glitching behavior.....	44
Figure 2-24 : Data in signal processing applications is often correlated.....	45
Figure 2-25 : Power-delay product exhibiting square law dependence for two different circuits.....	48
Figure 2-26 : Data demonstrating delay characteristics follow simple first order theory	49
Figure 2-27 : Data showing improvement in power-delay product at the cost of speed	

for various circuit approaches.....	51
Figure 2-28 : Current behavior with no output load [Veendrick84].....	53
Figure 2-29 : Short-circuit current as a function of load capacitance [Veendrick84].....	56
Figure 2-30 : Short-circuit component of power.....	57
Figure 2-31 : Reverse bias diode leakage currents.....	58
Figure 2-32 : Sub-threshold leakage component of power.....	60
Figure 2-33 : Degenerated voltage level feeding a CMOS gate resulting in static power.....	61
Figure 2-34 : Implementing complex logic using pseudo-NMOS.....	62
Figure 3-1 : Drain doping profile for a conventional profile and an LDD device.....	67
Figure 3-2 : Reliability driven supply voltage selection [Davari88].....	68
Figure 3-3 : Model for computing delay and critical voltage.....	69
Figure 3-4 : Two components of delay and critical voltage [Kakamu90].....	72
Figure 3-5 : Energy x Delay product vs. V_{dd}	73
Figure 3-6 : Circuit model for analyzing the effect of transistor sizing.....	76
Figure 3-7 : Plot of energy vs. transistor sizing factor for various parasitic contributions.....	76
Figure 3-8 : A uni-processor implementation of a generic function.....	78
Figure 3-9 : A multi-processor implementation of a generic function.....	79
Figure 3-10 : Voltage reduction vs. number of processors as a function of V_t	81
Figure 3-11 : Power reduction (P1/PN) vs. number of processors as a function of V_t	84
Figure 3-12 : Capacitance function for the processing and input routing overhead.....	87
Figure 3-13 : Power vs. N for various levels of overhead capacitance.....	88
Figure 3-14 : Architecture Driven Optimum Supply Voltage.....	89
Figure 3-15 : Power vs. N for various threshold voltages.....	90
Figure 3-16 : N stage pipelining allows reduction of critical path and voltage.....	91
Figure 3-17 : Power reduction for N-stage pipelining for various overhead factors.....	92
Figure 3-18 : A simple datapath with corresponding layout.....	93
Figure 3-19 : Parallel implementation of the simple datapath.....	95
Figure 3-20 : Pipelined implementation of the simple datapath.....	96
Figure 3-21 : Optimum voltage of operation for the add/compare computation.....	98
Figure 3-22 : Using speedup transformations to reduce power.....	102
Figure 3-23 : Speed optimizing is different than power optimization.....	103
Figure 3-24 : Effect of threshold reduction on the delay for various supply voltages.....	106
Figure 3-25 : Compromise between dynamic and leakage power dissipation through V_t variation.....	107
Figure 3-26 : A Buck regulator that be used to generate arbitrary voltages.....	108
Figure 3-27 : Timing for soft-switching of the Buck converter.....	110
Figure 3-28 : Effects of varying load conditions on waveforms.....	111
Figure 3-29 : Adaptive dead-time control.....	112

Figure 3-30 : Optimum sizing of power transistors.....	113
Figure 3-31 : Die photo of a 1.5V Buck regulator.....	113
Figure 3-32 : Level-conversion I/O pad buffer.....	115
Figure 3-33 : Inductive noise on the power supplies.....	116
Figure 4-1 : Data in signal processing applications is often correlated.....	120
Figure 4-2 : Dependence of activity on statistics: correlated vs. random input	121
Figure 4-3 : Video compression/decompression using Vector Quantization.....	123
Figure 4-4 : Codebook organization for Full-search Vector Quantization	124
Figure 4-5 : Tree structured Vector Quantization.....	125
Figure 4-6 : Hardwired shift-add and time-muxed shift-add implementations	127
Figure 4-7 : Example of common sub-expression elimination.....	128
Figure 4-8 : Color conversion matrix for video applications	129
Figure 4-9 : Exploiting common sub-expression to minimize number of operations	130
Figure 4-10 : Data coding approach to reduction of switching activity on highly capacitive inter-chip I/O Lines. Note that $m \geq n$	131
Figure 4-11 : Temporal transition activity comparison for instruction addresses [Su94]	135
Figure 4-12 : Temporal transition activity comparison for data addresses [Su94].....	136
Figure 4-13 : Theoretically predicted reduction in switching activity for a stream of uniformly distributed data words [Srivastava94]	138
Figure 4-14 : Transition activity for different number representations	142
Figure 4-15 : Transition activity for two's complement representation	143
Figure 4-16 : Activity reduction for sign-magnitude over two's complement	143
Figure 4-17 : Two's complement implementation of an accumulator.....	144
Figure 4-18 : Signal statistics for two's complement implementation of the accumulator datapath assuming random inputs.....	146
Figure 4-19 : Signal Value for two's complement implementation for random inputs	146
Figure 4-20 : Sign magnitude implementation of an accumulator	147
Figure 4-21 : Signal statistics for Sign Magnitude implementation of the accumulator datapath assuming random inputs.....	148
Figure 4-22 : Reducing activity by re-ordering inputs	150
Figure 4-23 : Reducing the glitching activity	151
Figure 4-24 : Activity trade-off for time-multiplexed hardware: bus-sharing example.....	154
Figure 4-25 : Activity trade-off for time-multiplexed hardware: adder example.....	156
Figure 4-26 : Histogram of transition activity for various adder topologies [Callaway92]	158
Figure 4-27 : Data dependent logic level shutdown	159
Figure 4-28 : Schematic of a modified TSPC latch that is used to generate gated clocks	160
Figure 4-29 : CMOS vs. Pass Gate Logic: adder example.....	164
Figure 4-30 : Silicon-on-Sapphire CMOS.....	166

Figure 4-31 : Three layout approaches for a large W/L transistor	167
Figure 5-1 : Circuit for measuring the power-delay product.....	173
Figure 5-2 : Measuring capacitance switched for a 1-bit full-adder example	174
Figure 5-3 : Switched capacitance vs. number of simulated periods	176
Figure 5-4 : Statistical parameter propagation [Landman93].....	178
Figure 5-5 : Hardware model used in HYPER	180
Figure 5-6 : Reducing capacitance while maintaining throughput.....	182
Figure 5-7 : Reducing capacitance at the expense of a higher supply voltage.....	182
Figure 5-8 : Trading multiplication for an addition.....	183
Figure 5-9 : Retiming for improving resource utilization: 2nd order IIR filter.....	186
Figure 5-10 : Direct form structure vs. Parallel form structure: 8th order Avenhous filter.....	188
Figure 5-11 : Average capacitance per addition for two different modules	191
Figure 5-12 : Statistical estimation of interconnect capacitance	195
Figure 5-13 : Total capacitance switched vs. number of states for the global controller	197
Figure 5-14 : Capacitance switched vs. predicted capacitance for local controllers.....	200
Figure 5-15 : Actual number of transitions vs. predicted transitions for local controllers.....	202
Figure 5-16 : Accuracy of V_{dd} estimation	204
Figure 5-17 : Overview of power estimation.....	204
Figure 5-18 : Overview of the HYPER-LP system	208
Figure 5-19 : Bus capacitance for the wavelet filter (estimated and extracted)	210
Figure 5-20 : Plot of # of units (a) and clock cycle period (b) vs. # of control cycles for the IIR example	212
Figure 5-21 : Power vs. V_{dd} for the IIR example	213
Figure 5-22 : Optimizing the volterra filter	214
Figure 6-1 : Overview of a future personal communications system.....	217
Figure 6-2 : Desired features of a Portable Multimedia Terminal.....	219
Figure 6-3 : Trends in battery technology for the last 20 years.....	221
Figure 6-4 : Low-power color TFT display (courtesy of SHARP)	222
Figure 6-5 : Partitioning of computation: X-server example	224
Figure 6-6 : Bandwidth requirement for a Frame-maker session.....	225
Figure 6-7 : Latency measurement for pen data [Burghardt94]	227
Figure 6-8 : Approach to high BER; transmit more “data” and minimal “control”	228
Figure 6-9 : Block Diagram of the InfoPad Terminal	230
Figure 6-10 : Block diagram of the protocol chip	232
Figure 6-11 : Packet format coming into the protocol chip.....	233
Figure 6-12 : Datapath for depacketization and demultiplexing	235
Figure 6-13 : Decoding of data types	236
Figure 6-14 : Block diagram of the text/graphics module.....	237

Figure 6-15 : Horizontal timing signals for the SHARP LCD Display	238
Figure 6-16 : Line synchronization clock generation	239
Figure 6-17 : Vertical timing signals for the SHARP LCD Display	239
Figure 6-18 : Field synchronization clock generation	240
Figure 6-19 : Normalized access time vs. V_{dd} for a 64Kbit SRAM.....	241
Figure 6-20 : Parallel memory access enables low-voltage operation	242
Figure 6-21 : Data and address busses for the text/graphics module	244
Figure 6-22 : Conversion from 32-bit frame-buffer data to 4-bit LCD data	245
Figure 6-23 : Optimizing placement for low-power: routing large data/control busses.....	246
Figure 6-24 : Application specific protocol: text/graphics information	247
Figure 6-25 : Gated clocks are used to shut down modules when not used	248
Figure 6-26 : Address generation for the SRAM	249
Figure 6-27 : State information and SRAM clock timing	251
Figure 6-28 : Interface signals between the speech module and the commercial codec	252
Figure 6-29 : Timing relationship between the clock and the data signals	252
Figure 6-30 : Speech downlink datapath	253
Figure 6-31 : Speech uplink datapath	254
Figure 6-32 : Signal swing reduction for memory circuits: FIFO example	256
Figure 6-33 : True single phase clock pipeline register.....	259
Figure 6-34 : Die photo of the protocol chip	260
Figure 6-35 : Die photo of the text/graphics frame-buffer.	262
Figure 6-36 : SRAM block organization	264
Figure 6-37 : Precharge, sense-amp and “Glitch” free tri-state buffer	266
Figure 6-38 : Block Diagram of IPGraphics terminal	269
Figure 6-39 : Overview of IPGraphics (Pen, Speech I/O and Text/graphics) Terminal.....	271
Figure 7-1 : Video compression/decompression using vector quantization.....	275
Figure 7-2 : Effect of channel errors on vector quantized images (16:1).....	277
Figure 7-3 : Block diagram of the video decompression.....	278
Figure 7-4 : Block diagram of the luminance decompression chip	280
Figure 7-5 : Addressing of the frame-buffers and lookup table	281
Figure 7-6 : Die photo of the luminance decompression chip.....	282
Figure 7-7 : Chrominance decompression chip block diagram	284
Figure 7-8 : Simplification of the NTSC system timing.....	287
Figure 7-9 : Protocol for the lookup table and Frame-buffer FIFOs	288
Figure 7-10 : Die photograph of the video controller chip.....	289
Figure 7-11 : Time-multiplexing can destroy signal correlation increasing activity.....	291
Figure 7-12 : Transition activity for time-multiplexing activity	291
Figure 7-13 : Low-voltage 6-bit Digital to Analog converter	293

Figure 7-14 : Die photo of the color space converter and digital to analog converter	296
Figure 7-15 : Block diagram of the video decompression test board	298
Figure 7-16 : Output of video decompression chips running at a supply voltage of 1.3V	298
Figure 8-1 : Energy efficient programmable computation required in portables	300
Figure 8-2 : Event-driven applications alternate between blocked and running states	304
Figure 8-3 : Conventional shutdown approaches	305
Figure 8-4 : States of X Display Server Process	309
Figure 8-5 : Sample trace of X server state under UNIX	311
Figure 8-6 : L-shaped $T_{off}[i]$ versus $T_{on}[i]$ Scatter Plot.....	315
Figure 8-7 : $T_{off}[i]$ versus $T_{off}[i-1]$ Scatter Plot.....	315
Figure 8-8 : Hit ratio curves	317
Figure 8-9 : Slowdown as a function of T_{cost}	318
Figure 8-10 : Trade-off between voltage and hardware-concurrency	320
Figure 8-11 : Speedup and V_{dd} vs. N	329
Figure 8-12 : % communications overhead vs. N	330
Figure 8-13 : Power vs. N	331
Figure 11-1 : Retiming for power minimization.	350

LIST OF TABLES

Table 2-1 :	Area and perimeter capacitances for a 1.2 μ m CMOS technology.....	27
Table 2-2 :	Area and perimeter capacitances for Metal 1 wire of length 100 λ by 3 λ for two different feature sizes.	28
Table 2-3 :	Capacitance breakdown at the module level.....	29
Table 2-4 :	Capacitance breakdown at the datapath level.	29
Table 2-5 :	Truth Table of 2 input NOR gate.....	30
Table 2-6 :	Truth Table of 2 input XOR gate.....	31
Table 2-7 :	Activity comparison of different logic styles.....	35
Table 2-8 :	Output transition probabilities for various static logic gates.	36
Table 2-9 :	Probabilities for tree and chain topologies.....	42
Table 2-10 :	Details of components used for the study in Figure 2-26.	50
Table 2-11 :	Description of circuits used for study in Figure 2-27.	51
Table 3-1 :	Results of architecture based voltage scaling	97
Table 3-2 :	Normalized Area/Power for various supply voltages for Plots 2,3 in Figure3-21.....	98
Table 4-1 :	Computational complexity of VQ encoding algorithms.....	126
Table 4-2 :	Binary and Gray-code representation.	133
Table 4-3 :	Number representation trade-off for arithmetic.....	148
Table 4-4 :	Average number of gate transitions per addition [Callaway92].	157
Table 4-5 :	The influence of layout optimization on physical capacitance.....	168
Table 5-1 :	Breakdown of power consumed for a 11 tap FIR filter before constant multiplications.	184
Table 5-2 :	Breakdown of power consumed for a 11 tap FIR filter after constant multiplications.	184
Table 5-3 :	Capacitance models for some of the library hardware units for a 1.2 μ m CMOS technology.....	190
Table 5-4 :	Results from various local controller units of the selected sample set.	198
Table 5-5 :	Results for the Wavelet Chip (1.2 μ m CMOS technology).....	211
Table 5-6 :	Summary of results.	214
Table 6-1 :	Bandwidth requirement for various data types.	226
Table 6-2 :	Statistics for the protocol chip.	260
Table 6-3 :	Summary of power reduction techniques applied to the protocol chip.	261

Table 6-4 :	Statistics for the text/graphics frame-buffer chip.....	267
Table 6-5 :	Summary of power reduction techniques applied to the frame-buffer chip.	268
Table 7-1 :	Computational complexity of the Discrete Cosine Transform.	274
Table 7-2 :	Asymmetrical algorithm required for low-power implementation.	276
Table 7-3 :	Summary of power reduction techniques applied to the luminance decompression chip.....	282
Table 7-4 :	Summary of power reduction techniques applied to the YIQ -> RGB conversion.....	292
Table 7-5 :	Summary of power reduction techniques applied to the DAC.....	295
Table 7-6 :	Summary of the video decompression chipset implemented in 1.2 μ m CMOS.....	297
Table 8-1 :	Analysis of X Server Traces for Potential Energy Reduction	311
Table 8-2 :	Summary of Energy Reduction.....	319

ACKNOWLEDGMENTS

My advisor, Bob Brodersen, provided great guidance during all phases of my project and was the most important person in making this project a success. He defined the low-power research area and spent a lot of time helping me define my project. I was spoiled with incredible computational resources, research facilities and support staff which made research a lot more fun. I am very grateful for the exposure he provided and he always encouraged me to look beyond my project to get the “big picture”. I am also grateful for his encouragement to publish and I really appreciate all his support. Of course, he did not succeed in getting me to lose weight!

Many students, faculty members and staff made significant technical contributions to this thesis. I would like to thank Prof. David Brillinger and Prof. John Wawrzynek for being on my thesis committee and providing invaluable feedback. I would also like to thank Prof. Larry Rowe for being on my qualifying committee and providing many very useful suggestions.

Phil Schrupp, Susan Mellers, and Ken Lutz setup and supported the infrastructure for designing and testing boards. Brian Richards was always around to answer questions about Lager and provided valuable suggestions concerning the design of the InfoPad chipset. I would like to thank Tom Boot, Peggy Brown, Carole Frank, Elise Mills, and Corey Schaffer for helping me with administrative issues. Heather Brown was also always very helpful with administrative matters. I would like to thank Kevin Zimmerman for doing a wonderful job supporting the computer system.

My venture into the CAD world would not have been possible without the help of Miodrag Potkonjak (Mr. Transformations) and Prof. Jan Rabaey. I appreciate all the invaluable feedback from Prof. Jan Rabaey on low-power design methodologies. I would like to thank Sean Huang for coding the local transforms used in the HYPER-LP system (Chapter 5), Paul Landman for providing software to generate data with different statistics, Renu Mehra for generating the various controllers and building the controller model, and the rest of the HYPER team for software support. The CAD tool, HYPER-LP, was started as a EE244 course project (CAD for IC's) and I

appreciate all the inputs from Dr. Lou Scheffer who was teaching the course. I especially thank Ingrid Verbauwhede for overall feedback on the project and I appreciate all the times she sat through my practice talks for conferences.

The design and demonstration of the InfoPad chipset and terminal would not have been possible without the help of several people. The first generation radio and protocol group consisting of Bill Barringer, Kathy Lu, Trevor Pering, and Tom Truman was responsible for designing the radio module that interfaced with the chipset described in Chapters 6 and 7. Tom Burd (who designed the radio receiver module), Andy Burstein (who designed the SRAM), and Shankar Narayanaswamy (responsible for the pen interface) contributed to the design of the chipset. Tom Burd spent several sleepless nights in Cory debugging the first generation IPGraphics terminal (not just because he had to, but because he has no life! - seriously, thanks Tom). I want to thank the students (especially Tom Burd and Andy Burstein) who developed the low-power cell-library and provided the infrastructure to design the chipset for the InfoPad terminal. Roger Doering and Chuck Smith were responsible for the packaging aspects of the InfoPad. Fred Burghardt, Brian Richards, and Shankar Narayanaswamy designed the basestation software and applications. It was just amazing how so many different pieces fit together in a short period of time before the InfoPad demo in October 1993.

For the video chipset, Ian O'Donnell prototyped the logic to verify the modified NTSC timing suggested by Ken Nishimura (the NTSC God) on a Xilinx. Ian was also an incredible help during the testing phase of the video chips - with all the skills he has picked up, he is going to be a permanent resident of Cory! I also want to thank Cormac Conroy and Robert Neff for their valuable suggestions on the DAC design. I also want to thank Dr. Tom Lookabaugh and Prof. Eve Riskin for providing me source code for VQ which I used as a base for the InfoPad video.

I want to thank Mani Srivastava for all the great technical discussions. He always has great solutions to every and any technical problem. I really enjoyed working with him on energy efficient programmable computation (Chapter 8). He was responsible for the modifications to the X-server and the analysis of the shutdown approaches.

I would like to thank my "other" family in 550 Cory Hall who made my life more enjoyable. My cubicle mates (old and new) included Kevin Komegay who made sure that I was eating enough so

he can call me Fats, Mani Srivastava and Jane Sun who always entertained me with “lively” discussions, Ian O’Donnell who is responsible for the best arch ever built in 550I using straws leftover from lunch meetings (he works on this when he is not breaking monitors), Kevin Stone (along with Lapoe Lynn) broke my ribs during a once in a life-time game of soccer, and Shankar Narayanaswamy who showed me the dangers of typing too much.

The other students residents of 550 who made my life more enjoyable included Andy Burstein who sits on big rubber ball; Tom Burd who stole money from me on 49er bets; Craig Teuscher, David Lidsky and Anthony Stratakos who played 2 on 2 tackle football with me in 550 cory; Monte Mar the “chip hacker” who had answers to all chip problems; Scarlett Wu who always had food on her desk; Alfred K. Yeung the other major food supplier; Sam “Tools” Sheng considered as God by many - I would like to thank him for his contributions on the low-power design methodology development and for providing me with the correlator example; Lisa Guerra who gave me those precious 14-day parking permits and who was always fun to talk with; Sekhar Narayanaswami who helped me practice my Tamil; Rajiv Murgai who was always around in the middle of the night to talk to; Renu Mehra who has collected leftover chips from group meetings for several years and has a big stock in her lower drawer; Arthur Abnous who hates being called Archer; Winston Sun who taught me that the answer to all problems is 10 and has always been a great friend; My Le who is always bothering me when she is not vacationing in London; Charmine Tung who introduced me to the Latte and was always there when I was bored or depressed; Weiling Chu who is always very supportive; Randy “Scumbag” Allmon who hated going to the Korean restaurant with me and always ended up choosing where we ate; Greg Uehara who brings us chocolates from Hawaii every time he visits; and Cormac Conroy who was very supportive and always fun to talk with.

I would also like to thank IBM for providing me with a fellowship for the academic year 1993-1994. I would also like to thank ARPA who funded this project.

Finally, I would like to acknowledge my family members. This thesis could not have been possible without the continued support of my parents. I am very grateful for their constant encouragement for me to strive for my best. They have been the greatest source of inspiration to me. I would like to thank Mrs. Charlotte Coe who provide great emotional and financial support

throughout my college years. She is a great inspiration to me. My wife Karthiyayani was very patient and accommodating to my weird schedules, especially during chip deadlines when we spent “quality” time from 2-3A.M! I would also like to thank my brother Nagarajan and sister-in-law Suchitra for their continued support.

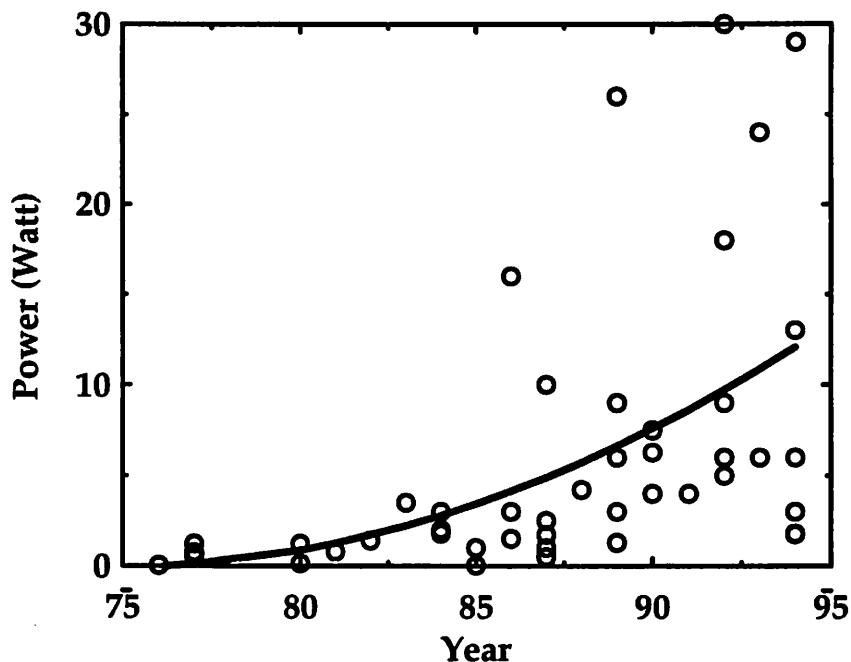
CHAPTER 1

Introduction

With much of research efforts of the past ten years directed toward increasing the speed of digital systems, present-day technologies possess computing capabilities which make possible powerful personal workstations, sophisticated computer graphics, and multi-media capabilities such as real-time speech recognition and real-time video. High-speed computation has thus become the expected norm from the average user, instead of being the province of the few with access to a powerful mainframe. Likewise, another significant change in the attitude of users is the desire to have access to this computation at any location, without the need to be physically tethered to a wired network. A major factor in the weight and size of portable devices is the amount of batteries which is directly impacted by the power dissipated by the electronic circuits.

Even when power is available in non-portable applications, the issue of low power design is becoming critical. Up until now, this power consumption has not been of great concern, since large packages, cooling fins and fans have been capable of dissipating the generated heat. However, as the density and size of the chips and systems continues to increase, the difficulty in providing adequate cooling might either add significant cost to the system or provide a limit on the amount of

functionality that can be provided. Figure 1-1 shows a plot of the power consumption for various micro-processors that have been reported at the International Solid-state Circuits Conference (ISSCC) for the last 20 years. The obsession of increasing the clock rate has resulted in single chip power levels in excess of 30W.



demanding the same computation capabilities as found in desktop machines. Equally demanding are developments in personal communications services (PCS), such as the current generation of digital cellular telephony networks which employ complex speech compression algorithms and sophisticated radio modems in a pocket sized device. Even more dramatic are the proposed PCS applications, with universal, portable multimedia access supporting full-motion digital video and control via speech recognition. In these applications, not only will voice be transmitted via wireless links, but data and video as well. Indeed, it is apparent that portability can no longer be associated with low-throughput; instead, vastly increased capabilities, actually in excess of that demanded of fixed workstations, must be placed in a low-power, portable environment.

In spite of these concerns, until recently, there has not been a major focus on a design methodology of digital circuits which directly addresses power reduction, with the focus rather on ever faster clock rates and logic speeds. The approach which will be presented here, takes another viewpoint, in which all possible aspects of a system design are investigated with the goal of reducing the power consumption. These considerations range from the technology being used for the implementation, the circuit and logic topologies, the digital architectures and even the algorithms being implemented. What is assumed is that the application, which is desired to be implemented with low power is known, and trade-offs can be made as long as the functionality required of this application is met within a given time constraint.

Maintaining a given level of computation or throughput is a common concept in signal processing and other dedicated applications, in which there is no advantage in performing the computation faster than some given rate, since the processor will simply have to wait until further processing is required. This is in contrast to general purpose computing, where the goal is often to provide the fastest possible computation without bound. One of the most important ramifications of only maintaining throughput is that it enables an architecture driven voltage scaling strategy, in which aggressive voltage reduction is used to reduce power, and the resulting reduction in logic speed is compensated through parallel architectures to maintain throughput. However, the techniques

presented are also applicable to the general purpose environment, if the figure of merit is the amount of processing per unit of power dissipation (e.g. MIPS/Watt). Since in this case the efficiency in implementing the computation is considered and voltage scaling decreases the energy expended per evaluation.

The optimization to minimize area, has only been secondary to the fixation on increasing circuit speed and again this should be examined with respect to its effect on power consumption. Some of the techniques that will be presented will come at the expense of increased silicon area and thus the cost of the implementation will be increased. The desirability of this trade-off can only be determined with respect to a given market situation, but in many cases a moderate increase in area can have substantial impact on the power requirements. It is clear that if power reduction is more important than increasing circuit clock rate, that the area consumed by large clock buffers, power distribution busses and predictive circuit architectures would be better spent to reduce the power dissipation.

1.1 Overview of Thesis

1.1.1 Low-power Design Methodologies [Chapters 2,3, and 4]

Chapter 2 reviews the sources of power consumption in CMOS circuits. Four components of power consumption will be described: switching power, short-circuit power, leakage power and static power. For a properly designed CMOS circuit, it is usually the switching component of power which dominates, contributing to more than 90% of the total power. The concept of switching activity (the number of capacitive transitions per clock cycle) will be introduced and the various factors affecting switching activity will be described. The effect of voltage scaling on the circuit performance will be described. This chapter will provide the fundamentals of power consumption in CMOS circuits and will provide the background to understand the various power minimization techniques presented in Chapters 3 and 4.

Approaches to Power Supply Voltage Scaling

It is evident that methodologies for the design of high-throughput, low-power digital systems are needed. Since power is proportional to the square of the supply voltage, it is clear that lowering the power supply voltage, V_{dd} , is the key to power reduction. Lowering V_{dd} however comes at the cost of increased circuit delays and therefore lower functional throughput. Chapter 3 will present various approaches to scaling the power supply voltage without loss in throughput.

Fortunately, there are clear technological trends that give us a new degree of freedom, so that it is possible to satisfy contradictory requirements of low-power and high throughput. Scaling of device feature sizes, along with the development of high density, low-parasitic packaging, such as multi-chip modules, will alleviate the overriding concern with the numbers of transistors being used. When MOS technology has scaled to $0.2\mu\text{m}$ minimum feature size it will be possible to place from $1-10 \times 10^9$ transistors in an area of 8" by 10" if a high-density packaging technology is used. The question then becomes how can this increased capability be used to meet a goal of low power operation. Previous analyses on the question of how to best utilize increased transistor density at the chip level, concluded that for high-performance microprocessors the best use is to provide increasing amounts of on-chip memory [Patterson80]. It will be shown here that for computationally intensive functions that the best use is to provide additional circuitry to parallelize the computation - our architecture driven voltage scaling strategy.

Another important consideration, particularly in portable applications, is that many computation tasks are likely to be real-time; the radio modem, speech and video compression, and speech recognition all require computation that is always at near-peak rates. Conventional schemes for conserving power in laptops, which are generally based on power-down schemes, are not appropriate for these continually active computations. On the other hand, there is a degree of freedom in design that is available in implementing these functions, in that once the real-time requirements of these applications are met, there is no advantage in increasing the computational

throughput. This fact, along with the availability of increasing numbers of transistors in scaled technology, allows a strategy to be developed for architecture design, which if it can be followed, will be shown to provide significant power savings.

Figure 1-2 shows an overview of various approaches to power supply voltage scaling which range from optimizing the technology and circuits used to the architectures and algorithms. The various techniques currently used to scale the supply voltage will be reviewed which include optimizing the technology and the device reliability. These techniques dictate a limited amount of voltage scaling, typically going from 5V down to 3V. Our architecture driven voltage scaling strategy will be presented which trades area for lower power and allows for voltage scaling down to 1V without loss in system performance. Combining architecture optimization with threshold voltage reduction can result in scaling of the supply voltage to the sub-1V range. The key to architecture driven voltage scaling is the exploitation of concurrency. For this, algorithmic transformations will be used. Chapter 3 will also describe the various support circuitry required for low voltage operation, which includes high-efficiency DC/DC converters and level converting.

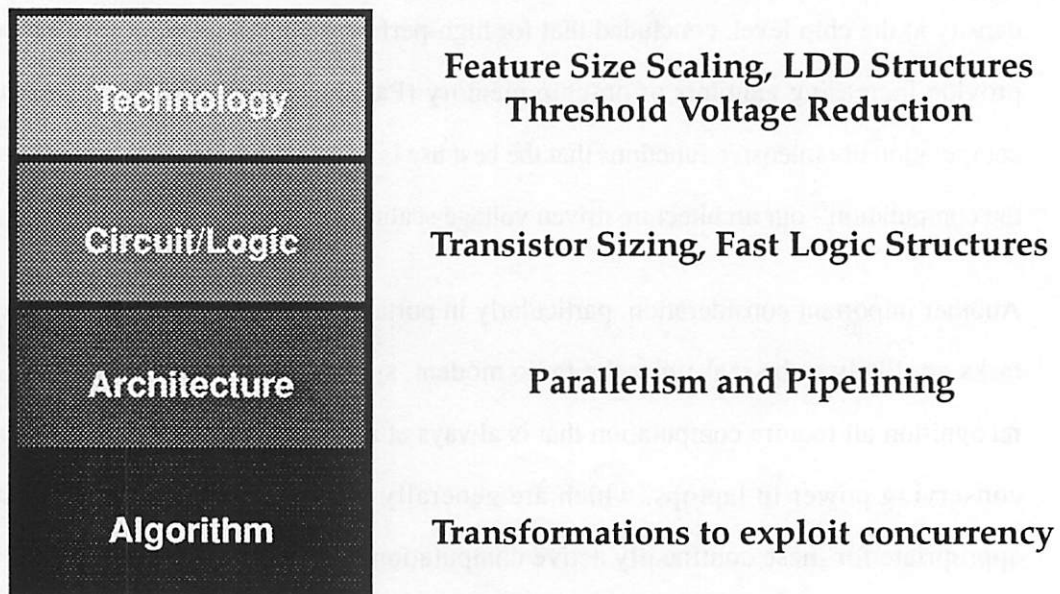


Figure 1-2 : A system level approach to supply voltage scaling.

Minimizing Switched Capacitance

Since CMOS circuits do not dissipate power if they are not switching, a major focus of low power design is to reduce the switching activity to the minimal level required to perform the computation. This can range from simply powering down the complete circuit or portions of it, to more sophisticated schemes in which the clocks are gated or optimized circuit architectures are used which minimize the number of transitions. An important attribute which can be used in circuit and architectural optimization, is the correlation which can exist between values of a temporal sequence of data, since switching should decrease if the data is slowly changing (highly positively correlated). An example of the difference in the number of transitions which can be obtained for a highly correlated data stream (human speech) versus random data is shown in Figure 1-3 - the transition activity for a few registers in an FIR filter design. For an architecture which does not destroy the data correlation, the speech data switches 80% less capacitance than the random input. In addition, the sequencing of operations can result in large variations of the switching activity due to these temporal correlations.

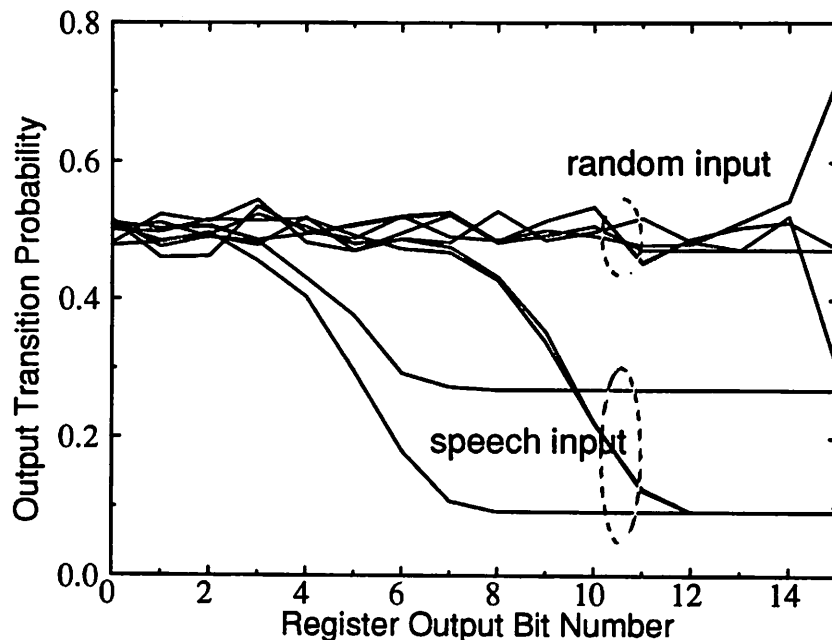


Figure 1-3 : Dependence of activity on statistics: correlated vs. random input.

Figure 1-4 shows an overview of the techniques described in Chapter 4 for reducing the effective switched capacitance. The knowledge about signal statistics can be exploited to reduce the number of transitions. Various techniques are described which once again span all levels of the system design ranging from the technology and circuits used to the architectures and algorithms. At the technology level, packaging and SOI can be used to minimize the physical capacitance. Logic minimization and logic level power down is a key technique to keeping the transition activity to a minimum. Data dependent power down will be introduced which is a dynamic technique to reduce switched capacitance. At the architecture level, the trade-offs between time-multiplexed vs. parallel architectures, sign-magnitude vs. two's complement datapaths, ordering of operations, etc. will be studied for their influence on the switched capacitance. At the algorithmic level, the computational complexity (minimizing the number of operations) and data representation will be optimized for minimizing the switched capacitance.

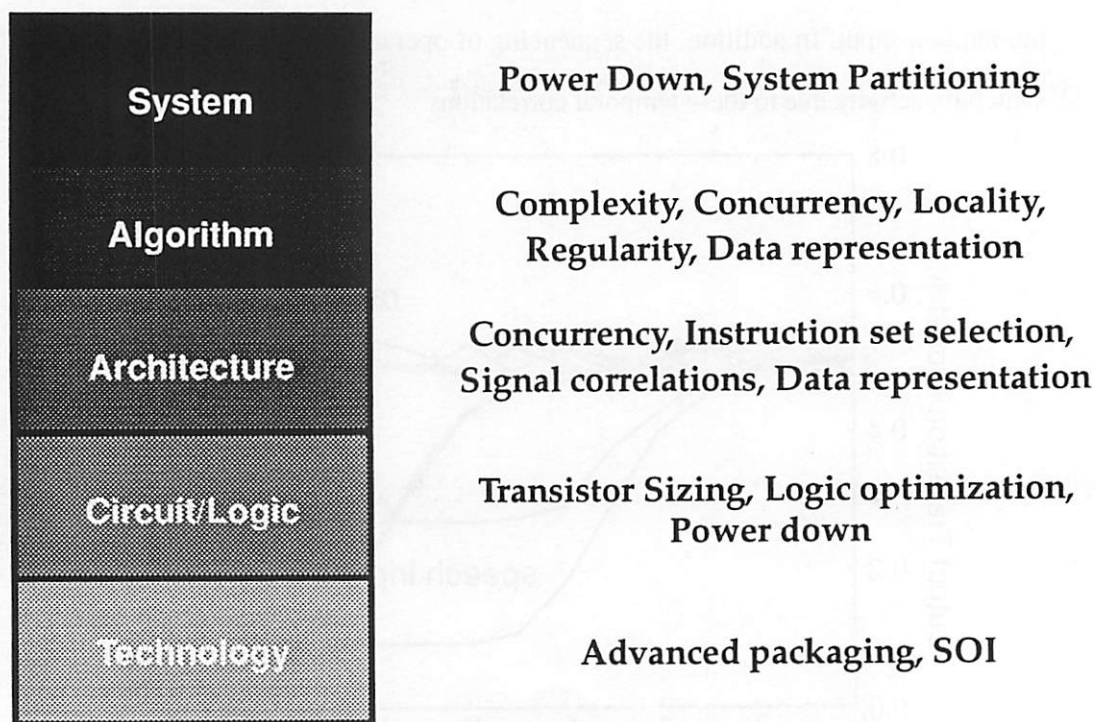


Figure 1-4 : A system level approach to minimizing the switched capacitance.

1.1.2 CAD Tools for Low-power Design [Chapter 5]

The focus of Chapter 5 is on automatically finding computational structures that result in the lowest power consumption for DSP applications that have a specific throughput constraint given a high-level algorithmic specification. The basic approach is to scan the design space utilizing various algorithmic flowgraph transformations, high-level power estimation, and efficient heuristic/probabilistic search mechanisms. While algorithmic transformations have been successfully applied in high-level synthesis with the goal of optimizing speed and/or area, they have not addressed the problem of minimizing power.

There are two approaches taken to explore the algorithmic design space to minimize power consumption. First, is the exploitation of concurrency which enables circuits to operate at the lowest possible voltage without loss in functional throughput. Second, computational structures are used that minimize the effective capacitance that is switched at a fixed voltage: through reductions in the number of operations, the interconnect capacitance, the glitching activity, and internal bit widths and using operations that require less energy per computation.

1.1.3 System Design Example: A Portable Multimedia Terminal [Chapters 6 and 7]

Future terminals will allow users to have untethered access to multimedia information servers that are interconnected through high bandwidth network backbones. Chapters 6 and 7 describe a low-power chipset for such a terminal, which is designed to transmit audio and pen input from the user to the network on a wireless uplink and will receive audio, text/graphics and compressed video from the backbone on the down link. The portability requirement resulted in the primary design focus to be on power reduction. Six chips provide the interface between a high speed digital radio modem and a commercial speech codec, pen input circuitry, and LCD panels for text/graphics and full-motion video display. This chipset demonstrates the various low-power techniques that are developed in Chapters 3 and 4.

A system approach to power reduction will be described which involves optimizing the physical design, the circuitry, the logic design, the architectures and system partitioning. This includes minimizing the number of operations at the algorithmic level, using the architecture driven voltage scaling strategy described in Chapter 3 to operate at supply voltages as low as 1.1V, using gated clocks to minimize the effective clock load and the switched capacitance in logic circuits, optimizing circuits to minimize the voltage swings and switched capacitance, using self-timed memory circuits to eliminate glitching on the data busses, utilizing a low-power cell-library that features minimum sized devices, optimized layout, and single-phase clocking, and using an activity driven place and route. The chips provide protocol conversion, synchronization, error correction, packetization, buffering, video decompression, and D/A conversion at a total power consumption of less than 5 mW - orders of magnitude lower than existing commercial solutions.

1.1.4 Energy Efficient Programmable Computation [Chapter 8]

Chapter 8 describes approaches for energy efficient programmable computation. Two key architectural approaches for energy efficient programmable computation are described: predictive shutdown and concurrency driven supply voltage reduction. Many computations are “event-driven” in nature with intermittent computation activity triggered by external events and separated by periods of inactivity - examples include X server, communication interfaces etc. An obvious way to reduce average power consumption in such computations would be to shut the system down during periods of inactivity. Shutting the system down - which would make power consumption zero or negligible - can be accomplished either by shutting off the clock ($f=0$) or in certain cases by shutting off the power supply ($V_{dd} = 0$). However the shutdown mechanism has to be such that there is no or little degradation in speed - both latency and throughput are usually important in event-driven computations. There are two main problems in shutdown that are addressed- *how* to shutdown, and *when* to shutdown. An aggressive shut down strategy based on a predictive technique will be presented which can reduce the power consumption by a large factor compared to the straightforward conventional schemes where the power down decision is based

solely on a predetermined idle time threshold.

Not all computation implemented as software is “event-driven” in nature - data-flow functions such as DSP are “continuous” in nature. Obviously, shutdown is not an effective mechanism for these systems. An alternative strategy is to operate at the lowest possible supply voltage, which, unfortunately, comes at the expense of reduced circuit speed. However, if only throughput and not latency is the metric of speed - as is true for many “continuous” applications such as speech coding - the reduction in circuit speed can be compensated for by architectural techniques like pipelining and parallelism. Compiler techniques for effective parallelization and pipelining play an important role in the success of this strategy to energy efficiency.

CHAPTER 2

Power Consumption in CMOS Circuits

In order to minimize the power consumption of a design, the various power components and their relative importance with respect to the overall power consumption must be examined. The two important types of power consumption relevant to circuit design are peak power consumption and average power consumption. Peak power affects both the circuit lifetime and performance. Excessive instantaneous current drawn from the power supply results in large voltage drops across power/ground lines due to the resistance of power wires and results in large instantaneous power dissipation. Voltage drops in the power supply voltage result in power supply glitches which in turn cause erroneous evaluation of logic circuits and results in soft errors. Large instantaneous power dissipation causes overheating of the devices which ultimately causes degradation in circuit performance.

The design of portable devices certainly requires consideration of the peak power consumption for reliability and proper circuit operation, but a much more important consideration is the time averaged power consumption which is directly proportional to the battery weight required to operate circuits for a given amount of time. In fact, the approaches which will be presented to

minimize the average power consumption will also reduce the peak power consumption and improve reliability.

There are four sources of power dissipation in digital CMOS circuits which are summarized in the following equation:

$$P_{avg} = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static} = \alpha_{0 \rightarrow 1} C_L \cdot V \cdot V_{dd} \cdot f_{clk} + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} + I_{static} \cdot V_{dd} \quad (\text{EQ 1})$$

The first term represents the switching component of power, where C_L is the loading capacitance, f_{clk} is the clock frequency and $\alpha_{0 \rightarrow 1}$ is the probability that a power consuming transition occurs (the activity factor). In most cases, the voltage swing, V , is the same as the supply voltage, V_{dd} ; however, some logic circuits, such as in single-transistor pass-transistor implementations, the voltage swing on some internal nodes may be slightly less. The second term is due to the direct-path short circuit current, I_{sc} , which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. The third term is due to the leakage current, $I_{leakage}$, which can arise from substrate injection and sub-threshold effects, is primarily determined by fabrication technology considerations. Finally, static currents, I_{static} , arise from circuits that have a constant source of current between the power supplies (such as bias circuitry, pseudo-NMOS logic families, etc.). The four components of power consumption are described in detail below.

2.1 Switching Component of Power

The switching component of power arises when energy is drawn from the power supply to charge parasitic capacitors (made up of gate, diffusion, and interconnect). Every parasitic capacitor in a circuit does not consume switching power every single clock cycle, but rather typically at a lower rate called the switching activity. The components of switching activity are described in this section and opportunities for power reduction are explored.

2.1.1 Computation of Switching Energy Per Transition

The energy drawn from the supply for each power consuming transition is derived for various types of circuits implemented in CMOS technology.

Conventional CMOS Circuits with Rail-to-Rail Swing

The switching or dynamic component of power consumption arises when the output of a CMOS circuit is charged through the power supply or is discharged to ground. The switching component of energy drawn from the power supply for a $0 \rightarrow V_{dd}$ transition at the output of a CMOS gate is given by $C_L V_{dd}^2$, where C_L is the physical load capacitance at the output node and V_{dd} is the supply voltage. Figure 2-6 shows an equivalent circuit model for computing the switching component of power. The switching component of energy is relatively independent of the function being performed (i.e. the interconnection network of the NMOS and PMOS transistors) and the slope or rise times for the input signals of the CMOS gate (i.e. the short-circuit currents are ignored) and is only dependent on the output load capacitance and the power supply voltage.

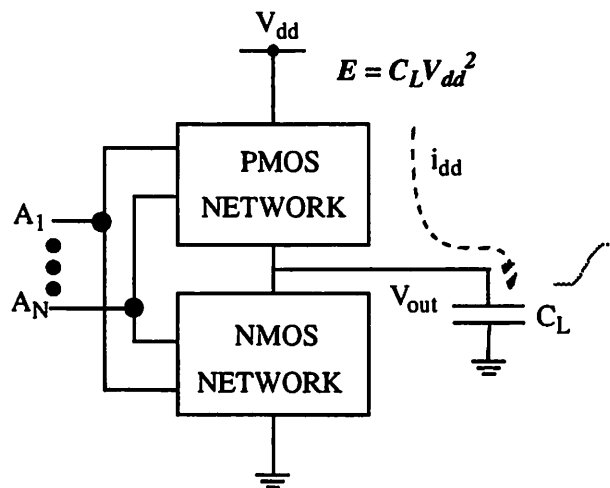


Figure 2-1 : Circuit model for computing the dynamic power of a CMOS gate.

To compute the average energy consumed per switching event for the generic CMOS gate shown in Figure 2-6, consider any input transition which causes a $0 \rightarrow V_{dd}$ transition at the output of the

CMOS gate. For this particular transition (ignoring short-circuit currents) the instantaneous power and energy drawn can be computed as follows:

$$P(t) = \frac{dE}{dt} = i_{supply} \cdot V_{dd} \quad (\text{EQ 2})$$

where i_{supply} is the instantaneous current being drawn from the supply voltage V_{dd} . Therefore, the average energy drawn from the power supply is given by:

$$E_{av} = \int_0^T p(t) dt = \int_0^T \frac{dE}{dt} dt = V_{dd} \int_0^T i_{supply}(t) dt = V_{dd} \int_0^{V_{dd}} C_L dV = C_L \cdot V_{dd}^2 \quad (\text{EQ 3})$$

From Equation 3, it is clear that the energy drawn from the power supply for a $0 \rightarrow V_{dd}$ transition at the output is $C_L V_{dd}^2$ regardless of the function being performed by the CMOS gate. For this transition, the energy stored in the output load capacitor is given by:

$$E_{cap} = \int_0^T p_{cap}(t) dt = \int_0^T V_{cap} i_{cap}(t) dt = \int_0^{V_{dd}} C_L V_{cap} dV = \frac{1}{2} C_L \cdot V_{dd}^2 \quad (\text{EQ 4})$$

Therefore, half of the energy drawn from the power supply is stored in the output capacitor and half of the energy is dissipated in the PMOS network. For the $V_{dd} \rightarrow 0$ transition at the output, no charge is drawn from the supply and the energy stored in the capacitor ($1/2 C_L V_{dd}^2$) is dissipated in the pull-down NMOS network.

Circuits Employing Voltage Clamping

The energy drawn from the power supply per switching event for a standard CMOS gate which has rail to rail output swing was calculated in Equation 3. This equation requires slight modification to account for logic that does not have rail to rail swing. This situation is typical for circuit topologies that use single transistor NMOS pass gates which do not have rail to rail swing. Figure 2-2 shows one example in which an NMOS transistor is used to charge an output load capacitor. For this

transition, the output rises only to $V_{dd} - V_t$, rather than all the way to the rail voltage.

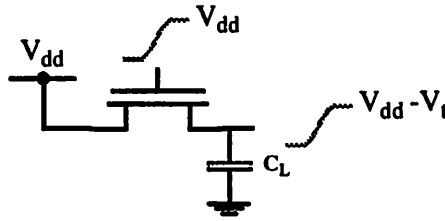


Figure 2-2 : Switching power for reduced swing logic.

The energy drawn from the supply for a $0 \rightarrow V_{dd} - V_t$ transition at the output is given by:

$$E_{av} = \int_0^T \frac{dE}{dt} dt = V_{dd} \int_0^T i_{supply}(t) dt = V_{dd} \int_0^{(V_{dd} - V_t)} C_L dV = C_L \cdot V_{dd} \cdot (V_{dd} - V_t) \quad (\text{EQ 5})$$

Dynamic Charge Sharing Circuits

Dynamic logic circuits suffer from charge sharing problems which result in extra parasitic switching power. For N-tree dynamic logic (with PMOS precharge transistor), the precharge operation forces the output to V_{dd} . During the evaluation operation, depending on the number of NMOS switches turn on, the output voltage is reduced to

$$V_{out} = \frac{C_L}{C_L + C_{int}} \cdot V_{dd} \quad (\text{EQ 6})$$

where C_{int} the sum of the internal node capacitances and C_L is the output load capacitance. Figure 2-3 shows a simple NAND gate example. Here input A goes high during the evaluate period while node B remains LOW. Therefore, the charge on C_L is redistributed between C_L and C_{int} during the evaluate period.

Therefore, during the next precharge period, the output has to be precharged back to V_{dd} . Therefore, the power consumed for this operation is obtained similar to the analysis of the previous section and is given by:

$$V_{out} = C_L \cdot V_{dd} \cdot \Delta V = C_L \cdot V_{dd} \cdot \left(V_{dd} - \left(\frac{C_L}{C_L + C_{int}} \cdot V_{dd} \right) \right) = C_L \cdot V_{dd}^2 \cdot \frac{C_{int}}{C_L + C_{int}} \quad (\text{EQ 7})$$

This is significant if the internal node capacitance is comparable to the load capacitance. Charge sharing is a very important consideration for evaluating both functionality and power consumption. If the output drops more than the threshold voltage of the PFET transistor of the next stage (for a NORA implementation), this can result in incorrect evaluation. The effect is less for Domino logic gates since properly designed inverter gates following the logic prevent propagation of low logic levels.

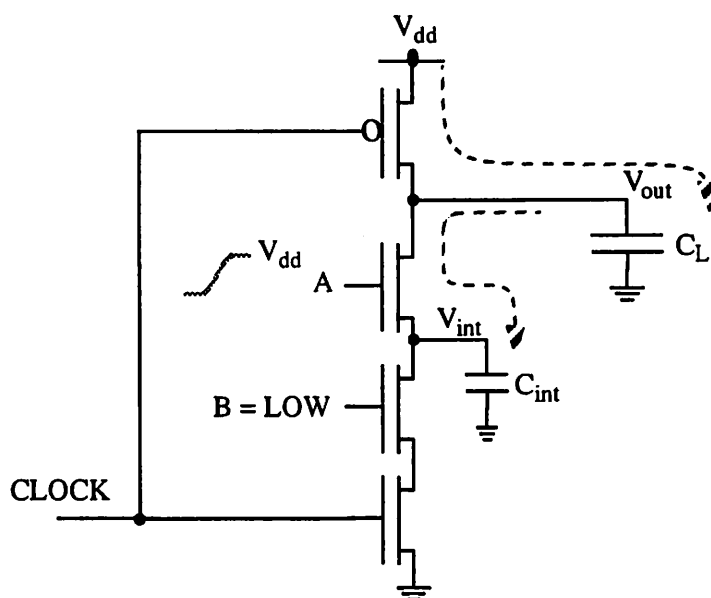


Figure 2-3 : Charge sharing in dynamic circuits result in “extra” switching power.

Adiabatic Circuits

In the conventional CMOS circuits described earlier, the output load capacitance is charged to the power supply voltage through MOSFET switches which have a certain ON resistance. If the load has physical capacitance C_L , then a charge of $C_L V_{dd}$ passes through the resistance. The voltage drop across the resistance varies from V_{dd} initially to a final value of zero. Therefore, the energy dissipated in the resistance for the 0 to 1 transition at the output is $Q\bar{V} = C_L V_{dd} V_{dd}/2$, where \bar{V} is the average voltage drop across the resistance. Adiabatic switching circuits reduce the dissipation

in the resistances by reducing the average voltage drop, \bar{V} (the charge that has to be transferred from the supply is set by the output capacitance and the voltage). Figure 2-4 shows such a circuit, which is a stepwise driver for a capacitance C_L [Svensson94].

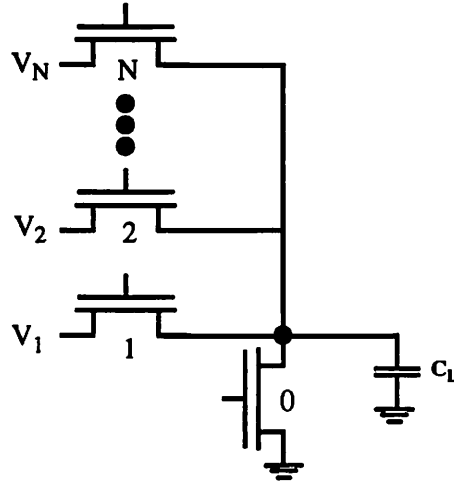


Figure 2-4 : Stepwise driver for a capacitive load C_L .

Here a bank of N voltage supplies with evenly distributed voltage is used: $V_i = (i/N)V_{dd}$. To charge the load, V_1 through V_N are connected to the load in succession (by closing switch 1, opening switch 1, closing switch 2, etc.). To discharge the load, V_{N-1} through V_1 are switched and then switch 0 is closed, shorting the output to ground. For each step, the dissipation is equal to the charge transferred and the average voltage drop across the switch resistance:

$$E_{step} = Q \cdot V_{avg} = C_L \cdot \frac{V_{dd}}{N} \cdot \frac{V_{dd}}{2N} = \frac{\frac{1}{2} \cdot C_L \cdot V_{dd}^2}{N^2} \quad (\text{EQ 8})$$

To charge the output all the way to the supply voltage V_{dd} , N steps are required and the total energy dissipated is:

$$E_{total} = N \cdot \frac{\frac{1}{2} \cdot C_L \cdot V_{dd}^2}{N^2} = \frac{E_{conventional}}{N} \quad (\text{EQ 9})$$

This simple analysis suggests that by using adiabatic switching as opposed to conventional

CMOS, the energy per cycle can be reduced by a factor of N .

2.1.2 Definition of Switching Activity Factor, α

The energy drawn for each $0 \rightarrow V_{dd}$ transition at the output of the CMOS gate is $C_L V_{dd}^2$. However, a logic gate will typically not make a $0 \rightarrow V_{dd}$ transition every single clock cycle. Therefore, estimating the power consumption for a CMOS gate as $C_L V_{dd}^2 f$ (where f is the rate at which the inputs to the gate arrive) can be very inaccurate and the equation has to be modified to include a node transition probability.

Let $f = 1/T$ be the rate at which the primary inputs A_1 through A_N arrive (or the clock rate for the system) for the CMOS gate shown in Figure 2-6. Consider N clock periods and let $n(N)$ be the number of $0 \rightarrow V_{dd}$ output transitions in the time interval $[0, N)$. The average switching power consumption of the CMOS gate for this interval is given by:

$$P_N = C_L \cdot V_{dd}^2 \cdot \frac{n(N)}{N} \cdot f_{clk} \quad (\text{EQ 10})$$

Since average power consumption corresponds to the average number of switching transitions for an extended period of time, the average power is given by:

$$P_{avg} = \lim_{N \rightarrow \infty} P_N = C_L \cdot V_{dd}^2 \cdot \lim_{N \rightarrow \infty} \frac{n(N)}{N} \cdot f_{clk} \quad (\text{EQ 11})$$

The limit term in the above equation is the expected value of the average number of transitions per clock cycle. That is,

$$E\left[\frac{n(N)}{N}\right] = \lim_{N \rightarrow \infty} \frac{n(N)}{N} \quad (\text{EQ 12})$$

where $E[\]$ denotes the expected value operator. The average power can then be expressed as

$$P_{avg} = E[P_N] = C_L \cdot V_{dd}^2 \cdot E\left[\frac{n(N)}{N}\right] \cdot f_{clk} \quad (\text{EQ 13})$$

the expected number of power consuming transition per clock cycle will be referred to as the node transition activity factor, α .

$$\alpha_{0 \rightarrow 1} = E\left[\frac{n(N)}{N}\right] \quad (\text{EQ 14})$$

The average switching component of power for a CMOS gate with an output load capacitance C_L is therefore given by:

$$P_{switching} = \alpha_{0 \rightarrow 1} C_L V_{dd}^2 f_{clk} \quad (\text{EQ 15})$$

where f_{clk} is the clock frequency (the rate at which the registers between logic are clocked) and α is the node transition activity or the fraction of the time the node makes a power consuming transition (i.e a 0 -> 1 transition) inside the clock period.

The assertion made in the previous section and in Equation 15 stating that the switching energy only depends on the output load capacitor and is independent of the logic function is true under the assumption that the output load capacitor is the dominant capacitance. Depending on the input transition pattern, however the internal nodes of a CMOS gate, which have diffusion capacitance to the substrate, can be making power consuming transitions even if the output does not transition. Therefore, the effective switching power consumption must also include the transition activity for the internal nodes that make a 0->1 transition drawing charge from the power supply. Figure 2-5 illustrates the extra transition activity for a CMOS gate due to internal node transitions.

amount x_d (as seen in Figure 2-7), the lateral diffusion.

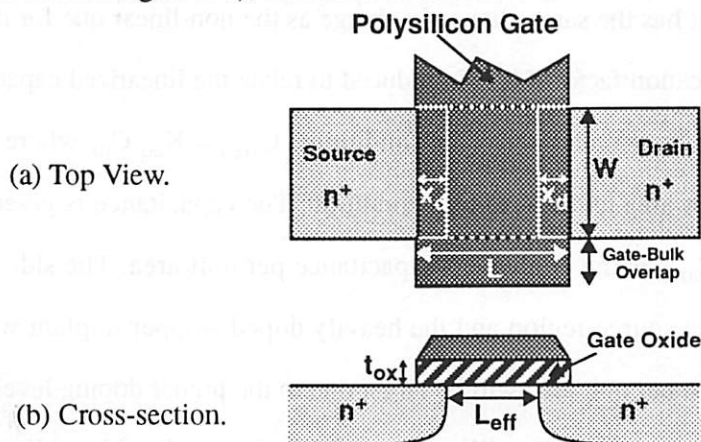


Figure 2-7 : Overlap capacitance for a MOS device.

Since M1 and M2 are either in cut-off or in the saturation region (in steady state), it is reasonable to assume that the only contributions to this capacitance are the *overlap capacitances* of both M1 and M2. This is a consequence of the observation that in those operation modes the gate capacitance is either completely between gate and bulk (cut-off) or gate and source (saturation). In the lumped capacitor model, the gate-drain capacitor is replaced by a capacitor to ground whose contribution is counted twice, due to the *Miller Effect*: the effective voltage change over the gate-drain capacitor during a low-high or high-low transition is actually twice the output voltage swing, as the terminals of the capacitor are moving in opposite directions. Since x_d is a technology parameter the gate to drain capacitance is given by $C_{gd} = 2 C_{GD0} W$ (where C_{GD0} is the overlap capacitance per unit width).

Drain/Source Junction Capacitances, C_{db1} and C_{db2} : There are two components to the drain/source junction capacitance - bottom plate junction capacitance and the side-wall junction capacitance. Figure 2-8 shows the cross section of a MOS device which illustrates the two components. The bottom plate capacitance is the reverse biased pn-junction formed between the drain and the lightly doped substrate. Such a capacitor is, unfortunately, quite non-linear and depends heavily on the applied voltage. In the lumped model, the non-linear capacitor is replaced

by a linear one, that has the same change in charge as the non-linear one for the voltage range of interest. A multiplication factor K_{eq} is introduced to relate the linearized capacitor to the value of the junction capacitance under zero bias conditions, $C_{area} = K_{eq} C_{j0}$, where C_{j0} is the junction capacitance per unit area under zero bias condition. The capacitance is given by $C_{bottom-plate} = C_{area} WL_s$ where C_{area} is the equivalent capacitance per unit area. The sidewall capacitance is formed between the source region and the heavily doped stopper implant with level N_A^+ . The sidewall capacitance per unit area will be larger due to the higher doping levels and the side-wall capacitance is given by, $C_{sw} = C_{jsw}(W + 2L_s)$, where C_{jsw} is the side wall capacitance per unit length (x_j is a fixed value for a given technology).

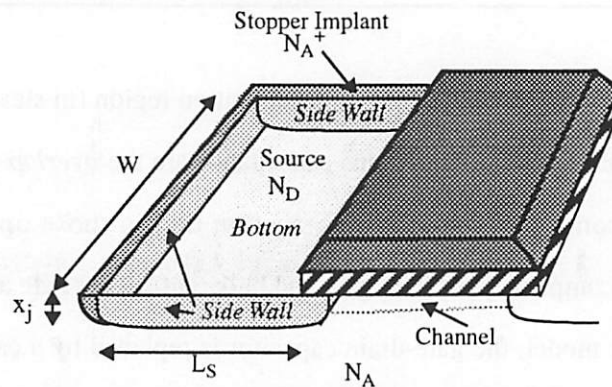


Figure 2-8 : Junction capacitance for a MOS device.

The total junction capacitance is given by:

$$C_{junction} = C_{bottom-plate} + C_{SW} = C_{area} WL_s + C_{jsw} (W+2L_s) \quad (EQ 18)$$

Wiring Capacitance, C_w : C_w is the wiring capacitance to interconnect logic circuits. The interconnect component in general is a function of the placement and routing, which for a chip level design is affected by the number of modules and busses, locality of data communication, etc. There are three main components to interconnect capacitance: area capacitance (parallel plate capacitance component), fringing field component, and wire-wire capacitance [Bakoglu90].

The parallel plate capacitance component can be modeled as shown in Figure 2-9. The parallel plate interconnect capacitance is given by:

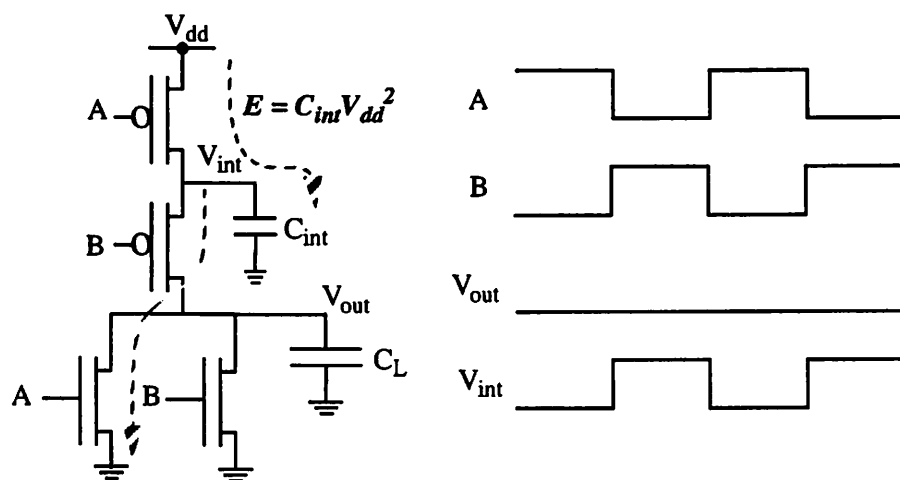


Figure 2-5 : Switching energy analysis must include internal node switching.

Here the input transitions are such that the output always remains in the low state. The internal node V_{int} , however, makes power consuming transitions. Accounting for this, total power dissipation in the circuit can be given by:

$$P_{total} = \left(\sum_{i=1}^{\text{number of nodes}} \alpha_i C_i V_i \right) V_{dd} f \quad (\text{EQ 16})$$

where the number of nodes is the sum of the internal nodes as well as the output nodes of each gate, α is the probability that the node make a power consuming transition drawing energy from the supply, and V_i is the voltage swing of node i which is charged up through the supply. For example, in Figure 2-5, the internal node discharges only to V_t and therefore the energy drawn for the next transition will be $C_{int} (V_{dd} - V_t) V_{dd}$.

For a given technology and gate topology, the product of power and propagation delay is generally a constant. This product is called the power-delay-product (or PDP) and can be considered as a quality measure for a switching device. Power-delay product can be interpreted as the amount of energy expended in each switching event (or transition) and is thus particularly useful in

comparing the power dissipation of various circuit styles. If it is assumed that only the switching component of the power dissipation is important then it is given by:

$$\text{Energy per transition} = P_{\text{total}} / f_{\text{clk}} = C_{\text{effective}} V_{\text{dd}}^2 \quad (\text{EQ 17})$$

where $C_{\text{effective}}$ is the effective capacitance being switched to perform a computation and is given by $C_{\text{effective}} = \alpha C_L$.

2.1.3 Components of Physical Load Capacitance

Figure 2-6 shows the various components of load capacitance for a CMOS inverter driving another identical stage. For the sake of analysis, the various parasitic components (some linear and some non-linear) are lumped into a single output load capacitance. The various components of the physical capacitance are summarized as follows from [Rabaey95].

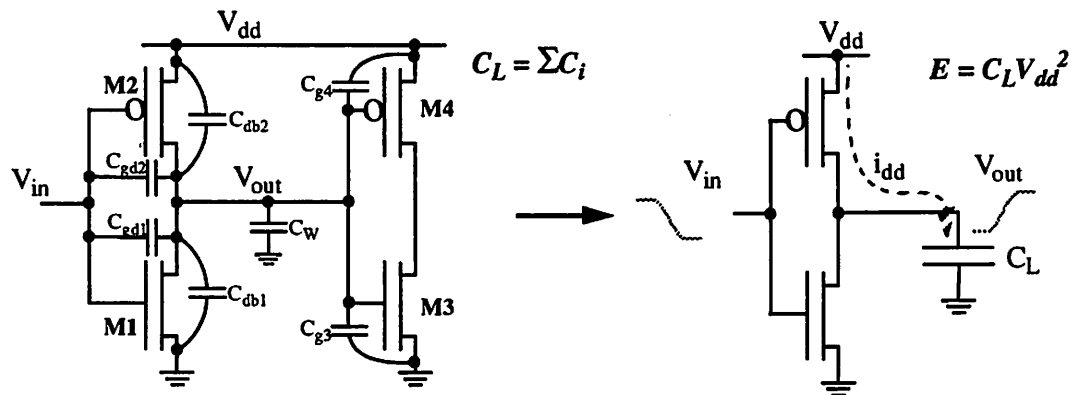


Figure 2-6 : Circuit model for computing parasitic load capacitance of a CMOS gate.

Gate Capacitances, C_{g3} and C_{g4} : The analysis is simplified here by assuming that the total gate capacitance of the loading transistors (M3 and M4) is connected between V_{out} and GND . The overlap and gate capacitances are lumped into a single component $C_g = C_{ox} W L$.

Overlap Capacitance, C_{gd12} : Ideally, the source and drain implants should end at the edge of the gate oxide. However, in reality, the source and drain implants tend to extend below the oxide by an

$$C_{parallel-plate} = \frac{\epsilon_{insulator}}{t_{insulator}} \cdot W \cdot L \quad (EQ 19)$$

If SiO_2 is used as the insulating material then $\epsilon_{ox} = 3.9\epsilon_0$. Typically wires are routed over field oxide which is much thicker than gate oxide hence resulting in a smaller capacitance per unit area. Also, wires on higher layers (such as Metal 2 relative to Metal 1) will have a lower capacitance per unit area.

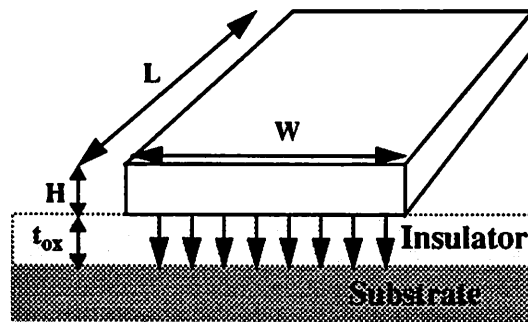


Figure 2-9 : Parallel plate capacitance model of interconnect capacitance.

Clearly, from Equation 19, by increasing the dielectric thickness, the parallel-plate capacitance can be reduced; however, this is not effective if the insulator thickness becomes comparable to the interconnect width and thickness because of the effects of fringing fields [Schaper83]. As shown in Figure 2-10, the capacitance of a single wire can be modeled as a parallel plate capacitor with width equal to $W-H/2$ (the $H/2$ term is to incorporate second order effects [Yuan82]) and a cylindrical wire with a diameter equal to H . An empirical formula for interconnect capacitance has been derived per unit length as:

$$C_{int} = \epsilon_{ox} \left\{ \frac{W}{t_{ox}} - \frac{H}{2t_{ox}} + \frac{2\pi}{\ln \left(1 + \frac{2t_{ox}}{H} \left(1 + \sqrt{1 + \frac{H}{t_{ox}}} \right) \right)} \right\} \quad (EQ 20)$$

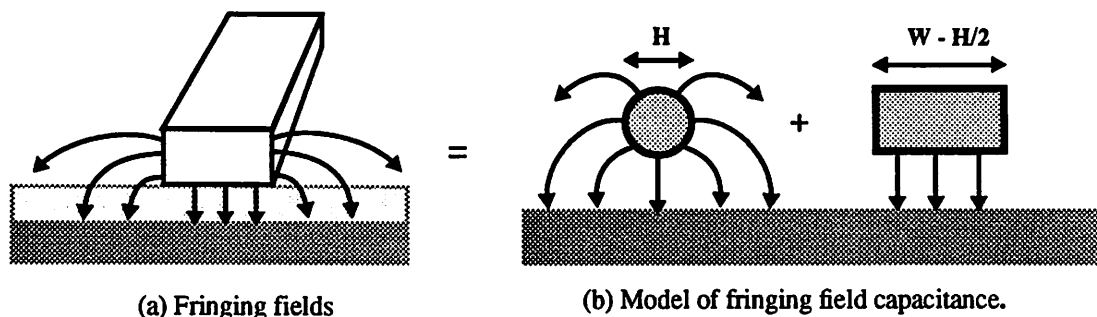


Figure 2-10 : Fringe field contribution to the total interconnect capacitance [Glasser85].

Figure 2-11 shows a plot of the interconnect capacitance which includes fringe effects as a function of W/t_{ox} for two H/t_{ox} ratios [Schaper83]. As seen from this plot, the interconnect capacitance stops decreasing when $W \approx t_{ox} \approx H$ and approaches an asymptotic value of 1pF/cm.

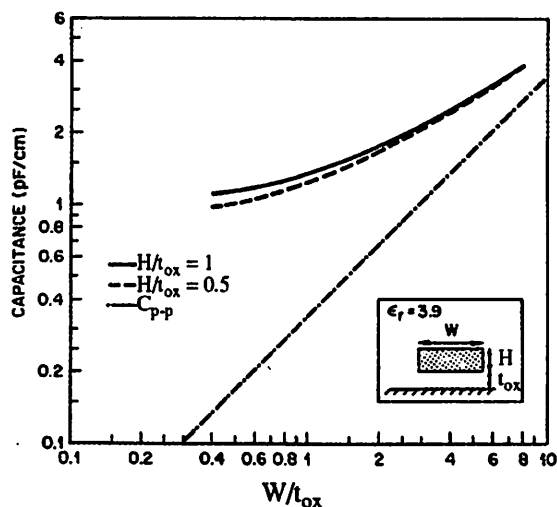


Figure 2-11 : Interconnect capacitance including fringe effect as a function of W/t_{ox} [Schaper83].

The third component of capacitance that is significant at small line widths is the capacitance between neighboring lines. In order to keep the interconnect RC constant of the wires small, the thickness of the wires have not scaled as fast as other dimensions. However, to increase packing density, the distance between wires have scaled. This increases the capacitance between the wires.

Figure 2-12 shows the total capacitance, C_{TOTAL} and its two sub-components - C_{GROUND} , the capacitance to ground which is the parallel-plate and fringe combined, and C_X which is the wire-to-wire capacitance [Schaper83]. The parallel-plate capacitance is also shown for reference. When the wire width (W) is much larger than the insulator thickness (t_{OX}), wire capacitance to ground is larger than the wire-to-wire capacitance. When W is smaller than H , the capacitance between the wires dominate. The minimum is obtained when $W/H=1.75$.

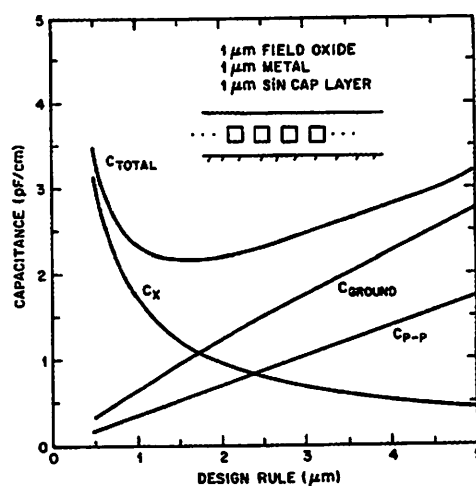


Figure 2-12 : Interconnect capacitance including wire-to-wire capacitance [Schaper83].

Interconnect Capacitance Parameters for a 1.2 μ m CMOS Technology

Table 2-1 shows the area and perimeter capacitances for different interconnect components of a Mosis technology with the feature size $\lambda = 0.6$ (which corresponds to a 1.2 μ m technology).

Table 2-1 : Area and perimeter capacitances for a 1.2 μ m CMOS technology.

Interconnect Capacitance Component	Area Capacitance AF/λ^2	Fringing Capacitance AF/λ
Metal 1 to Substrate	13	35
Metal 2 to Substrate	7	34
Poly to Substrate	23	30
Metal1 to Metal2	24	54

Table 2-1 : Area and perimeter capacitances for a 1.2 μm CMOS technology.

Interconnect Capacitance Component	Area Capacitance AF/λ^2	Fringing Capacitance AF/λ
Metal1 to Poly	25	47
Metal2 to Poly	10	41

Consider a metal 1 wire whose length is 100λ with the minimum size width of 3λ (the area is $300\lambda^2$ and perimeter is $2 * (100 + 3) = 206\lambda$). Table 2-2 shows the capacitance for two different technologies (ignoring any wire to wire capacitance). Also shown in this table is the percentage contribution of perimeter capacitance to the total capacitance. As seen from this table, the fringe component is a significant portion of the total capacitance.

Table 2-2 : Area and perimeter capacitances for Metal 1 wire of length 100λ by 3λ for two different feature sizes.

Technology, λ	Area Capacitance	Perimeter Capacitance	Total Capacitance	$C_P/(C_P + C_A)$
0.6	3.9fF	7.2fF	11.1fF	65%
0.5	2.4fF	4.5fF	6.9fF	65%

As described earlier the wire to wire capacitance can be quite significant and cannot be ignored. For a 1.2 μm CMOS process, the wire to wire capacitance has been determined experimentally from test structures to be $30\text{AF}/\lambda$ for metal1 and $38\text{AF}/\lambda$ for metal2 [Kingsbury94].

Table 2-3 shows the break down between the various components of switched capacitance for several cells from a low-power cell-library. Table 2-4 shows the breakdown of node capacitance switched for datapaths implemented using two different implementation approaches (tiled datapath and standard cells). This data was obtained from modifying a switch-level simulator (IRSIM) to isolate parts of the capacitance. As will be shown in the next chapter, the data about the breakdown between the various physical capacitance components is critical to optimizing

state is 1/4. From Table 2-5 and Equation 22, the ZERO to ONE output transition probability of a 2-input static CMOS NOR gate is given by:

$$\alpha_{0 \rightarrow 1} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} = \frac{3 \cdot (2^2 - 3)}{2^{2 \cdot 2}} = \frac{3}{16} \quad (\text{EQ 23})$$

The state transition diagram annotated with transition probabilities is shown in Figure 2-13. Note that the output probabilities are no longer uniform.

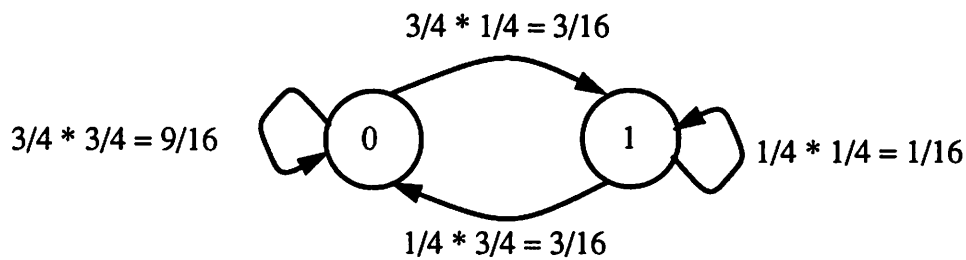


Figure 2-13 : State transition diagram for a 2 input NOR gate.

Table 2-5 shows the truth table for another example gate, a 2-input static XOR gate.

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Table 2-6 : Truth Table of 2 input XOR gate.

The ZERO to ONE output transition probability of a 2-input static CMOS XOR gate is given by:

$$\alpha_{0 \rightarrow 1} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} = \frac{2 \cdot (2^2 - 2)}{2^{2 \cdot 2}} = \frac{1}{4} \quad (\text{EQ 24})$$

Figure 2-14 shows the transition activity factor for NAND, NOR and XOR gates as a function of the number of inputs. For an XOR, the truth table output will have 50% 1's and 50% 0's and therefore assuming a uniform input distribution, the output will always have a transition probability of 1/4, independent of the number of inputs. For a NAND or NOR, the truth table will have only one 1 for the output regardless of the number of inputs. Evaluating Equation 24,

$$\alpha_{0 \rightarrow 1} = \frac{(2^N - 1) \cdot (2^N - (2^N - 1))}{2^{2N}} = \frac{(2^N - 1)}{2^{2N}} \quad (\text{EQ 25})$$

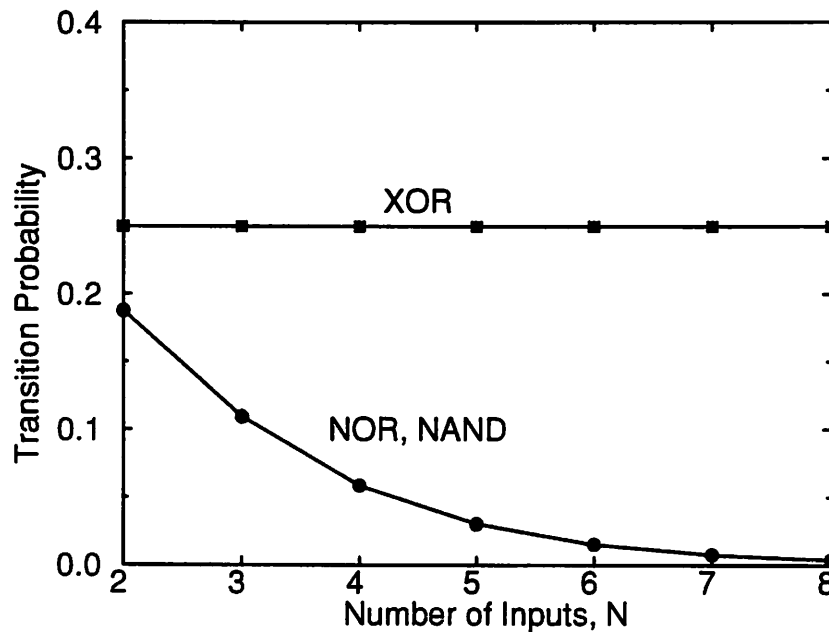


Figure 2-14 : Transition activity for different gates as function of the number of inputs.

Type of Logic Style

For a given function, the type of logic style (static CMOS, dynamic logic, dual rail self-timed implementation) has a strong impact on the node transition activity. In the previous section the transition activity for a static gate was shown to be $P_0 P_1 = P_0 (1 - P_0)$. For dynamic logic, the output transition probability does not depend on the state (history) of the inputs but rather on the

transistor sizes for low-power.

Table 2-3 : Capacitance breakdown at the module level.

MODULE	GATE	DIFFUSION	INTERCONNECT
Adder (Ripple Carry)	30%	45%	25%
Adder (CSA)	37%	31%	32%
TSPC Counter	32%	26%	36%
LOG Shifter (8 bit shift by 4)	15%	42%	43%
Comparator	33%	38%	29%

Table 2-4 : Capacitance breakdown at the datapath level.

MODULE	GATE	DIFFUSION	INTERCONNECT
Adder Chain (7 adders)	38%	38%	24%
Wave Digital Filter	31%	29%	40%
Address Generator (Standard Cell Block)	56%	24%	20%
Video NTSC Sync Genera- tor (Standard Cell Block)	45%	25%	30%

2.1.4 Influence of Logic Level Statistics and Circuit Topologies on Switching Activity, α

There are two components to switching activity: a static component (which does not take into account the timing behavior and is strictly a function of the topology of and the signal statistics) and a dynamic component (which takes into account the timing behavior of the circuit). The switching activity of a design is a complex function of several factors including: type of logic function, logic style, circuit topology, data statistics and the sequencing of operations.

Type of logic function

The amount of transition activity is a strong function of the logic function (NOR vs. XOR vs.

NAND etc.) being implemented. For a static logic design style, the static transition probability (which is computed strictly based on the boolean function and is not a function of the timing skew) assuming independent inputs is the probability that the output will be in the ZERO state in one cycle multiplied by the probability that the output will be in the ONE state in the next clock cycle.

$$\alpha_{0 \rightarrow 1} = P_0 \cdot P_1 = P_0 \cdot (1 - P_0) \quad (\text{EQ 21})$$

where P_0 is the probability that the output will be in the ZERO state and P_1 is the probability that the output will be in the ONE state. Assuming that the inputs are independent and uniformly distributed, any N-input static gate will have a transition probability that corresponds to:

$$\alpha_{0 \rightarrow 1} = \frac{N_0}{2^N} \cdot \frac{N_1}{2^N} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} \quad (\text{EQ 22})$$

where N_0 is the number of ZERO entries in the truth table for the output of the N-input function and N_1 is the number of ONE entries in the truth table for the output of the N-input function.

To illustrate, consider a static 2-input NOR gate whose truth table is shown in Table 2-5. Assume that only one input transition is possible during a clock cycle and also assume that the inputs to the NOR gate have a uniform input distribution of high and low levels.

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Table 2-5 : Truth Table of 2 input NOR gate.

This means that the four possible states for inputs A and B (00, 01, 10, 11) are equally likely. For a NOR gate, the probability that the output is in the ZERO state is $3/4$ and that it will in the ONE

just the signal probabilities. For an N-tree structure (in which the P-network is replaced by a single PMOS precharge transistor) dynamic gate, the output will make a 0 to 1 transition during the precharge phase only if the output was discharged to by the N-tree logic during the evaluate phase. The ZERO to ONE transition probability for an N-tree structure is therefore

$$\alpha_{0 \rightarrow 1} = P_0 \quad (\text{EQ 26})$$

where P_0 is the probability that the output is in the ZERO state. For uniformly distributed inputs, this means that the transition probability is

$$\alpha_{0 \rightarrow 1} = \frac{N_0}{2^N} \quad (\text{EQ 27})$$

where N_0 is once again the number of ZERO entries in the truth table.

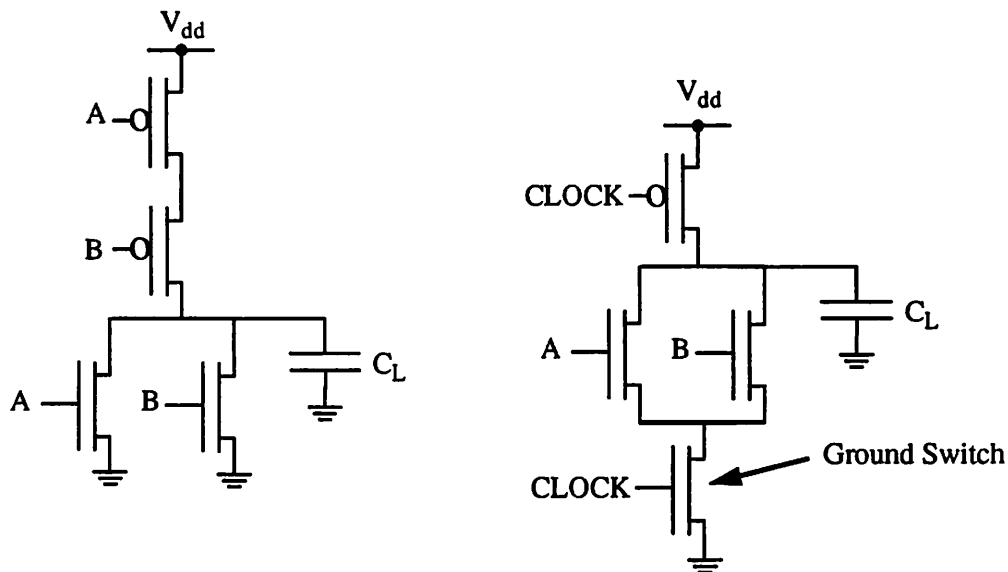


Figure 2-15 : Static NOR vs. N-tree based dynamic NOR.

To illustrate the activity for a dynamic gate, once again consider a 2 input NOR gate. An N-tree dynamic NOR gate is shown in Figure 2-15 along with the static counterpart. For the dynamic implementation, power is consumed during the precharge operation for the times when the output capacitor was discharged the previous cycle. For equi-probable input, there is then a 75%

probability that the output node will discharge immediately after the precharge phase, implying that the activity for such a gate is 0.75 (i.e. $P_{\text{NOR}} = 0.75 C_L V_{\text{dd}}^2 f_{\text{clk}}$). From the truth table shown in Table 2-5 and using Equation 27, the transition probability is given by,

$$\alpha_{0 \rightarrow 1} = \frac{N_0}{2^N} = \frac{3}{4} \quad (\text{EQ 28})$$

The corresponding activity is a lot smaller, 3/16, for a static implementation (as computed in the previous sub-section). Note that for the dynamic case, the activity depends only on the signal probability, while for the static case the transition probability depends on previous state. If the inputs to a static CMOS gate do not change from the previous sample period, then the gate does not switch. This is not true in the case of dynamic logic in which gates can switch. For a dynamic NAND gate, the activity is 1/4 (since there is a 25% the output will be discharged) while it is 3/16 for a static implementation.

Yet another logic style (often used in self-timed circuits) is the dual rail coding logic style Differential Cascode Voltage Switch Logic (DCVSL). A generalized DCVSL gate is shown Figure 2-16 [Jacobs90].

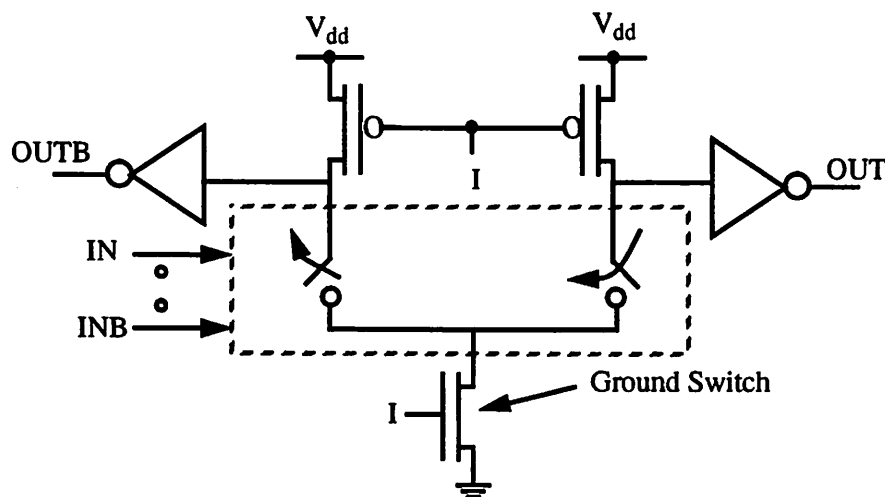


Figure 2-16 : A generalized DCVSL gate.

It is a pre-charged logic family that is very similar to domino logic. The gate also has an NMOS tree which implements the required function but has two complementary outputs. Similar to domino logic, there is a pre-charge phase (I is LOW) when both OUT and OUTB are precharged to LOW. During evaluation only one tree will pull low and therefore only one output will go high. Since one output is guaranteed to go high, there is a guaranteed transition for each input transition; that is, the activity is 1 regardless of the input statistics and this is very power inefficient. That is,

$$\alpha_{0 \rightarrow 1} = 1 \quad (\text{EQ 29})$$

Table 2-7 shows the summary of activity factor for a 2-input NOR gate implemented using different logic styles for independent and uniformly distributed inputs.

Table 2-7 : Activity comparison of different logic styles.

Logic Style	$\alpha_{0 \rightarrow 1}$
Static CMOS	3/16
Dynamic CMOS	3/4
DCVSL	1

Signal Statistics

Signal statistics have a very strong influence on power consumption. In the previous sections, the NOR gate was analyzed assuming random inputs. However, signals in a circuit are typically not random and this attribute can be exploited to reduce the power consumption. For a CMOS logic gate, it was shown earlier in Equation 21 that the probability of transition is $P_0 P_1$. The transition probabilities were derived assuming that the inputs were bit-wise uncorrelated and that they were equi-probable. In the analysis to follow, the inputs will still be assumed to be uncorrelated, but the equi-probable assumption will not be assumed. To illustrate the influence of signal statistics on power consumption, consider once again a 2 input static NOR gate, and let P_a and P_b be the probabilities that the inputs A and B are ONE. In this case the probability that the output node is a ONE is given by:

$$P_1 = (1 - P_a)(1 - P_b) \quad (\text{EQ 30})$$

Therefore, the probability of a transition from 0 to 1 is:

$$\alpha_{0 \rightarrow 1} = P_0 P_1 = (1 - (1 - P_a)(1 - P_b))(1 - P_a)(1 - P_b) \quad (\text{EQ 31})$$

Figure 2-17 show the transition probability as a function of P_a and P_b . From this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

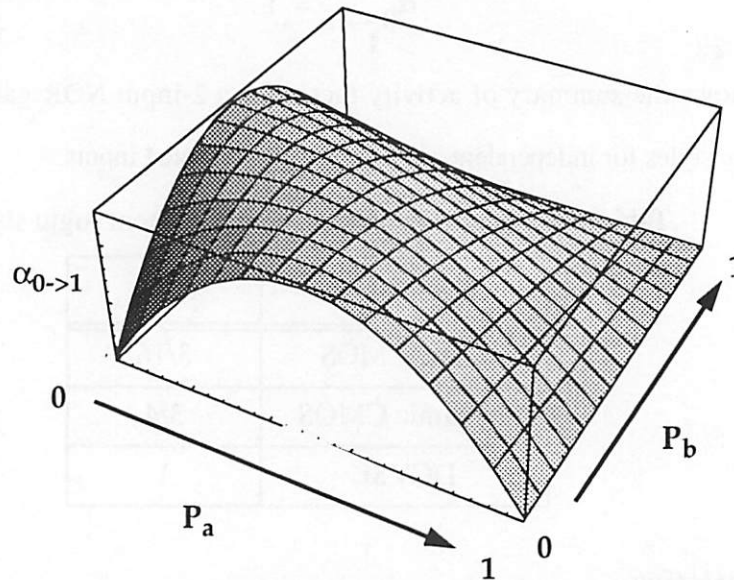


Figure 2-17 : Transition probability for a 2 input NOR as a function of input statistics.

Table 2-8 shows the transition probabilities for other logic gates as a function of the input signal probabilities.

Function	$P_{0 \rightarrow 1}$
AND	$(1 - P_A P_B) P_A P_B$
OR	$(1 - P_A)(1 - P_B)(1 - (1 - P_A)(1 - P_B))$
XOR	$(1 - (P_A + P_B - 2P_A P_B))(P_A + P_B - 2P_A P_B)$

Table 2-8 : Output transition probabilities for various static logic gates.

Figure 2-17 shows the transition activity for a 2-input XOR gate as a function on the input probabilities.

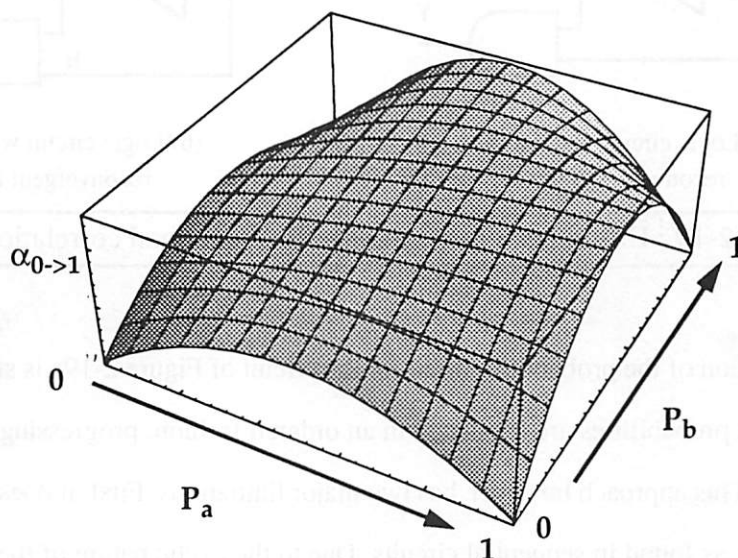


Figure 2-18 : Transition probability for a 2 input XOR gate.

Inter-signal Correlations

In the previous sections, the transition probability of a gate was computed assuming that the inputs are independent. The evaluation of the switching activity becomes more evolved, when complex logic networks have to be analyzed since when signals propagate through a number of logic layers, they often become correlated or 'colored'. The logic shown in Figure 2-19 will be used to illustrate the effect of signal correlations on switching activity. First consider the circuit shown in Figure 2-19a and assume that the primary inputs, A and B, are uncorrelated and are uniformly distributed. Node C will have a signal probability of 1/2 and a 0->1 transition probability of 1/4. The probability that the node Z undergoes a power consuming transition is determined from the expression shown in Table 2-8.

$$\alpha_{0 \rightarrow 1} = (1 - P_a P_b) P_a P_b = (1 - 1/2 \cdot 1/2) 1/2 \cdot 1/2 = 3/16 \quad (\text{EQ 32})$$

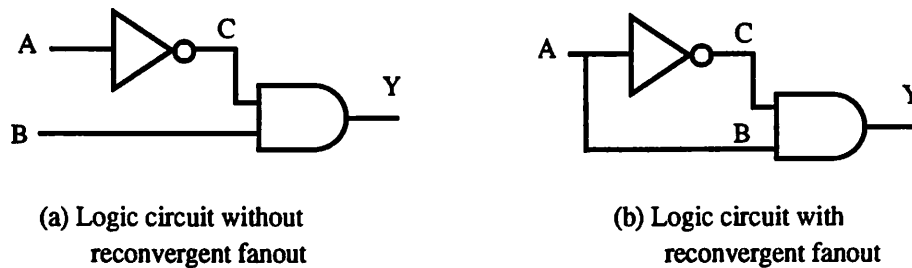


Figure 2-19 : Examples illustrating the effect of signal correlations.

The computation of the probabilities used for the circuit of Figure 2-19a is straightforward: Signal and transition probabilities are evaluated in an ordered fashion, progressing from the input to the output node. This approach however, has two major limitations. First, it does not deal with circuits with feedback as found in sequential circuits. Due to the cyclic nature of these circuits, finding an input to output traversal order is not possible. Secondly, it assumes that the signal probabilities at the input of a gate are independent. This is rarely the case in actual circuits, where reconvergent fanout often causes inter-signal dependencies.

Consider, for example, the logic network of Figure 2-19b. The inputs to the AND gate (C and B) are interdependent, as B is also a function of A (they are identical). It is easy to see that the approach to compute probabilities presented previously fails under these circumstances. Traversing from inputs to outputs yields a transition probability of $3/16$ for node Z (similar to the previous example). This value for transition probability is clearly false, as a simple logic minimization shows that the complete network can be reduced to $Z = C \cdot B = A \cdot \bar{A} = \text{logical zero}$.

To get the precise results in the progressive analysis approach, it is essential to take signal interdependencies into account. This can be accomplished with the aid of conditional probabilities. This will be explained with the aid of this simple example. For an AND gate, Z equals 1 if and only if B and C are equal to 1.

$$P_Z = P(Z=1) = P(B=1, C=1) \quad (\text{EQ 33})$$

where $P(B=1,C=1)$ represents the probability that B and C are equal to 1 simultaneously. If B and C are independent, as with the circuit shown in Figure 2-19a, $P(B=1,C=1)$ can be decomposed into $P(B=1) \cdot P(C=1)$, and this yields the expression for the AND-gate, derived earlier: $P_Z = P(B=1) \cdot P(C=1) = P_B P_C$. If a dependency between the two exists (as is the case in Figure 2-19b), a conditional probability has to be employed, such as

$$P_Z = P(C=1|B=1) \cdot P(B=1) \quad (\text{EQ 34})$$

The first factor in Equation 34 represents the probability that $C=1$ given that $B=1$. The extra condition is necessary as C is dependent upon B. Inspection of the network shows that this probability is equal to 0, since C and B are logical inversions of each other, resulting in the signal probability for Z, $P_Z = 0$.

Deriving those expressions in a structured way for large networks with reconvergent fanout is complex, especially when the networks are also sequential and contain feedback loops and therefore computer support is essential. The goal is to estimate or analyze the transition probabilities at the nodes of a network. To be meaningful, the analysis program has to take in a typical sequence of input signals, as the power dissipation will be a strong function of statistics of those signals.

Circuit Topology

The manner in which logic gates are interconnected can have a strong influence on the overall switching activity. So far, the emphasis has been on the static component of activity which does not take into account the timing behavior and is strictly a function of the topology and the signal statistics. Another important component of switching power is the dynamic component which takes into account the timing behavior of the circuit.

A node in circuit implemented using static logic can have multiple transitions inside a clock cycle before settling to the correct logical value. The number of extra transition or glitching transition is

a function of the logic depth, the signal skew due to different arrival times of the input, and the signal patterns. In the worst case, the number of “extra” transitions (also called glitching transitions) can grow as $O(N^2)$, where N is the logic depth.

Figure 2-20 shows a generic logic circuit with a depth of $N+1$, for which the worst case transition activity will be computed. Assume that the primary inputs A_0-A_N , B_0-B_N , and C_0 all arrive at the same time (this is a reasonable assumption if they are all output from registers and the wiring skew is not significant). Assume that each level of logic takes a fixed unit time to evaluate. Let arrival time be defined as the time at which a signal arrives at a logic gate. The first level of logic (level 0) will make at most one transition based on C_0 , A_0 , and B_0 and therefore C_1 will have an arrival time at the level 1 logic block at time $T=1$. The second level of logic (level 1) will first evaluate based on C_1 at time $T=0$ and the value of the new inputs A_1 and B_1 (whose arrival time is at $T=0$). This will cause C_2 to have a transition at $T=1$. When C_1 arrives at time $T=1$, the level 1 logic will re-evaluate with the same A and B inputs (which are stable from time $T=0$ onwards) and will cause another transition at C_2 . That is, node C_2 will have 2 new arrival times in a clock period for a worst case input pattern. This can cause three transitions at node C_3 since the logic block of level 3 will evaluate with inputs A_3 and B_3 at time $T=0$, and with three values of C_3 at times $T=0$, $T=1$, and $T=2$.

It is easy to see that for the N^{th} logic stage, the output will transition N times in the worst case, while only one transition is useful. The number of extra transitions for logic level 0 is 0, the extra transitions for level 1 is 1, level 2 is 2, and for level N is N . The total number of extra transitions is $1+2+3+\dots+N-1+N = N(N+1)/2$. That is the number of extra glitching transitions in the worst case grows as $O(N^2)$. In reality, the transition activity due to glitching will be a lot less since the worst input pattern will occur very infrequently and the sequence of input data patterns will determine the total amount of glitching.

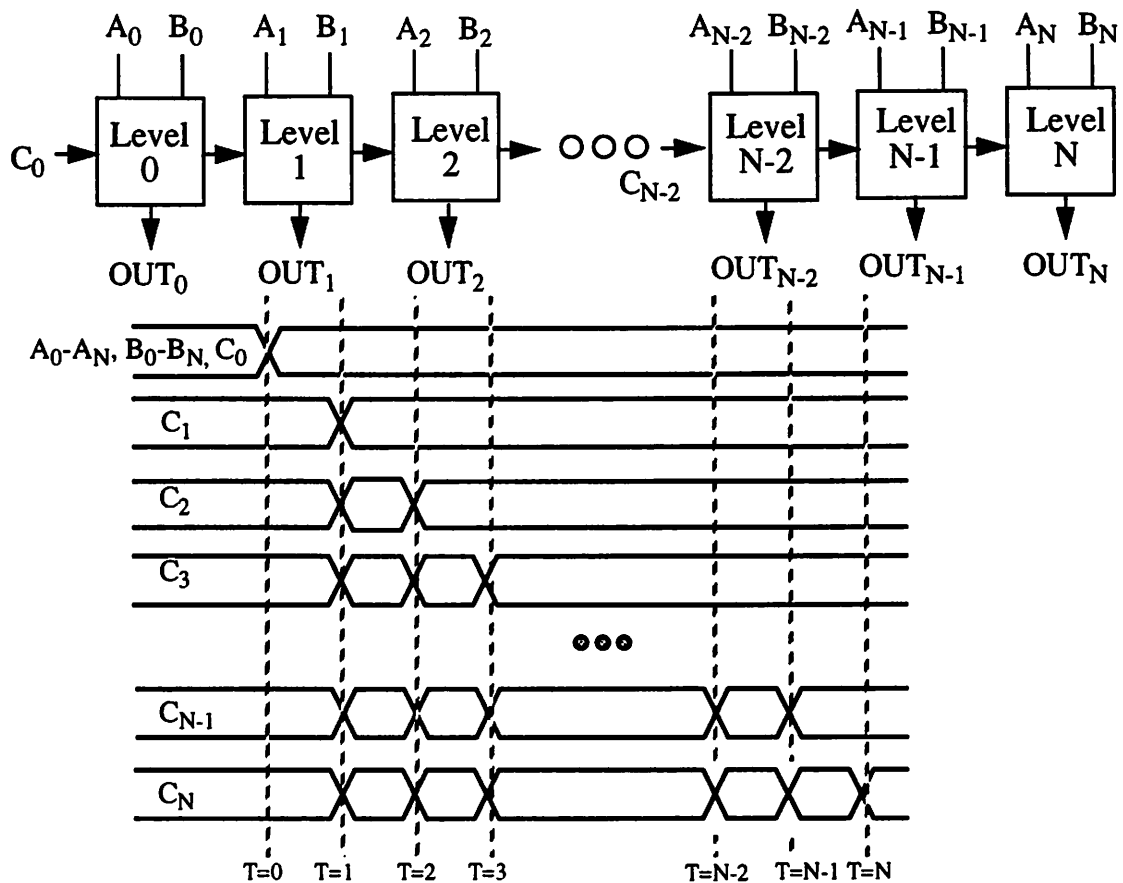


Figure 2-20 : Dynamic component of switching power.

In making power trade-offs between logic structures, it is important to consider both the static and dynamic transition activities simultaneously. A circuit can have a lower static transition probability while having a larger dynamic transition probability relative to another circuit. To illustrate this point consider two alternate implementations of $F = A \cdot B \cdot C \cdot D$ as shown in Figure 2-21.

First consider the static behavior assuming that all primary inputs (A,B,C,D) are uncorrelated and random (i.e $P_{1,(a,b,c,d)} = 0.5$). For an AND gate, the probability that the output is in the 1 state is given by:

$$P_1 = P_a P_b \quad (\text{EQ 35})$$

Therefore the probability that the output will make a 0 to 1 transition is given by:

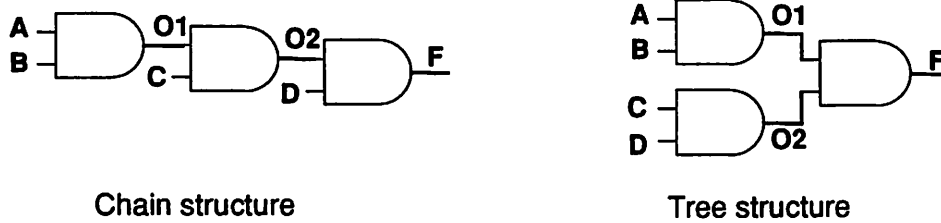


Figure 2-21 : Simple example to demonstrate the influence of circuit topology on activity.

$$P_{0 \rightarrow 1} = P_0 P_1 = P_0 (1 - P_0) = (1 - P_a P_b) P_a P_b \quad (\text{EQ 36})$$

Given this, the signal and transition probabilities can be computed for the two topologies shown in Figure 2-21 which is summarized in Table 2-9.

Table 2-9 : Probabilities for tree and chain topologies

	O1	O2	F
P_1 (chain)	1/4	1/8	1/16
$P_0 = 1 - P_1$ (chain)	3/4	7/8	15/16
$P_{0 \rightarrow 1}$ (chain)	3/16	7/64	15/256
P_1 (tree)	1/4	1/4	1/16
$P_0 = 1 - P_1$ (tree)	3/4	3/4	15/16
$P_{0 \rightarrow 1}$ (tree)	3/16	3/16	15/256

The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However, as mentioned before, looking strictly at the static behavior of the circuit is not adequate, and it is also important to consider the timing behavior to accurately make power trade-offs of various topologies. Timing skew between signals can cause spurious transitions resulting in extra power consumption. To illustrate this trade-off, consider once again the two topologies in Figure 2-21. For the chain implementation with an input transition of 1110 \rightarrow 1011 for ABCD, assuming a unit delay for each gate, ignoring dynamic switching effects, the output node will not make a transition. However, due to timing skew through

the logic, the output will make an “extra” transition; i.e O2 will only be valid 2 units after the inputs arrive, causing the output AND gate to evaluate with the new input D and the previous value of O2 (Figure 2-22). The tree implementation, on the other hand, is balanced and is glitch free. This example demonstrates that a circuit topology can have a smaller static component of activity while having a higher dynamic component.

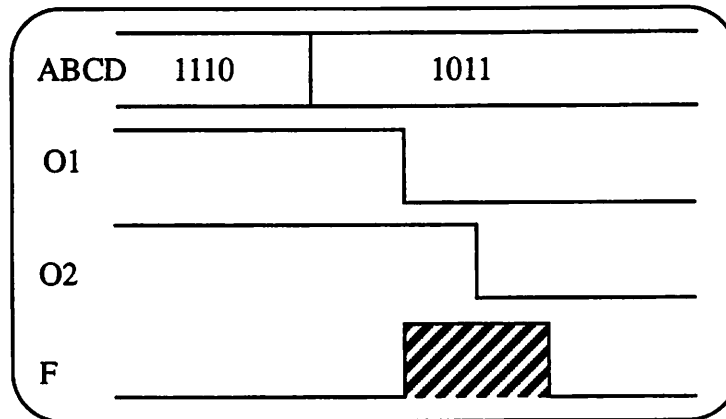


Figure 2-22 : Circuit imbalances result in spurious transitions.

The actual waveforms illustrating the glitching behavior in static CMOS circuits is shown in Figure 2-23, which is the SPICE simulation of a static 16-bit adder, with all bits of IN0 and the CIN of the LSB going from “zero” to “one”, with all the bits of IN1 set to “zero”. For all bits, the resultant sum should be zero; however, the propagation of the carry signal causes a “one” to appear briefly at most of the outputs. These spurious transitions dissipate extra power over that strictly required to perform the computation. Note that some of the bits only have partial glitching. This is due to the inertial delay of the logic gate - that the timing skew between the inputs must be greater than a certain value to for the glitch propagate through the gate. The number of these extra transitions is a function of input patterns, internal state assignment in the logic design, delay skew, and logic depth. Though it is possible with careful logic design to eliminate these transitions (for example using balanced paths as described earlier), a major advantage of dynamic logic is that it intrinsically does not have this problem, since any node can undergo at most one power-

consuming transition per clock cycle.

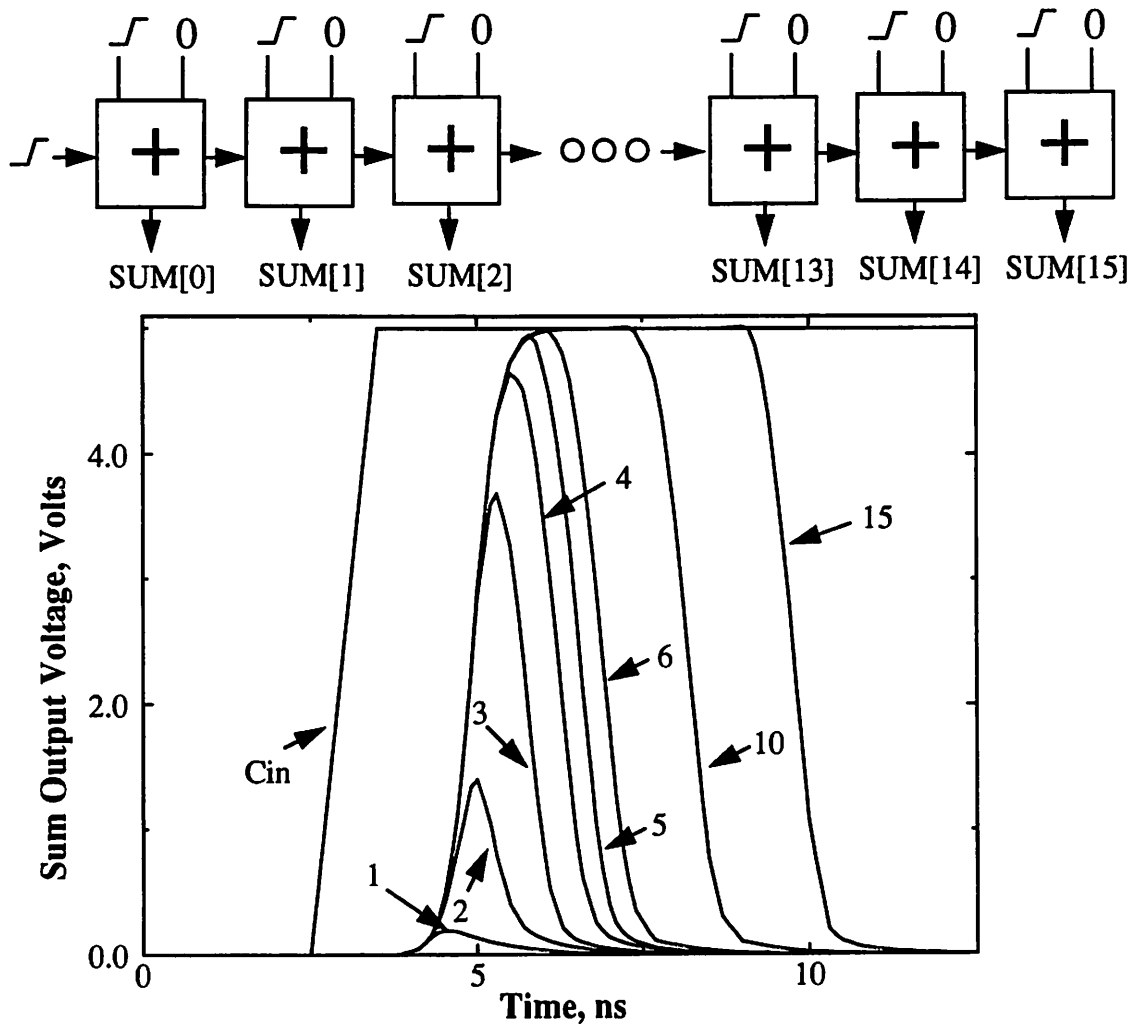


Figure 2-23 : Waveforms for a 16-bit adder demonstrating glitching behavior.

2.1.5 Word Level Signal Statistics Influencing Activity, α

A very important attribute of signal processing applications which can be used in minimizing the switched capacitance, is the correlation which can exist between values of a temporal sequence of data, since switching should decrease if the data is slowly changing (highly correlated). This is in contrast to general purpose computation, where the data being processed tends to be random. To illustrate the correlation in signal processing applications, consider the word level transition

characteristics of two common DSP signals: human speech and video (chrominance component).

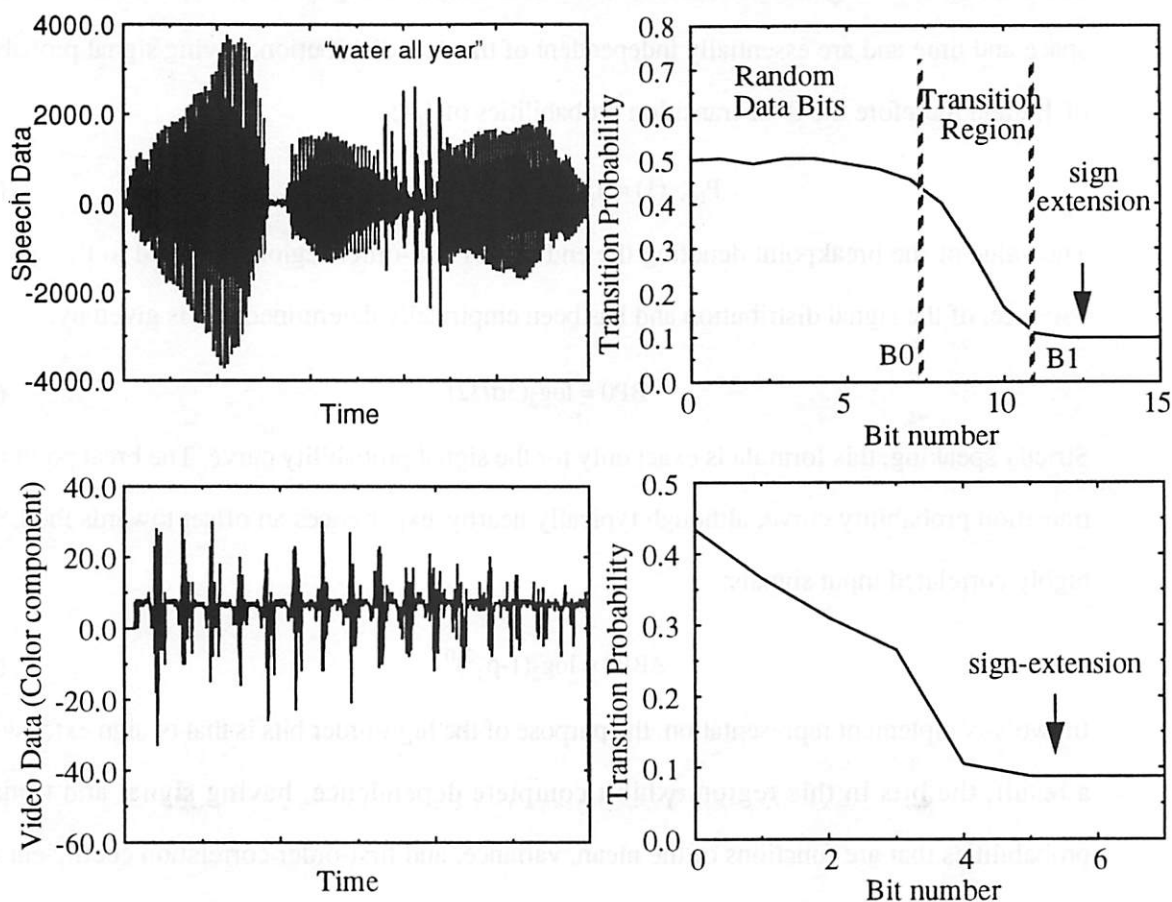


Figure 2-24 : Data in signal processing applications is often correlated.

It has been shown that there exists a direct relationship between bit level probabilities and word level statistics [Landman93]. This correlation is illustrated in Figure 2-24 for the speech and video input signals. For each input, the right hand portion of the figure shows both signal and transition probabilities for each bit of the input data word (two's-complement assumed). Clearly, these signals follow a similar pattern which can be exploited to yield a simple piecewise linear model with breakpoints BP0 and BP1. The important features of this model - the values of the breakpoints and the signal and transition probabilities - can all be extracted from three statistical parameters: the mean, μ ; the variance, σ^2 ; and the lag one correlation coefficient, $\rho_1 = \text{cov}(X_t, X_{t+1})/\sigma^2$.

In the lower-order region from the LSB to BP0, as might be expected, the bits are uncorrelated in space and time and are essentially independent of the data distribution, having signal probabilities of 1/2 and therefore the 0->1 transition probabilities of 1/4:

$$P_{\text{lsb}'s}(1) = 1/2, P_{\text{lsb}'s}(0 \rightarrow 1) = 1/4 \quad (\text{EQ 37})$$

The value of the breakpoint denoting the end of this low-order region is related to the spread, or variance, of the signal distribution and has been empirically determined and is given by:

$$\text{BP0} = \log_2(3\sigma/32) \quad (\text{EQ 38})$$

Strictly speaking, this formula is exact only for the signal probability curve. The breakpoint for the transition probability curve, although typically nearby, experiences an offset towards the LSB for highly correlated input signals:

$$\Delta\text{BP0} = \log_2(1-\rho_1^2)^{0.5} \quad (\text{EQ 39})$$

In two's-complement representation, the purpose of the high-order bits is that of sign extension. As a result, the bits in this region exhibit complete dependence, having signal and transition probabilities that are functions of the mean, variance, and first-order correlation coefficient of the data word:

$$P_{\text{msb}'s}(1) = P(-) = F_1(\mu/\sigma) \quad (\text{EQ 40})$$

$$P_{\text{msb}'s}(0 \rightarrow 1) = P(+ \rightarrow -) = F_{01}(\mu/\sigma, \rho_1) \quad (\text{EQ 41})$$

The exact probabilities depend, of course, on the distribution of the signal; however, noting that many typical DSP inputs are closely approximated by Gaussian processes, the univariate and bivariate normal distribution functions can be substituted for F_1 and F_{01} , respectively. The breakpoint for the sign extension region is determined by the maximum extent of the signal distribution into either positive or negative values and is given specifically by:

$$\text{BP1} = \log_2(|\mu|+3\sigma) \quad (\text{EQ 42})$$

In the middle region between BP0 and BP1, the correlation of the bits falls between the extremes represented by the lower and higher-order regions and as a result a linear approximation for the

probabilities in this transition region models the situation well.

These signal correlations can be exploited to minimize the bit transition activity through data coding on busses, optimized representation for arithmetic, optimized time-multiplexing, ordering of operations, and activity driven placement (these techniques will be described in Chapter 4).

2.1.6 Influence of Voltage Scaling

Since power consumption in CMOS circuits is proportional to the square of the supply voltage (assuming that the dynamic component of power consumption dominates), it is clear voltage reduction will have a significant impact on power; indeed, reducing the supply voltage is the key to low-power operation, even after taking into account the modifications to the system architecture which is required to maintain the computational throughput. A review of circuit behavior (delay and energy characteristics) as a function of scaling supply voltage and feature sizes will be presented. By comparison with experimental data, it is found that simple first order theory yields an amazingly accurate representation of the various dependencies over a wide variety of circuit styles and architectures.

Impact on Delay and Power-Delay Product

As noted in Equation 3, the energy per transition or equivalently the power-delay product is proportional to V^2 . This is seen from Figure 2-25, which is a plot of two experimental circuits which exhibit the expected V^2 dependence. Therefore, it is only necessary to reduce the supply voltage for a *quadratic* improvement in the power-delay product of a logic family.

Unfortunately, this simple solution to low power design comes at a cost. As shown in Figure 2-26, the effect of reducing V_{dd} on circuit delay is shown for a variety of different logic circuits, that range in size from 56 to 44,000 transistors spanning a wide variety of functions, all exhibiting essentially the same dependence (see Table 2-10). Clearly, we pay a speed penalty for a V_{dd} reduction, with the delays drastically increasing as V_{dd} approaches the sum of the threshold

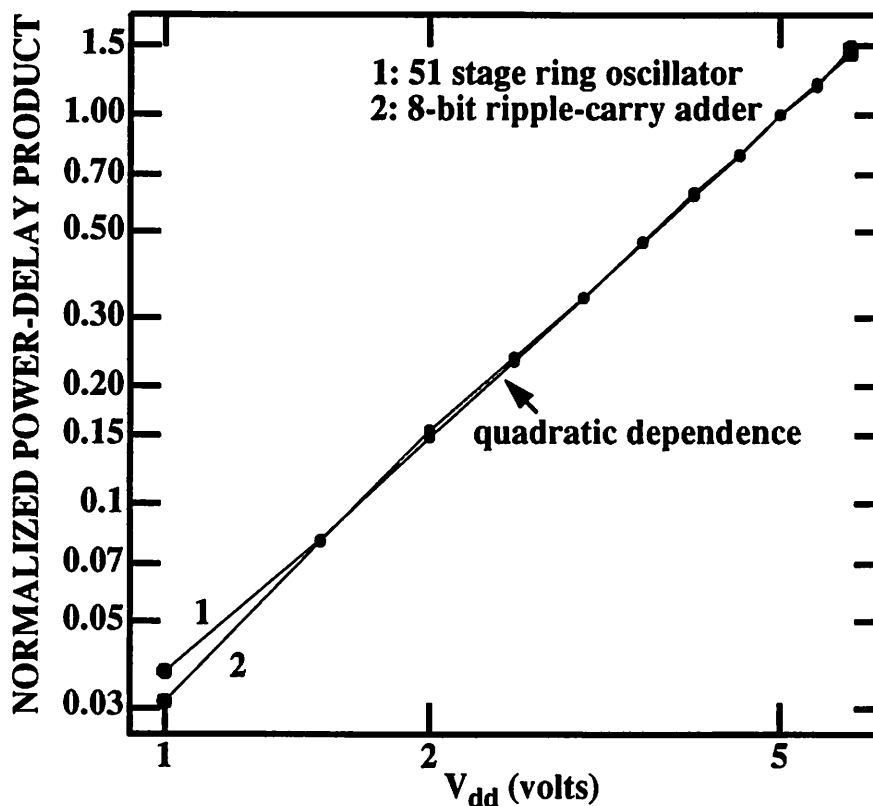


Figure 2-25 : Power-delay product exhibiting square law dependence for two different circuits.

voltages of the devices. Even though the exact analysis of delay is quite complex if the non-linear characteristic of a CMOS gate are taken into account, it is found that a simple first-order derivation adequately predicts the experimentally determined dependence and is given by:

$$T_d = \frac{C_L \times V_{dd}}{I} = \frac{C_L \times V_{dd}}{\mu C_{ox} (W/L) (V_{dd} - V_t)^2} \quad (\text{EQ 43})$$

We also evaluated (through experimental measurements and SPICE simulations) the energy and delay performance for several different logic styles and topologies using an 8-bit adder as a reference; the results are shown on a log-log plot in Figure 2-27. Table 2-10 describes the circuits used for analysis in Figure 2-27. We see that the power-delay product (or the energy per transition) improves as delays increase (through reduction of the supply voltage), and therefore a CMOS

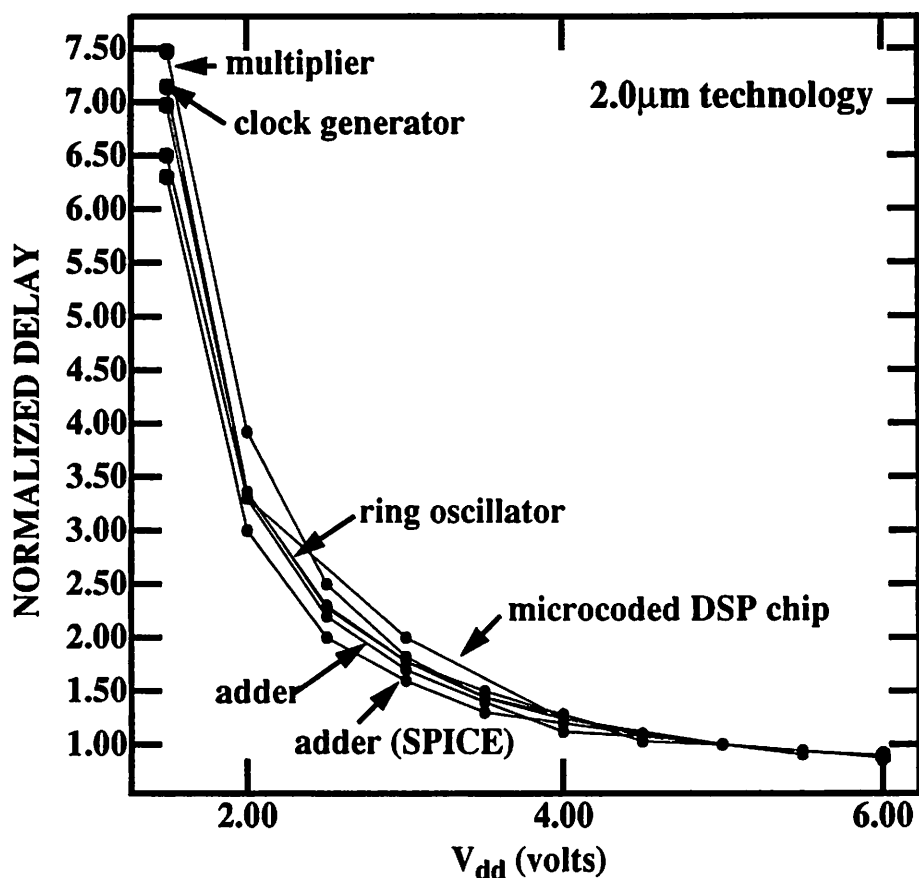


Figure 2-26 : Data demonstrating delay characteristics follow simple first order theory.

circuit is most effective when it operates at the *slowest* possible speed. Since the objective is to reduce power consumption while maintaining the overall system throughput, compensation for these increased delays at low voltages is required. Of particular interest in this figure is the range of energies required for a transition at a given amount of delay. The best logic family analyzed (over 10 times better than the worst that was investigated) was the pass gate family, CPL if a reduced value for the threshold is assumed [Yano90].

Component (all in 2 μ m)	# of transistors	Area	Comments
Microcoded DSP Chip	44802	94mm ²	20-bit datapath
Multiplier	20432	12.2mm ²	24x24 bits
Adder	256	0.083mm ²	conventional static
Ring Oscillator	102	0.055mm ²	51-stages
Clock Generator	56	0.04mm ²	cross-coupled NOR

Table 2-10 : Details of components used for the study in Figure 2-26.

Figures 2-25, 2-26, and 2-27 suggest that the delay and energy behavior as a function of V_{dd} scaling for a given technology is “well-behaved” and relatively independent of logic style and circuit complexity. We will use this result during our optimization of architecture for low-power by treating V_{dd} as a free variable and by allowing the architectures to vary to retain constant throughput. By exploiting the monotonic dependencies of delay and energy versus supply voltage that hold over wide circuit variations, it is possible to make relatively strong predictions about the types of architectures that are best for low power design. Of course, as mentioned previously, there are some logic styles such as NMOS pass-transistor logic without reduced thresholds whose delay and energy characteristics would deviate from the ones presented above, but even for these cases, though the quantitative results will be different, the basic conclusions will still hold. The next chapter will describe the architecture driven voltage scaling strategy that allows voltage reduction while keeping throughput constant.

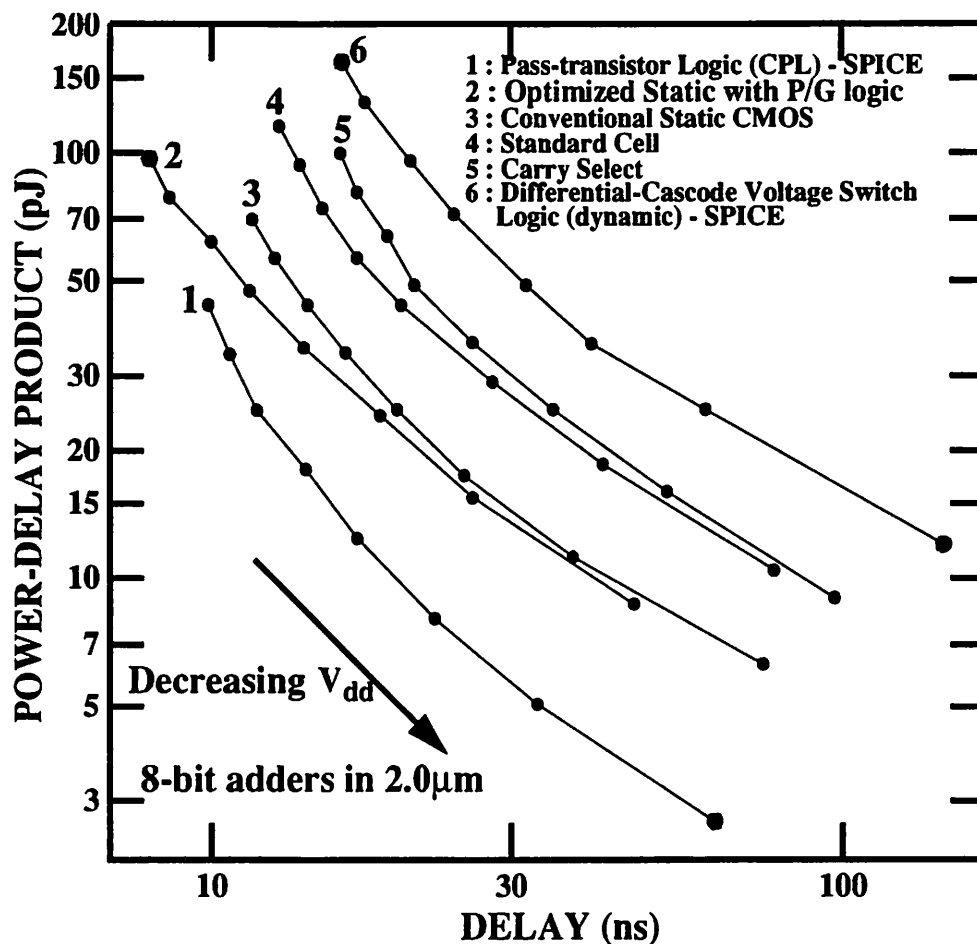


Figure 2-27 : Data showing improvement in power-delay product at the cost of speed for various circuit approaches.

Table 2-11 : Description of circuits used for study in Figure 2-27.

Adder Number in Figure 2-27	Short Description of Each Adder
1	Differential pass transistor logic made of single-transistor NMOS pass-gates (as opposed to NMOS and PMOS connected in parallel). The circuit is described in [Yano90] and in Chapter 4. A reduced threshold voltage is assumed for the pass transistors.

Table 2-11 : Description of circuits used for study in Figure 2-27.

Adder Number in Figure 2-27	Short Description of Each Adder
2	Optimized static logic with propagate and generate circuitry. The sum uses complementary pass-transistor circuits while the carry logic is implemented using conventional CMOS.
3	A conventional CMOS implementation of an adder without propagate/generate logic [Weste88].
4	A standard cell implementation of a conventional adder. The devices are significantly bigger and the interconnect capacitance is larger since the placement is not as structured as a tiled datapath approach [Brodersen88].
5	A carry select implementation realized using conventional CMOS gates [Brodersen88].
6	A DCVSL adder (Differential Cascode Voltage Swing Logic) is a precharged implementation that uses dual-rail coding. This circuit is typically used in self-timed circuits [Jacobs90]. Due to the differential nature of this topology, there is a guaranteed transition on every bit for each access resulting in high switching activity.

2.2 Short-circuit Component of Power

The previous section analyzed the switching component of power consumption which corresponds to the amount of energy required to charge parasitic capacitors. The switching component of power is independent of the rise and fall times at the input of logic gates. Finite rise and fall times of the input waveforms however result in a direct current path between V_{dd} and GND which exist for a short period of time during switching. Specifically, when $V_{Tn} < V_{in} < V_{dd} - |V_{Tp}|$ holds for the input voltage, there will be a conductive path open between V_{dd} and GND because both the NMOS and PMOS devices are ON. Such a path never exists in dynamic circuits, as precharge and evaluate transistors should never be on simultaneously as this would lead to incorrect evaluation. Short-Circuit currents are, therefore, a problem solely encountered in

static designs. Figure 2-28 shows the behavior of an inverter assuming no output load. On a low-to-high transition at the input, the NMOS will start to conduct when V_{in} is equal to V_{tn} , and the PMOS will stop conducting when V_{in} is equal to V_{tp} . Under zero capacitive load conditions, all the current that is drawn from the supply goes to short-circuit power.

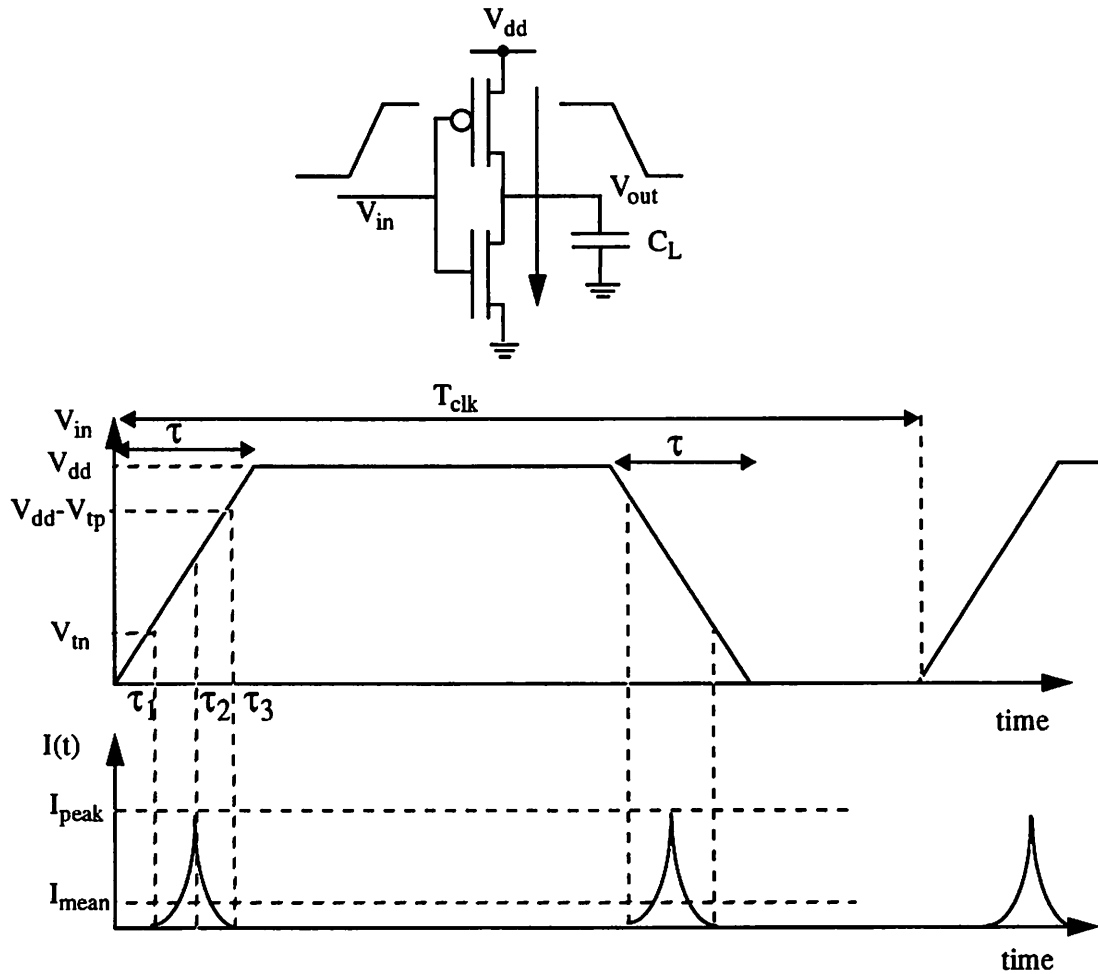


Figure 2-28 : Current behavior with no output load [Veendrick84].

Let τ be the rise time of the input signal, and let $V_t = V_{tn} = V_{tp}$. Also, the effective transistor strengths be equal for the NMOS and PMOS; let $\beta = \beta_n W_n = \beta_p W_p$. From τ_1 to τ_2 , the NMOS device is in saturation since the output voltage will be greater than $V_{in} - V_t$. Then the current for the NMOS is given by:

$$I = \frac{\beta}{2} (V_{in} - V_t)^2 \quad \text{for } 0 < I < I_{max} \quad (\text{EQ 44})$$

The current will reach a maximum, I_{max} at $V_{in} = V_{dd}/2$ due to the symmetric assumption. The mean current can be computed [Veendrick84] to determine the average short-circuit current that is drawn from the supply.

$$I_{mean} = 2 \cdot \frac{2}{T_{clk}} \cdot \int_{\tau_1}^{\tau_2} I(t) dt = \frac{4}{T_{clk}} \cdot \int_{\tau_1}^{\tau_2} \frac{\beta}{2} (V_{in}(t) - V_t)^2 dt \quad (\text{EQ 45})$$

Here, $V_{in}(t)$ is given by:

$$V_{in}(t) = \frac{V_{dd}}{\tau} \cdot t \quad (\text{EQ 46})$$

And therefore, τ_1 to τ_2 are given by:

$$\tau_1 = \frac{V_t}{V_{dd}} \cdot \tau \quad (\text{EQ 47})$$

$$\tau_2 = \frac{\tau}{2} \quad (\text{EQ 48})$$

Plugging into Equation 45,

$$I_{mean} = \frac{2\beta}{T_{clk}} \int_{\tau/2}^{V_t \cdot \tau / V_{dd}} \left(\frac{V_{dd} \cdot t}{\tau} - V_t \right)^2 dt \quad (\text{EQ 49})$$

which results in:

$$I_{mean} = \frac{\beta}{12 \cdot V_{dd}} (V_{dd} - 2V_t)^3 \frac{\tau}{T_{clk}} \quad (\text{EQ 50})$$

The short-circuit power is then given by:

$$P_{short-circuit} = V_{dd} \cdot I_{avg} = \frac{\beta}{12} (V_{dd} - 2V_t)^3 \frac{\tau}{T_{clk}} \quad (\text{EQ 51})$$

In the case with no capacitive load, the short-circuit current is directly proportional to the risetime and the effective transistor strength, β .

Short circuit currents are significant when the rise/fall time at the input of a gate is much longer than the output rise/fall time. This is because the short-circuit path will be active for a longer period of time. Consider a static CMOS inverter with a low to high input transition. Assume first that the load capacitance is very large, such that the output fall time is significantly larger than the input rise times. Here, the input will go through the transient region before the output starts to change. Since the source-drain voltage of the PMOS device is essentially 0 during that period, the device shuts off without ever delivering any current. The short-circuit current is therefore essentially zero. If the output capacitance is very small and the output fall time is substantially smaller than the input rise time, the drain-source voltage of the PMOS device is equal to V_{dd} during most of the transition period, resulting in large short-circuit current (equal to the saturation current of the PMOS) during most of the transient period. Figure 2-29 shows a graph of the short-circuit current as a function of the output capacitance for a fixed input rise time [Veendrick84].

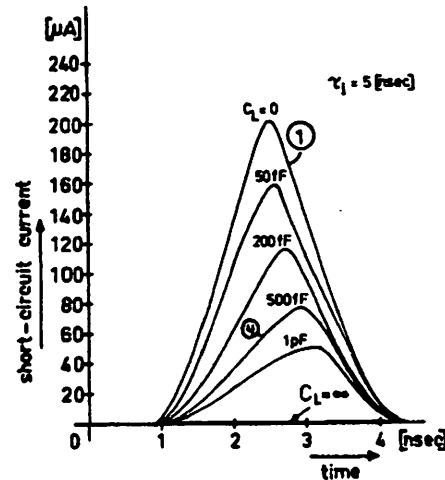


Figure 2-29 : Short-circuit current as a function of load capacitance [Veendrick84].

This above analysis implies that short-circuit dissipation is minimized by making the output rise/fall time larger than the input rise/fall times.

On the other hand, making the output rise/fall times too large slows down the circuit and might cause short-circuit current in the fanout-gates. Therefore a good compromise to minimize the total average short-circuit current, it is desirable to have equal input and output edge times [Veendrick84].

Figure 2-30 plots the fraction of energy consumed by short-circuit current versus the ratio of the input rise time to the output rise time. The ΔE increases with increasing input edge time. In this case, the power consumed by the short-circuit currents is typically less than 10% of the total dynamic power.

An important point to note is that if the supply is lowered to be below the sum of the thresholds of the transistors, $V_{dd} < V_{Tn} + |V_{Tp}|$, the short-circuit currents can be virtually (since sub-threshold currents will still flow) *eliminated* because both devices cannot conduct simultaneously for any value of the input voltage.

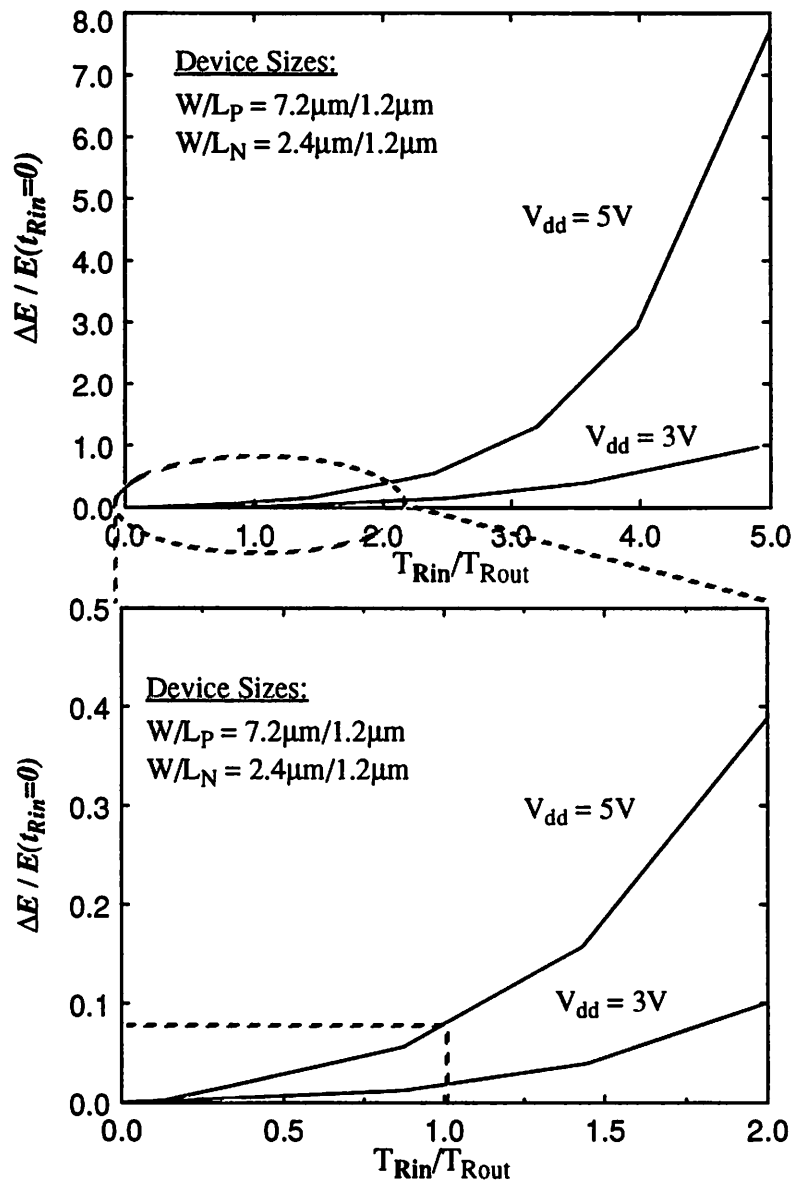


Figure 2-30 : Short-circuit component of power.

2.3 Leakage Component of Power

There are two types of leakage currents: reverse-bias diode leakage at the transistor drains, and sub-threshold leakage through the channel of an “off” device. The magnitude of these currents is set predominantly by the processing technology; however, there are some things that a designer

can do to minimize their contribution.

2.3.1 Diode Leakage

The diode leakage occurs when a transistor is turned off and another active transistor charges up/down the drain with respect to the former's bulk potential. For example, consider an inverter with a high input voltage, in which the NMOS transistor is turned on and the output voltage is driven low. The PMOS transistor will be turned off, but its drain-to-bulk voltage will be equal to $-V_{dd}$ since the output voltage is at 0V and the bulk for the PMOS is at V_{dd} (as shown in Figure 2-31).

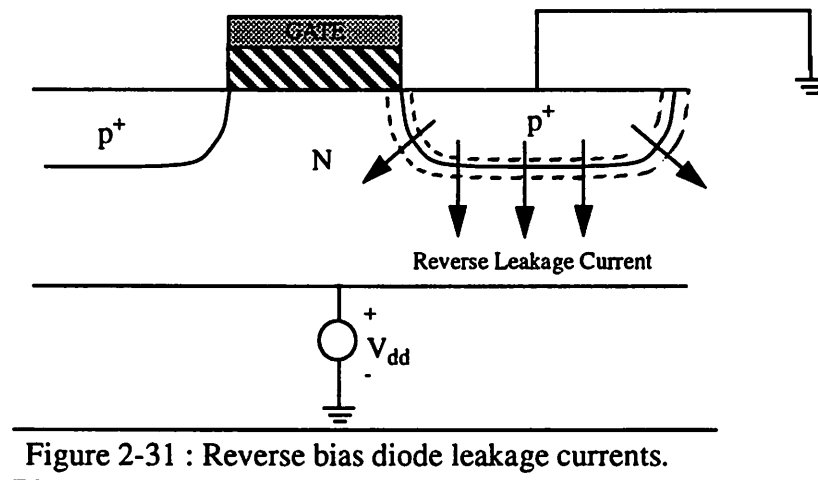


Figure 2-31 : Reverse bias diode leakage currents.

The leakage current for the diode is given by

$$I_{leakage} = I_s \left(e^{\frac{V}{V_T}} - 1 \right) \quad (\text{EQ 52})$$

where I_s is the reverse saturation current, V is the diode voltage, $V_T = KT/q$ is the thermal voltage. Due to exponential dependence, even with a small reverse bias voltage across the diode, the leakage current will be relatively independent of the voltage and will equal to the reverse saturation current.

The reverse saturation current per unit area is defined as the current density J_s , and the resulting

current for any will be approximately $I_L = A_D J_S$, where A_D is the area of the drain diffusion. For a typical CMOS process, J_S is approximately $1\text{-}5 \text{ pA}/\mu\text{m}^2$ (25°C), and the minimum A_D is $7.2 \mu\text{m}^2$ for a $1.2 \mu\text{m}$ minimum feature size. J_S doubles with every 9 degree increase in temperature. For a 1 million transistor chip, assuming an average drain area of $10 \mu\text{m}^2$, the total leakage current is on the order of $25 \mu\text{A}$. While this is typically a small fraction of the total power consumption in most chips, it could be significant for a system application which spends much of its time in standby operation, since this power always being dissipated even when no switching is occurring.

2.3.2 Sub-threshold Leakage

The other component is the subthreshold leakage which occurs due to carrier diffusion between the source and the drain when the gate-source voltage, V_{gs} , has exceeded the weak inversion point, but is still below the threshold voltage V_t , where carrier drift is dominant. In this regime, the MOSFET behaves similarly to a bipolar transistor, and the subthreshold current is exponentially dependent on the gate-source voltage V_{gs} (as seen from Figure 2-32). The current in the subthreshold region is given by:

$$I_{ds} = K e^{(V_{gs} - V_t)/(nV_T)} \left(1 - e^{-\frac{V_{ds}}{V_T}} \right) \quad (\text{EQ 53})$$

where K is a function of the technology, V_T is the thermal voltage (KT/q) and V_t is the threshold voltage and $n = 1 + \Omega t_{ox}/D$, where t_{ox} is the gate oxide thickness, D is the channel depletion width, and $\Omega = \epsilon_{si}/\epsilon_{ox}$. For $V_{ds} \gg V_T$, $(1 - e^{-V_{ds}/V_T}) \approx 1$; that is, the drain to source current leakage current is independent of the drain-source voltage V_{ds} , for V_{ds} approximately larger than 0.1V .

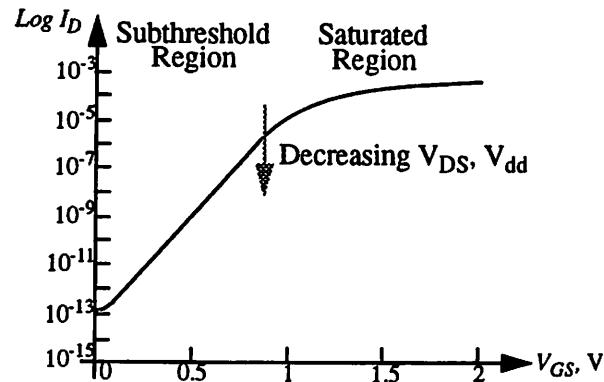


Figure 2-32 : Sub-threshold leakage component of power.

Associated with this is the subthreshold slope S_{th} , which is the amount of voltage required to drop the subthreshold current by one decade. The subthreshold slope can be determined by taking the ratio of two points in this region (from Equation 53):

$$\frac{I_1}{I_2} = e^{(V_1 - V_1)/(nV_T)} \quad (\text{EQ 54})$$

which results in $S_{th} = nV_T \ln(10)$. At room temperature, typical values for S_{th} lie between 60 to 90 mV/(decade current), with 60 mV/dec being the lower limit. Clearly, the lower S_{th} is, the better, since it is desirable to have the device “turn-off” as close to V_t as possible.

As a reference, for an $L=1.5\mu$, $W=70\mu$ NMOS device, at the point where V_{gs} equals V_t , with V_t defined as where the surface inversion charge density is equal to the bulk doping, approximately $1\mu\text{A}$ of leakage current is exhibited, or $.014\mu\text{A}/\text{micron}$ of gate width[Sze81]. The issue is whether this extra current is negligible in comparison to the time-average current during switching. For a CMOS inverter (PMOS: $W=4\mu$, NMOS: $W=8\mu$), the current was measured to be $64\mu\text{A}$ over 3.7nsec at a supply voltage of 2V. This implies that there would be a 100% power penalty for subthreshold leakage if the device were operating at a clock speed of 25 MHz with an activity

factor of $p_t = 1/6$ th, i.e. the devices were left idle and leaking current 83% of the time. It is not advisable, therefore, to use a true zero threshold device, but instead to use thresholds of at least 0.2V, which provides for at least two orders of magnitude of reduction of subthreshold current. The topic of optimal threshold voltage selection is presented in the next chapter.

2.4 Static Power

Migrating from NMOS technology to CMOS, has resulted in logic circuits that primarily consume power only during switching. There are two situations, however, when circuits dissipate power in steady state operation (i.e. non transient operation): degenerated voltage levels feeding into complementary static gates and pseudo NMOS circuit styles.

2.4.1 Reduced Voltage Levels Feeding CMOS Gates

Figure 2-33 shows an NMOS pass transistor driving a CMOS inverter. With V_{dd} applied to both the gate and the drain of the NMOS pass transistor, node B will rise to $V_{dd} - V_{tn}$. In steady state, this input will cause the inverter output to be forced low. The PMOS device will however be weakly ON (since $|V_{gs} - V_t|$ is approximately equal to 0), conducting static current from the supply to ground. This might consume significant power if the circuit is idle most of the time, and in this case, a weak level restoring PMOS transistor must be introduced from the output node to node B to pull it all the way to the rail.

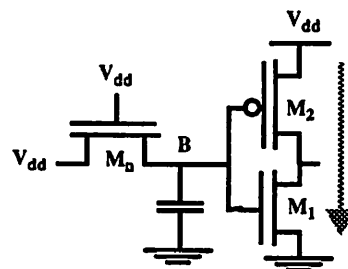


Figure 2-33 : Degenerated voltage level feeding a CMOS gate resulting in static power.

2.4.2 Pseudo-NMOS Logic Style

Pseudo-NMOS circuits also dissipate static power. In this logic implementation style, the PMOS pullup network of the CMOS implementation is replaced by a single PMOS device whose gate is grounded and is therefore always ON. When the output is driven low by a resistive path from the output node to ground through the NMOS network, there is a conducting path from the supply to ground. This implementation style can reduce power only for implementation that require complex logic switching at high frequencies. Figure 2-34 shows an example of such a gate -- a wide AND-OR-Invert gate. To implement this in full static CMOS would require several times the area to implement the stacked PMOS transistors. The extra PMOS transistors would also increase the capacitance on the input nodes, which would load down the previous gates. There is usually a large area savings as well. This is primarily applicable for higher-frequency circuits, where the savings of dynamic power is proportionally higher. If the circuit is idle most of the time (when the circuit is clocked at low frequencies), then the static power will tend to increase the total power consumption.

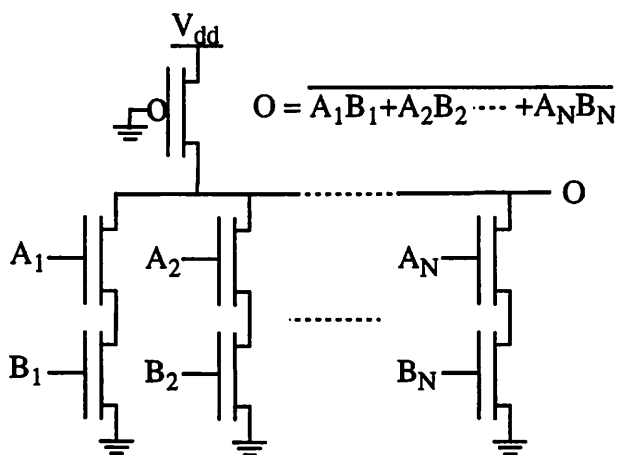


Figure 2-34 : Implementing complex logic using pseudo-NMOS.

2.5 Summary

There are four components to power consumption in CMOS technology arising from switching

currents, short-circuit currents, leakage currents and finally static currents. For “properly” designed circuits the switching component will dominate and will contribute to more than 90% of the total power consumption, making it the primary target for power reduction.

The switching component of power for a CMOS gate with load capacitor C_L is given by $\alpha C_L V_{dd}^2 f$ where α is node transition activity, C_L is physical load capacitance, V_{dd} is supply voltage and f is operating frequency. There are two components to node transition activity: transitions due to the static behavior of the circuit (in which the transitions are determined strictly from the boolean logic ignoring timing skew) and transitions that occur due to the dynamic nature of the circuit (that is due to timing skew). The node transition activity factor is a function of the logic function being implemented, the logic style, the circuit topologies, signal statistics, signal correlations, and the sequencing of operations. A system level approach which involves optimizing algorithms, architectures, logic design, circuit design and physical design can be used to minimize the switched capacitance and is described in detail in Chapter 4. The components of physical capacitance include the transistor parasitics including the gate and diffusion capacitances and the interconnect capacitance. The physical capacitance can be minimized through choice of substrate (like SOI), layout optimization, device sizing (as described in the next chapter) and choice of logic style. The choice of supply voltage has the greatest impact on the power-delay product, which is the amount of energy required to perform a given function. It is only necessary to reduce the supply voltage to quadratically improve the power-delay product. Unfortunately, a reduction in circuit speed is associated with supply voltage scaling. However, if the goal is to reduce the power consumption for realizing a fixed functionality or if the goal is to increase the MIPS/Watt in general purpose computing for a fixed MIPS level, then various architectural schemes can be used for voltage reduction (Chapter 3).

The second component of power is the short-circuit power (also called direct path power) which exists when there is a direct conducting path from supply to ground. Through proper choice of transistors, the short-circuit power can be kept to less than 10%. Alternatively, operating the circuits

at a supply voltage less than the sum of the NMOS and PMOS threshold voltages will essentially eliminate any short-circuit currents. The architecture-driven voltage scaling strategy described in the next chapter results in supply voltages which satisfy this condition.

The third component of power is due to leakage currents. There are two types of leakage currents: reverse-bias diode leakage at the transistor drains, and sub-threshold leakage through the channel of an “off” device. The subthreshold leakage occurs due to carrier diffusion between the source and the drain when the gate-source voltage, V_{gs} , has exceeded the weak inversion point, but is still below the threshold voltage V_t , where carrier drift is dominant. In this regime, the MOSFET behaves similarly to a bipolar transistor, and the subthreshold current is exponentially dependent on the gate-source voltage V_{gs} . An important figure of merit for a low-power technology is the subthreshold slope S_{th} , which is the amount of voltage required to drop the subthreshold current by one decade. The lower the subthreshold slope, the better since the devices can be turned-off as close to V_t as possible.

The final component of power is the static currents found in circuits that do not have rail-to-rail swing feeding other circuits, pseudo-NMOS logic styles, and in analog bias circuits. The static currents must be minimized as much as possible. For example, in SRAM sense amplifiers, pulsed circuits should be used to minimize static currents in sense amplifiers.

CHAPTER 3

Approaches to Voltage Scaling and Issues for Low Voltage Operation

Since the dominant component of power consumption for a properly designed CMOS circuit is proportional to the square of the supply voltage, and as Figure 2-27 illustrates, operating circuits at slower speeds through lower voltages is the key to minimizing the energy consumed per operation. However, the approach of running circuits at the slowest possible speed must not compromise realization of the functional throughput. The approach which will be presented here, assumes that the application, which is desired to be implemented with low power is known, and trade-offs can be made as long as the functionality required of this application is met within a given time constraint.

In section 3.1, various approaches to supply voltage scaling are presented. First, a survey of three previous approaches to supply-voltage scaling is presented, which are focused on maintaining reliability, maintaining performance, and optimizing the speed-energy characteristics for a given technology. Then the approach of voltage scaling through transistor sizing is presented, from which conclusions are made about the optimum transistor sizing strategy for low-power. This is followed by our architecture-driven approach, from which an “optimal” supply voltage based on

technology, architecture and noise margin constraints is derived. The effect of technology modification (specifically changing the threshold voltage) on the “optimum” supply voltage is presented. In section 3.2, the various support circuitry for low-voltage operation, which include high efficiency DC/DC converters and voltage level-shifters, are presented.

3.1 Approaches to Supply Voltage Scaling

3.1.1 Reliability-Driven Voltage Scaling

One approach to the selection of an optimal power supply voltage for deep-submicron technologies is based on optimizing the trade-off between speed, long-term reliability, and power dissipation [Davari88]. Circuit speed is increased in designs with higher electric fields owing to increased carrier concentration and velocity. Constant-voltage scaling - the most commonly used technique - results in higher electric fields that create hot carriers. As a result of this, the devices degrade with time (including changes in threshold voltages, degradation of transconductance, and increase in subthreshold currents), leading to eventual breakdown.

One approach to reduce the number of hot carriers is to change the device structure to reduce the fields inside or to limit the spatial extent of the high field regions so that carriers do not gain enough energy from the field to cause difficulty [Watts89]. Since high fields occur only near the drain, the drain doping profile can be tailored such that the potential gradient is more gradual. One solution to reducing the number of hot carriers is to change the physical device structure, such as the use of LDD (lightly doped drain), usually at the cost of decreased performance. Figure 3-1 shows the schematic diagram of the drain doping profile of a conventional MOSFET and an LDD device.

A lightly doped implant is introduced between the heavily doped n+ region and the channel. This structure can be obtained by two independent implantations, with one of them offset by an oxide

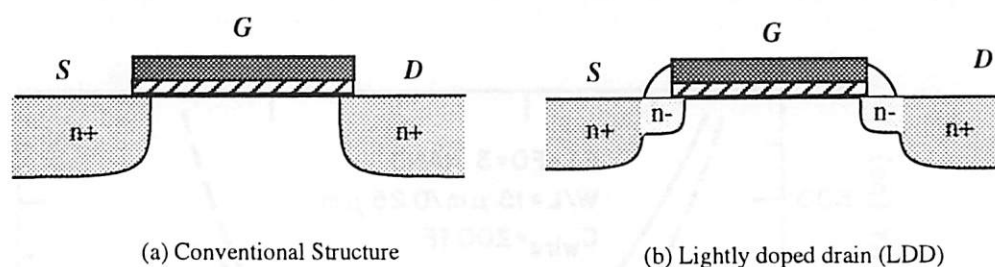


Figure 3-1 : Drain doping profile for a conventional profile and an LDD device.

sidewall on the gate structure. This structure reduces the peak electric field on the drain side of the channel improving reliability. The main problem with LDD is however an increase in the series resistance of the drain and source which results in reduced performance.

Thus the reliability driven power supply voltage selection is based on a trade-off between performance and reliability. An empirical relationship between channel hot carriers (CHC) limited voltage and device series resistance (added due to the drain modification of LDD structures) has been developed from measurements on a variety of junction technologies [Davari88]. Using this relationship, a model of circuit performance of a technology as a function of power supply voltage and CHC margin can be determined. Figure 3-2 shows a plot of simulated delay vs. power supply voltage. The solid line represents the delay curve for a fixed device structure. The dotted line describes the case where the hot carrier margin is kept fixed by altering the device structure (i.e. varying the series resistance of the source/drain junctions).

The delay starts to increase with increasing supply voltage since the parasitic resistance of the LDD structure must be increased to maintain the CHC margin and this limits the circuit performance. Therefore a reliability driven optimum supply voltage can be found; for this $0.25\mu\text{m}$ technology used, an optimal voltage of 2.5V was found by choosing the minimum point on the delay vs. V_{dd} curve.

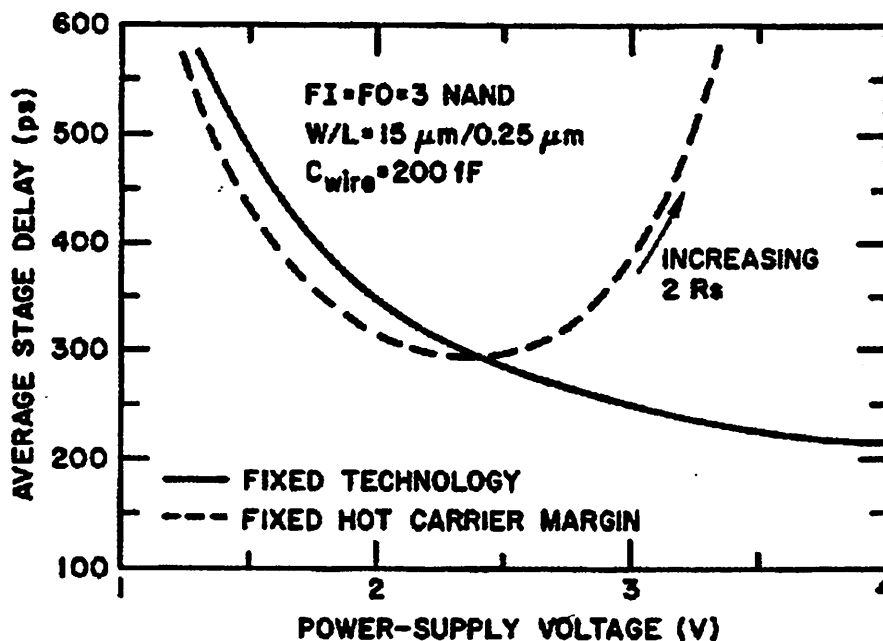


Figure 3-2 : Reliability driven supply voltage selection [Davari88].

Clearly it does not make sense from a power perspective to operate above the optimum voltage since throughput degrades and the energy consumed is increased. However, operating below the reliability driven optimum voltage is desirable but will result in loss of circuit speed. As will be shown in Section 3.1.5, architecture modification can result in lower power consumption at voltages much lower than the optimum determined in Figure 3-2.

3.1.2 Technology-Driven Voltage Scaling

Another approach for voltage selection has been proposed based on exploiting device level properties involving velocity saturation. The simple first order delay analysis presented in Section 2.1.6 is reasonably accurate for long channel devices. However, as feature sizes shrink below $1.0\mu\text{m}$, the delay characteristics as a function of lowering the supply voltage deviate from the first order theory presented since it does not consider carrier velocity saturation under high electric fields [Bakoglu90]. As a result of velocity saturation, the current is no longer a quadratic function of the

voltage but linear; hence, the current drive is significantly reduced and is approximately given by

$$I = WC_{ox} (V_{dd} - V_t) v_{max} \quad (\text{EQ 55})$$

Given this and the equation for delay as $C_L V_{dd}/I$, we see that the delay for submicron circuits is relatively independent of supply voltages at high electric fields.

A “technology” based approach proposes choosing the power supply voltage based on maintaining the speed performance for a given submicron technology [Kakumu90]. By exploiting the relative independence of delay on supply voltage at high electric fields, the voltage can be dropped to some extent for a velocity-saturated device with very little penalty in speed performance. This implies that there is little advantage to operating above a certain voltage. This idea has been formalized by Kakumu et. al., yielding the concept of a “critical voltage” which provides a lower limit on the supply voltage [Kakumu90].

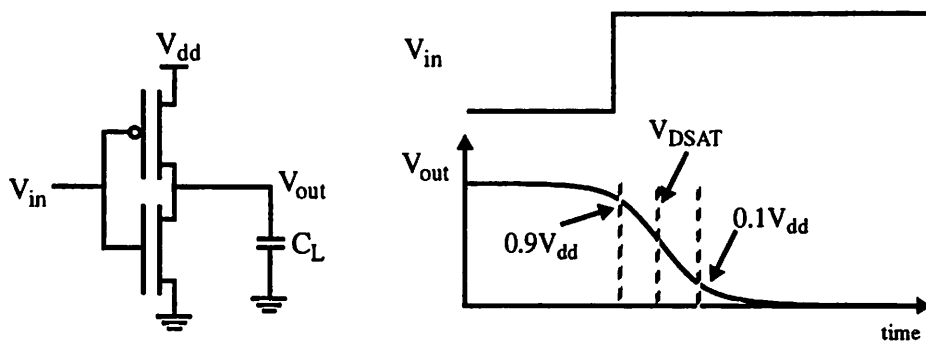


Figure 3-3 : Model for computing delay and critical voltage.

Figure 3-3 shows the model for computing the delay of circuit under high electric field effects and the “critical voltage” as defined by Kakumu. The model used for the MOS devices under high electric field is the one proposed by Sodini [Sodini84].

$$I_D = \frac{W_{eff}\mu_{eff}C_{ox}\left(V_G' - \frac{V_{DS}}{2}\right)V_{DS}}{L_{eff}\left(1 + \frac{V_{DS}}{E_C L_{eff}}\right)} \quad (V_{DS} \leq V_{DSAT}) \quad (\text{EQ 56})$$

$$I_D = \frac{W_{eff}\mu_{eff}C_{ox}\left(V_G' - V_{DSAT}\right)E_C}{2} \quad (V_{DS} > V_{DSAT}) \quad (\text{EQ 57})$$

where

$$V_{DSAT} = \frac{E_C L_{eff} V_G'}{E_C L_{eff} + V_G'} \quad (\text{EQ 58})$$

and V_G' is the gate to source voltage minus the threshold voltage and E_C is the critical electric field at which the carrier velocity is saturated and is equal to $2v_{sat}/\mu_{eff}$. The delay of the circuit is the time taken for the output to transition from $0.9V_{dd}$ to $0.1V_{dd}$. For this range the NMOS transistor goes through two regions: from $0.9V_{dd}$ to V_{DSAT} (saturation) and from V_{DSAT} to $0.1V_{dd}$ (linear).

Lets these times be defined as τ_1 and τ_2 . The delays are then determined as:

$$\tau_1 = \frac{2C_L(0.9V_{dd} - V_{DSAT})}{W_{eff}\mu_{eff}C_{ox}\left(V_G' - V_{DSAT}\right)E_C} \quad (\text{EQ 59})$$

$$\tau_2 = \frac{C_L L_{eff}}{W_{eff}\mu_{eff}C_{ox}} \left(\frac{1}{V_G'} \ln \frac{V_{DSAT}}{0.1V_{dd}} A + \frac{2}{E_C L_{eff}} \ln A \right) \quad (\text{EQ 60})$$

where A is

$$A = \frac{2V_G' - 0.1V_{dd}}{2V_G' - V_{DSAT}} \quad (\text{EQ 61})$$

To understand the two delay terms some simplifications can be made. For τ_1 it can be assumed that $(0.9V_{dd} - V_{DSAT})/(V_G' - V_{DSAT})$ is approximately equal to 1. This means that τ_1 is to first order independent of the power supply voltage. For τ_2 , the second term in the bracket of Equation 60 is

negligible and therefore τ_2 can be approximated as:

$$\tau_2 \approx \frac{C_L L_{eff}}{W_{eff} \mu_{eff} C_{ox}} \left(\frac{1}{V_G} \ln \frac{V_{DSAT} A}{0.1 V_{dd}} \right) \quad (\text{EQ 62})$$

since the \ln term is nearly constant, it can be seen that τ_2 is inversely proportional to V_{dd} . Therefore, the total delay $= \tau_1 + \tau_2$ contains a term which is independent of supply voltage and one term which is dependent on supply voltage. Figure 3-4 plots the fall time as a function of the power supply voltage on a log-log plot and the voltage dependent and voltage independent terms are identified. The delay time decreases initially with increasing power supply voltage since the delay time is dominated by τ_2 . On the other hand, when V_{dd} becomes larger than the critical voltage, V_C (the voltage at which, $\tau_1 = \tau_2$), where τ_1 is longer than τ_2 and thus the delay is dominated by τ_1 . At voltages much greater than the critical voltage, the delay is virtually independent of the power supply voltage. The delay improvements are not very significant above the critical voltage and therefore this voltage is regarded as the lower limit of power supply voltage.

Kakumu and Kinugawa have computed this critical voltage making the assumption that the threshold voltage is $0.2V_{dd}$, and have come up a critical voltage of $1.1E_c L_{eff}$, where E_c is the critical electric field causing velocity saturation; For an experimental delay vs. V_{dd} curve, this is the voltage at which the curve approaches a $\sqrt{V_{dd}}$ dependence. For 0.3μ technology, the proposed lower limit on supply voltage (or the critical voltage) was found to be 2.43V.

Because of this effect, there is strong movement to a 3.3V industrial voltage standard since at this level of voltage reduction there is not a significant loss of circuit speed[Bell91][Dhale91]. This was found to achieve a 60% reduction in power when compared to a 5 volt operation [Dhale91].

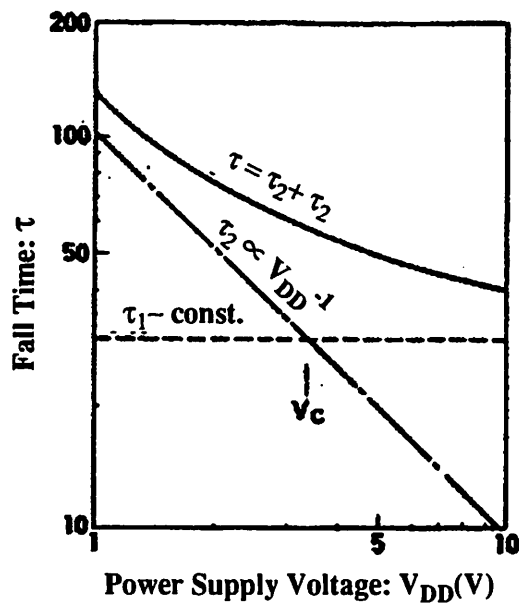


Figure 3-4 : Two components of delay and critical voltage [Kakamu90].

3.1.3 Energy x Delay Minimum Based Voltage Scaling

Another approach has been proposed for voltage reduction which involves minimizing the energy x delay product [Burr91]. For a fixed technology (both feature size and threshold voltage), [Burr91] proposes a supply voltage that compromises between the quadratically reduced energy per computation (due to a lower supply voltage) and the increased circuit delays. The “optimum” supply voltage can be computed analytically assuming that the transistors operate mostly in the saturation region and assuming that the threshold voltages for the NMOS and PMOS are equal:

$$\text{Energy} \cdot T_d = K \cdot C_L \cdot V_{dd}^2 \cdot V_{dd} / (V_{dd} - V_t)^2 \quad (\text{EQ 63})$$

Taking the derivative of the Energy $\cdot T_d$ and solving for the supply voltage results in a minimum (or optimum based on this strategy) of $3V_t$. Assuming a $V_t = 0.8$ (average of the $V_{tn} = 0.7$ and $V_{tp} = -0.9$ for a Mosis technology), the minimum lies at 2.4V. This can be seen from the plot of measure normalized Energy $\cdot T_d$ plot (Figure 3-5) for a ring oscillator as a function of V_{dd} . In this

case, power consumption is lowered at the expense of a *reduced* computational throughput. This is not desirable for throughput constrained functions where the hardware has to meet a required functional throughput.

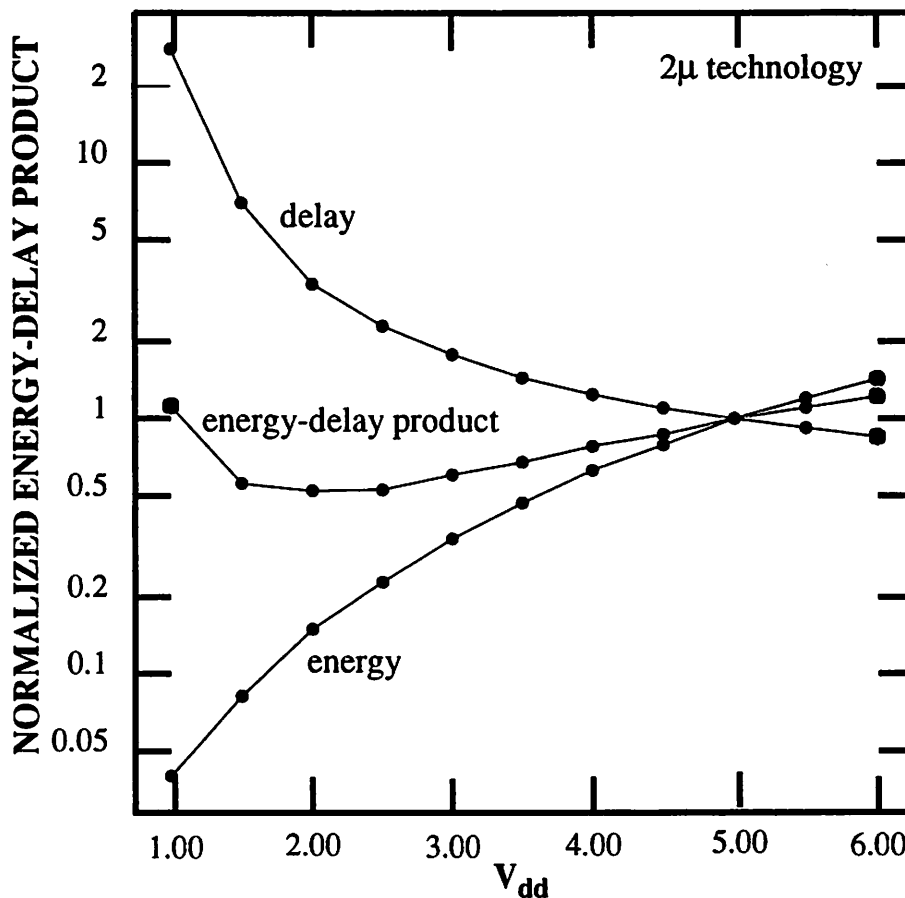


Figure 3-5 : Energy x Delay product vs. V_{dd} .

3.1.4 Voltage Scaling Through Optimal Transistor Sizing

Independent of the choice of logic family or topology, optimized transistor sizing will play an important role in reducing power consumption. For low power, as is true for high speed design, it is important to equalize all delay paths so that a single critical path does not unnecessarily limit the performance of the entire circuit. However, beyond this constraint, there is the issue of what extent the (W/L) ratios should be uniformly raised for all the devices, yielding a uniform decrease in the gate delay and hence allowing for a corresponding reduction in voltage and power. It is shown in

this section, that if voltage is allowed to vary, that the optimal sizing for low power operation is quite different from that required for high speed.

In Figure 3-6, a simple two-gate circuit is shown, with the first stage driving the gate capacitance of the second, in addition to the parasitic capacitance C_p due to substrate coupling and interconnect. Assuming that the input gate capacitance of both stages is given by NC_{ref} , where C_{ref} represents the gate capacitance of a MOS device with the smallest allowable (W/L), then the delay through the first gate at a supply voltage V_{ref} is given by:

$$T_N = K \frac{(C_p + NC_{ref})}{(NC_{ref})} \frac{V_{ref}}{(V_{ref} - V_t)^2} = K (1 + \alpha/N) \frac{V_{ref}}{(V_{ref} - V_t)^2} \quad (\text{EQ 64})$$

where α is defined as the ratio of C_p to C_{ref} , and K represents terms independent of device width and voltage. For a given supply voltage V_{ref} , the speed up of a circuit whose W/L ratios are sized up by a factor of N over a reference circuit using minimum size transistors ($N=1$) is given by $(1 + \alpha/N)/(1 + \alpha)$. In order to evaluate the energy performance of the two designs at the same speed, the voltage of the scaled solution is allowed to vary as to keep delay constant. Assuming that the delay scales as $1/V_{dd}$ (ignoring threshold voltage reductions in signal swings) the supply voltage, V_N , where the delay of the scaled design and the reference design are equal is given by:

$$V_N = \frac{(1 + \alpha/N)}{(1 + \alpha)} V_{ref} \quad (\text{EQ 65})$$

Under these conditions, the energy consumed by the first stage as a function of N is given by:

$$\text{Energy}(N) = (C_p + NC_{ref}) V_N^2 = \frac{NC_{ref} (1 + \alpha/N)^3 V_{ref}^2}{(1 + \alpha)^2} \quad (\text{EQ 66})$$

After normalizing against E_{ref} (the energy for the minimum size case), Figure 3-7 shows a plot of

Energy(N) / Energy(1) vs. N for various values of α . When there is no parasitic capacitance contribution (i.e. $\alpha = 0$), the energy increases linearly with respect to N, and the solution utilizing devices with the smallest (W/L) ratios results in the lowest power. At high values of α , when parasitic capacitances begin to dominate over the gate capacitances, the power decreases temporarily with increasing device sizes and then starts to increase, resulting in an optimal value for N. The initial decrease in supply voltage achieved from the reduction in delays more than compensates the increase in capacitance due to increasing N. However, after some point the increase in capacitance dominates the achievable reduction in voltage, since the incremental speed increase with transistor sizing is very small (this can be seen in Equation 64, with the delay becoming independent of α as N goes to infinity). Throughout the analysis we have assumed that the parasitic capacitance is independent of device sizing. However, the drain and source diffusion and perimeter capacitances actually increase with increasing area, favoring smaller size devices and making the above a worst-case analysis.

Also plotted in Figure 3-7 are simulation results using the SPICE simulator from extracted layouts of an 8-bit adder carry chain for three different device (W/L) ratios (N=1, N=2, and N=4). The curve follows the simple first-order model derived very well, and suggests that this example is dominated more by the effect of gate capacitance rather than parasitics. In this case, increasing devices (W/L)'s does not help, and the solution using the smallest possible (W/L) ratios results in the best sizing.

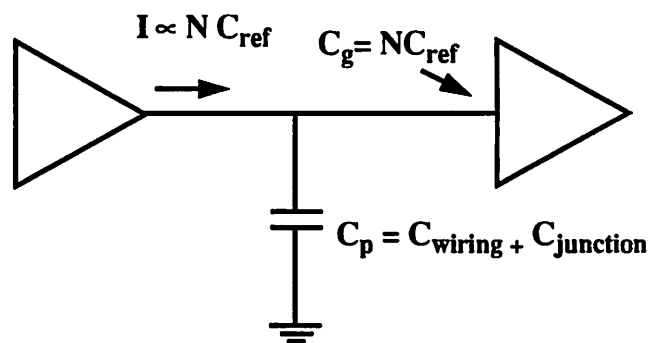


Figure 3-6 : Circuit model for analyzing the effect of transistor sizing.

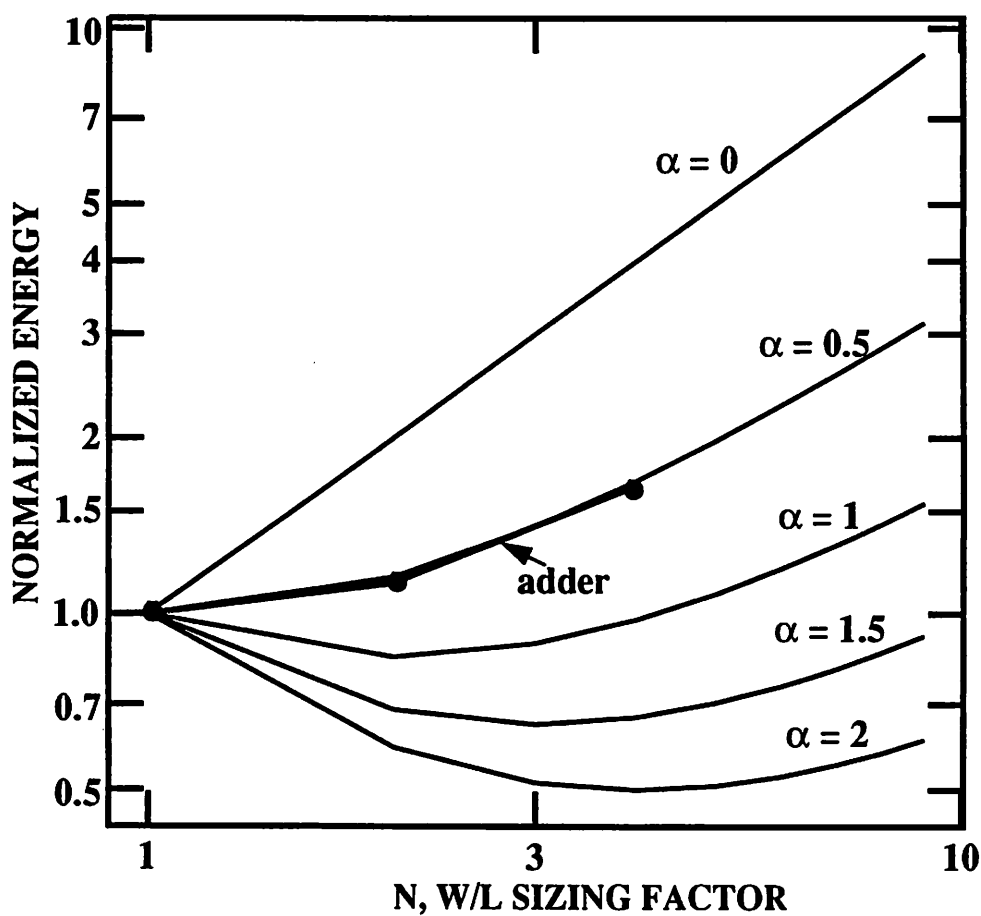


Figure 3-7 : Plot of energy vs. transistor sizing factor for various parasitic contributions.

3.1.5 Architecture-Driven Voltage Scaling

The “technology” based approaches presented in Sections 3.1.1 and 3.1.2 are focusing on reducing the voltage while maintaining device speed, and are not attempting to achieve the minimum possible power. As shown in Figures 2-25 and 2-27, CMOS logic gates achieve lower power-delay products (energy per computation) as the supply voltages are reduced. In fact, once a device is in velocity saturation there is a further degradation in the energy per computation, so in minimizing the energy required for computation, Kakumu’s critical voltage for low power therefore provides an *upper* bound on the supply voltage whereas for his analysis it provided a *lower* bound! It now will be the task of the architecture to compensate for the reduced circuit speed, that comes with operating below the critical voltage.

Trading Area for Lower power through Hardware Duplication

Figure 3-8 shows a conventional uni-processor implementation of a generic function, $F(\text{INPUT})$. Let $f_{\text{sample}} (= 1/T_{\text{sample}})$ be the throughput clock rate for this implementation (or the rate at which the input samples are streamed through) and also let f_{sample} be the fastest clock rate at which the function $F(\text{INPUT})$ can be clocked at (that is, the critical path of $F(\text{IN})$ is T_{sample} at a supply voltage of $V_{\text{ref}} = 5\text{V}$). Let C_{ref} be the capacitance switched every cycle,

$$C_{\text{ref}} = C_{\text{in-reg}} + C_{\text{F}} + C_{\text{out-reg}} \quad (\text{EQ 67})$$

where $C_{\text{in-reg}}$ is the capacitance switched in the input register, C_{F} is the capacitance switched to implement the function, and $C_{\text{out-reg}}$ is the capacitance of the output register.

The power consumption of this datapath is given by,

$$P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{sample}} \quad (\text{EQ 68})$$

Figure 3-8 also shows the timing diagram for input and output data for this implementation. The output data is valid one clock cycle after the input data has been latched into the input register. The latency is therefore one clock cycle.

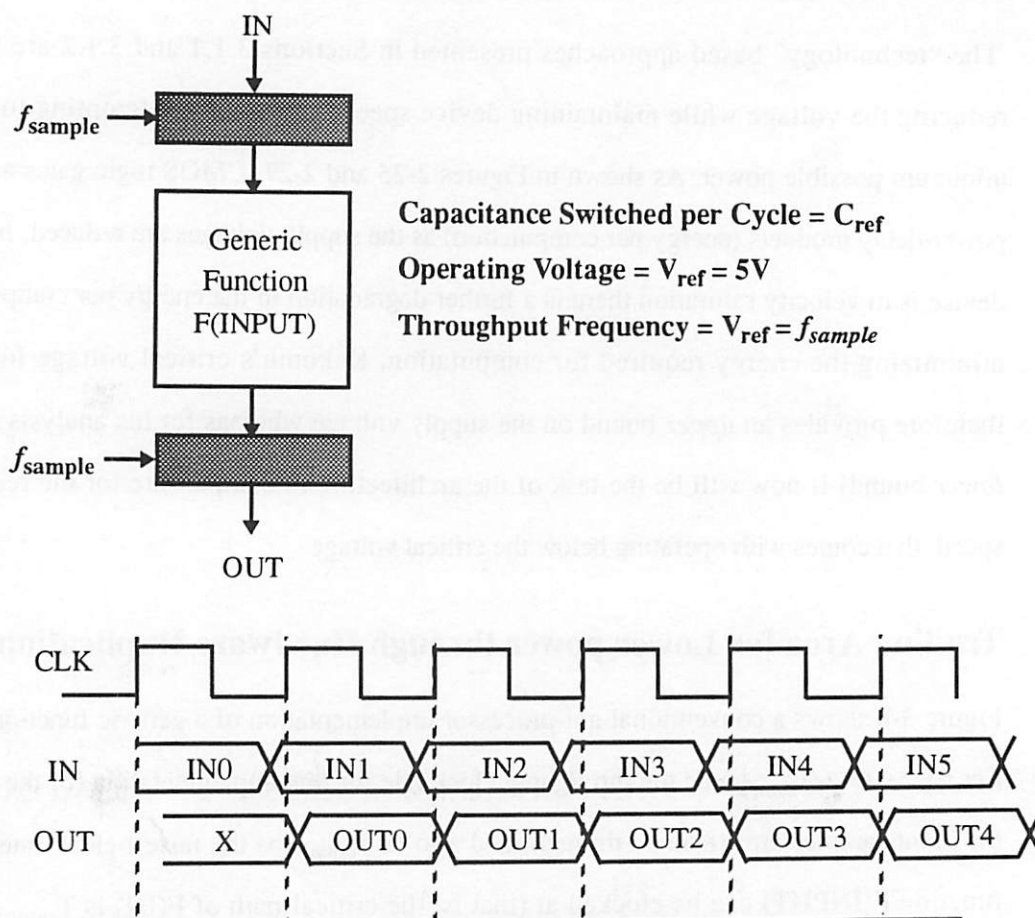


Figure 3-8 : A uni-processor implementation of a generic function.

Another approach for implementing this generic function $F(\text{INPUT})$ is shown in Figure 3-9, which shows the generalized approach of using parallelism to reduce the clock requirement. Hardware duplication is used in which N processing elements each implementing the function F are computing in parallel. The input is broadcast to all the registers of the N processing elements and gated clocks are used to load each register every N cycles. That is, the clocks to each register are skewed such that IN_0 is loaded in REG_0 , IN_1 is loaded into REG_1 , and so on. The output of the N processing elements is multiplexed and sent to an output register which operates at the throughput clock rate (f_{sample}). Since each input register is clocked only at f_{sample}/N , the time available to compute the function F for each input sample is increased by a factor of N over the uni-processor

implementation. That is, $T_{\text{available}}/T_{\text{critical-path}} = N$. This implies that the supply voltage can be lowered to increase the circuit delays until the critical path is extended to equal to the clock cycle.

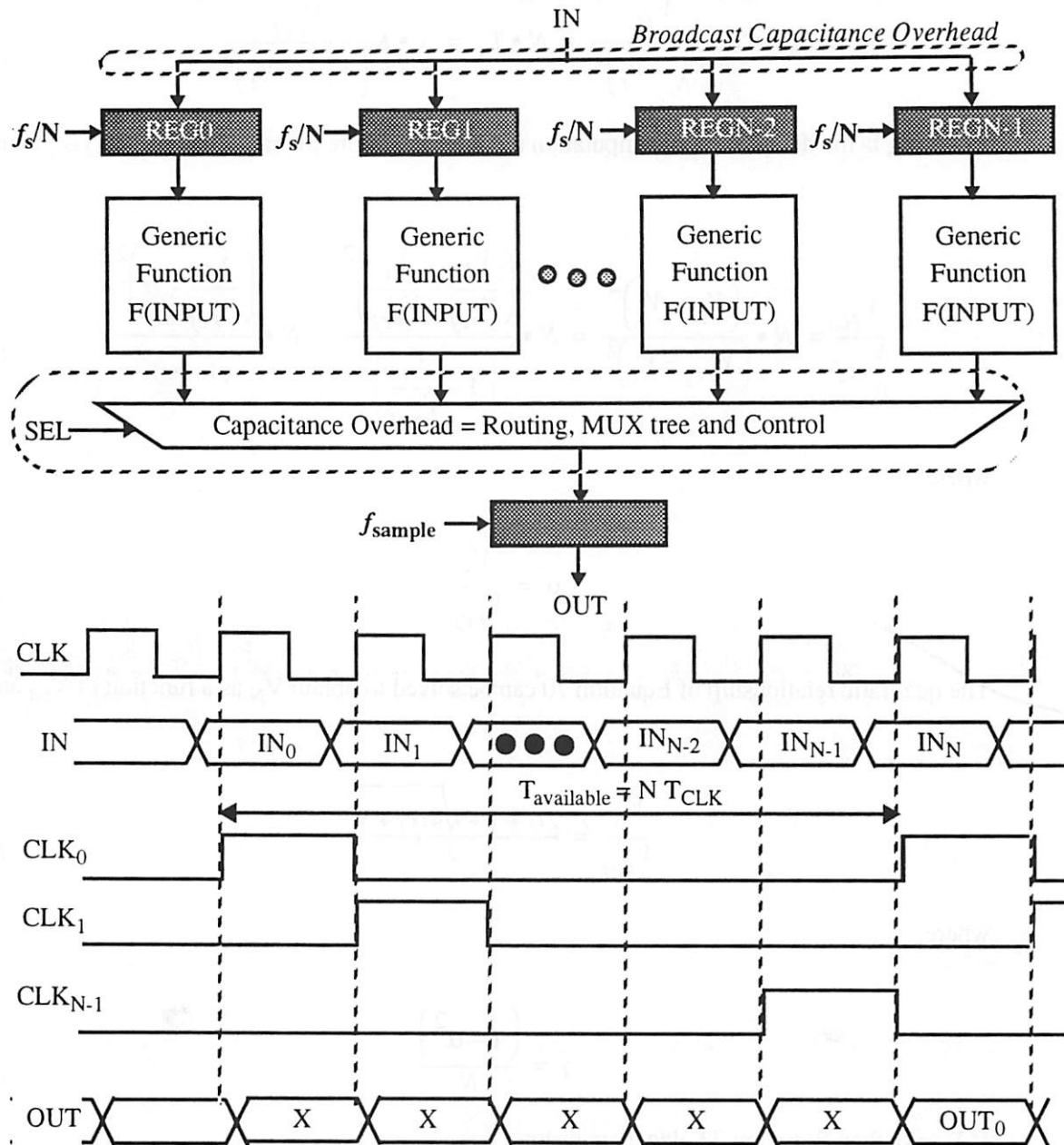


Figure 3-9 : A multi-processor implementation of a generic function.

The power supply voltage corresponding to N -level parallelism, V_N , is determined below:

$$T_N = K \frac{V_N}{(V_N - V_t)^2} = N \cdot T_1 = N \cdot K \frac{V_{ref}}{(V_{ref} - V_t)^2} \quad (\text{EQ 69})$$

where T_N is the delay of each computation module. Therefore the ratio of V_N to V_{ref} is given by:

$$\frac{V_N}{V_{ref}} = N \cdot \frac{(V_N - V_t)^2}{(V_{ref} - V_t)^2} = N \cdot \frac{\left(\frac{V_N}{V_{ref}} - \frac{V_t}{V_{ref}}\right)^2}{\left(1 - \frac{V_t}{V_{ref}}\right)^2} = N \cdot \frac{\left(\frac{V_N}{V_{ref}} - \alpha\right)^2}{(1 - \alpha)^2} \quad (\text{EQ 70})$$

where

$$\alpha = \frac{V_t}{V_{ref}} \quad (\text{EQ 71})$$

The quadratic relationship of Equation 70 can be solved to obtain V_N as a function of V_{ref} and V_t .

$$\frac{V_N}{V_{ref}} = \frac{2\alpha + \gamma + \sqrt{4\alpha\gamma + \gamma^2}}{2} \quad (\text{EQ 72})$$

where

$$\gamma = \frac{(1 - \alpha^2)}{N} \quad (\text{EQ 73})$$

If $V_t = 0$, then Equation 72, degenerates to:

$$V_N = \frac{V_{ref}}{N} \quad (\text{EQ 74})$$

Figure 3-10 shows the supply voltage as a function of the number of processors. The best case situation is the bottom curve of Figure 3-10, where $V_t = 0$ and the power supply drops linearly with the supply voltage, as derived in Equation 74. Figure 3-10 also shows the effect of increasing the,

threshold voltage as modeled in Equation 72. As can be concluded from this figure, a high threshold voltage limits the amount of voltage scaling that can be achieved for given level of parallelism. A lower threshold voltage allows for more voltage scaling, however, results in more leakage currents. The issue of optimal V_t selection is described later in this chapter.

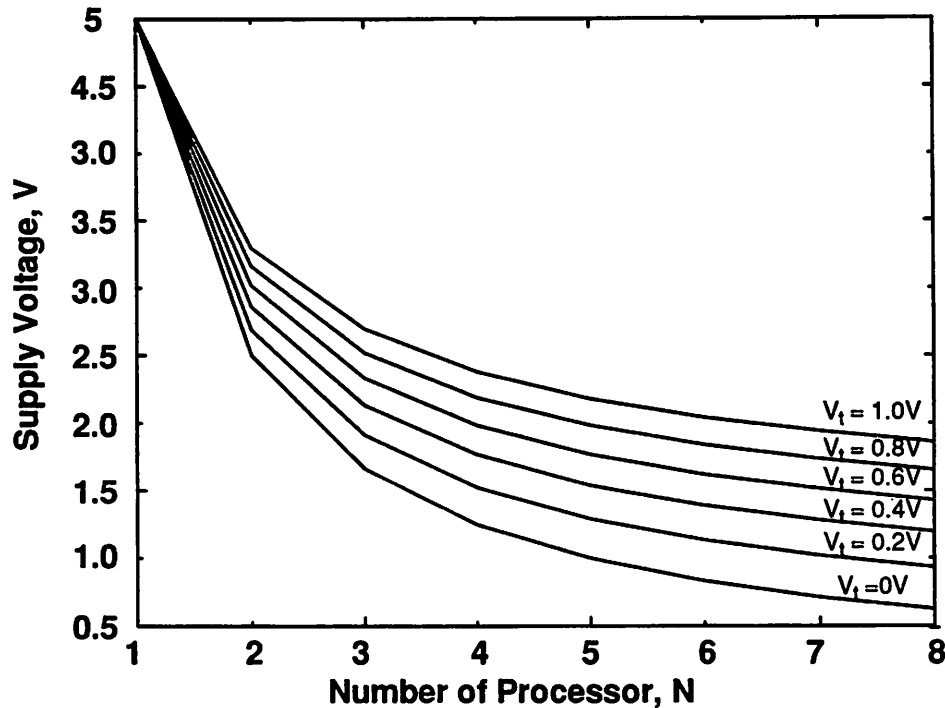


Figure 3-10 : Voltage reduction vs. number of processors as a function of V_t .

The parallel architecture shown in Figure 3-9 does have overhead circuitry which requires extra power. There are three sources of overhead power: routing capacitance due to broadcast of the input which is switched at the throughput rate, the output routing from the processors being clocked at f_{sample}/N and the multiplexor overhead circuitry operating at the sample rate. Therefore the power consumption of the parallel implementation is:

$$P_N = P_{in-routing} + P_{processing} + P_{out-routing} + P_{mux} + P_{outreg} \quad (\text{EQ 75})$$

The various components of power consumption are outlined below:

$$P_{in-routing} = C_{in}(N) V_N^2 f_{sample} \quad (\text{EQ 76})$$

where $C_{in}(N)$ is the capacitance switched in the input load capacitance which is equal to the interconnect capacitance plus the load of the registers and V_N , as defined before is the supply voltage of the N-processor implementation. The capacitance switched in the input routing is a function of N since the wire length (and therefore the corresponding physical capacitance) will increase with the number of processors.

$$P_{processing} = N(C_{inreg} + C_F) V_N^2 \frac{f_{sample}}{N} = (C_{inreg} + C_F) V_N^2 f_{sample} \quad (\text{EQ 77})$$

The processing power decreases quadratically with the voltage.

$$P_{out-routing} = N C_{out-routing}(N) V_N^2 \frac{f_{sample}}{N} = C_{out-routing}(N) V_N^2 f_{sample} \quad (\text{EQ 78})$$

Here $C_{out-routing}(N)$ is average capacitance switched per bus. Once again, it is a function of N since the average wire length per bus will increase as the number of processors increases.

$$P_{mux} = C_{mux}(N) V_N^2 f_{sample} \quad (\text{EQ 79})$$

where C_{mux} is the multiplexor switched capacitance which is function of N. That is, the capacitance per select is a function of the number of processors.

$$P_{outreg} = C_{outreg} V_N^2 f_{sample} \quad (\text{EQ 80})$$

where $C_{out-latch}$ is the capacitance switched per register access.

If the overhead components of capacitance described above, the multiplexor, input-routing and output-routing are assumed to be zero, then the power consumption can be arbitrarily reduced by

making the computation parallel. The power in this case is given by:

$$P_N = P_{processing} + P_{outreg} \quad (\text{EQ 81})$$

$$P_N = (C_{inreg} + C_F + C_{outreg}) V_N^2 f_{sample} = C_{ref} V_N^2 f_{sample} \quad (\text{EQ 82})$$

The power improvement of the multi-processor implementation relative to a uni-processor implementation under the assumptions of zero circuit overhead is:

$$\frac{P_N}{P_1} = \frac{V_N^2}{V_{ref}^2} \quad (\text{EQ 83})$$

where V_N/V_{ref} can be determined from Equation 72. If $V_t=0$, then the power improvement of the multi-processor implementation relative to a uni-processor implementation is:

$$\frac{P_N}{P_1} = \frac{V_N^2}{V_{ref}^2} = \frac{1}{N^2} \quad (\text{EQ 84})$$

This is the theoretical *maximum* reduction in power achievable with architecture driven voltage scaling! This also assumes that leakage currents are zero, which is not a good assumption and will be addressed later.

Figure 3-11 plots the improvement in power as function of N taking into account the limitations to voltage scaling due to finite threshold voltages. As seen from this figure, even assuming ideal conditions for overhead circuitry (where overhead is zero), results in power reductions which saturate as the number of processors is increased for high values of the threshold voltage.

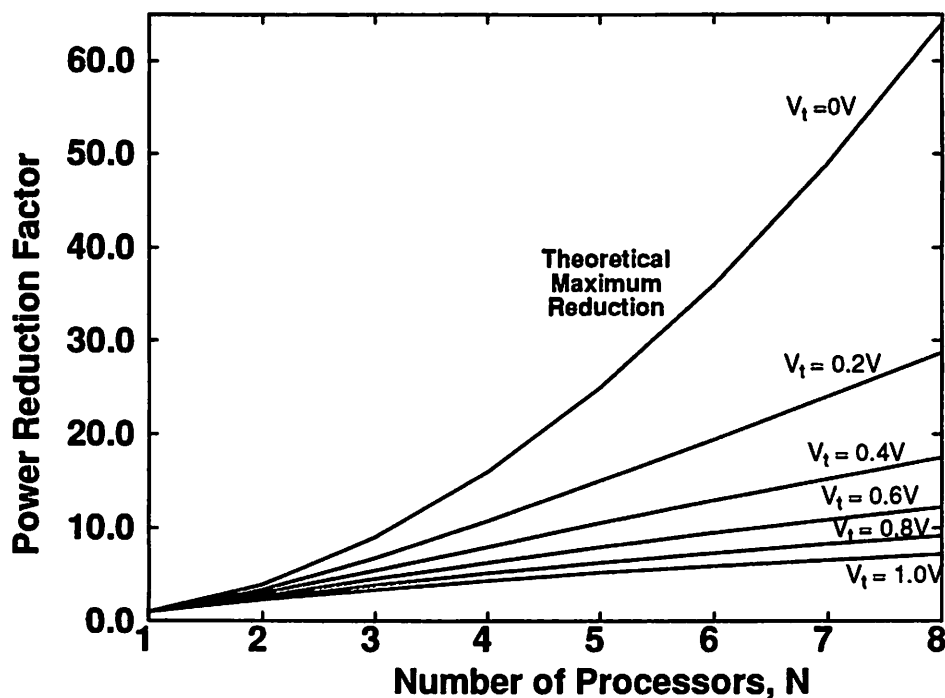


Figure 3-11 : Power reduction (P_1/P_N) vs. number of processors as a function of V_t .

There are two key consequences of architecture driven voltage scaling. First, which is quite obvious is the area increase which will grow faster than the number of processors, and second, is an increase in latency. The latency is increased over the uni-processor implementation and is equal to N cycles. Fortunately, for most signal processing applications like video compression, throughput is the critical metric and latency can be tolerated.

Optimal Supply Voltage for Architecture Driven Voltage Scaling

In the previous section, we saw that the delay increase due to reduced supply voltages below the critical voltage can be compensated by exploiting parallel architectures. However, as seen in Figure 2-26 and Equation 43, as supply voltages approach the device thresholds, the gate delays increase rapidly. Correspondingly, the amount of parallelism and overhead circuitry increases to a point where the added overhead dominates any gains in power reduction from further voltage

reduction, leading to the existence of an “optimal” voltage from an architectural point of view. To determine the value of this voltage, the following model is used for the power for a fixed system throughput as a function of voltage (and hence degree of parallelism) obtained from Equations 75 thru 82.

$$P_N = NC_{ref}V_N^2 \frac{f_s}{N} + C_{in}(N)V_N^2 f_s + C_{out-routing}(N)V_N^2 f_s + C_{mux}(N)V_N^2 f_s \quad (\text{EQ 85})$$

where N is the number of parallel processors, f_s is the sample frequency, C_{ref} is the capacitance switched by a single processor as defined in Equation 67, C_{in} is the capacitance switched on the input bus which is used to broadcast data to all the input registers of the parallel processing elements, $C_{out-routing}$ is the output capacitance switched per output bus (to communicate between the processing elements and the multiplexor in Figure 3-9) which is a function of N since the average interconnect capacitance per bus will increase with the number of processors, C_{mux} is the multiplexor capacitance that is switched at the sample rate. The power improvement of a multiprocessor implementation over a uni-processor implementation (i.e. without parallelism) can be expressed as:

$$P_{normalized} = \left(1 + \frac{C_{in}(N)}{C_{ref}} + \frac{C_{out-routing}(N)}{C_{ref}} + \frac{C_{mux}(N)}{C_{ref}} \right) \left(\frac{V_N}{V_{ref}} \right)^2 \quad (\text{EQ 86})$$

At very low supply voltages (near the device thresholds), the number of processors (and hence the corresponding overhead in the above equation) typically increases at a faster rate than the V^2 term decreases, resulting in a power increase with further reduction in voltage. The minimum on the Power vs. V_{dd} curve is defined as the “optimum” voltage for architecture driven voltage scaling.

Another constraint on the lowest allowable supply voltages is set by system noise margin constraints ($V_{noise\ margin}$). Thus, we must lower-bound the “optimal” voltage by:

$$V_{noise\ margin} \leq V_{optimal} \leq V_{critical} \quad (\text{EQ 87})$$

with $V_{critical}$ defined in Section 3.1.2. Hence, the “optimal” supply voltage (for a fixed technology) will lie somewhere between the voltage set by noise margin constraints and the critical voltage.

As mentioned before, the contribution of overhead circuitry is included to the power consumption, the power vs. number of processors will have an optimum point, parallelizing beyond which will result in an increase of the total power consumption. This results in an optimum level of parallelism and voltage. This can be illustrated by considering only one component of overhead capacitance, the input broadcast capacitance. Therefore, power consumption is given by:

$$P_N = C_{ref} V_N^2 f_{sample} + C_{in}(N) V_N^2 f_{sample} \quad (\text{EQ 88})$$

Let $C_{ref} = 1$ and let $C_{in}(N)$ be modelled as a linear function of N . When $N = 1$, the interconnect capacitance is equal to zero. In general,

$$C_{in}(N) = mN - m \quad (\text{EQ 89})$$

where m is the slope of the interconnect function as a function of the number of processors. Figure 3-12 shows the capacitance as a function of the number of processors.

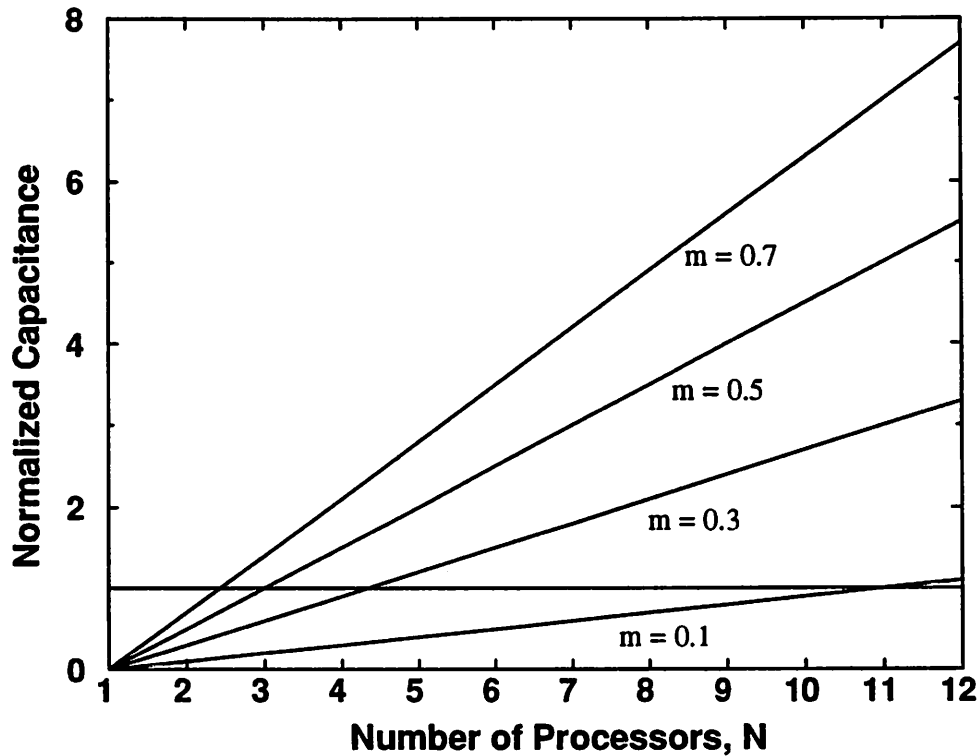


Figure 3-12 : Capacitance function for the processing and input routing overhead.

The power reduction of an N-processor implementation in this case is given by:

$$\frac{P_N}{P_1} = \frac{C_{ref} V_{ref}^2 f_{sample} + C_{in} (N) V_N^2 f_{sample}}{C_{ref} V_{ref}^2 f_{sample}} = \left(1 + \frac{C_{in}}{C_{ref}} \right) \frac{V_N^2}{V_{ref}^2} = \left(1 + C_{in} \right) \frac{V_N^2}{V_{ref}^2} \quad (\text{EQ 90})$$

Figure 3-13 shows a plot of the power as a function of the number of processors, N, for various values of m. All curves shown are for a $V_t=0.8V$. With $m=0$, there is no overhead for parallelism, and the power reduces as $1/N^2$ as derived earlier in Equation 84. For higher values of m, the power reaches minimum at a certain level of parallelism, resulting in an optimum level of parallelism. The minimum point shifts to the left, as the capacitance overhead factor, m, increases.

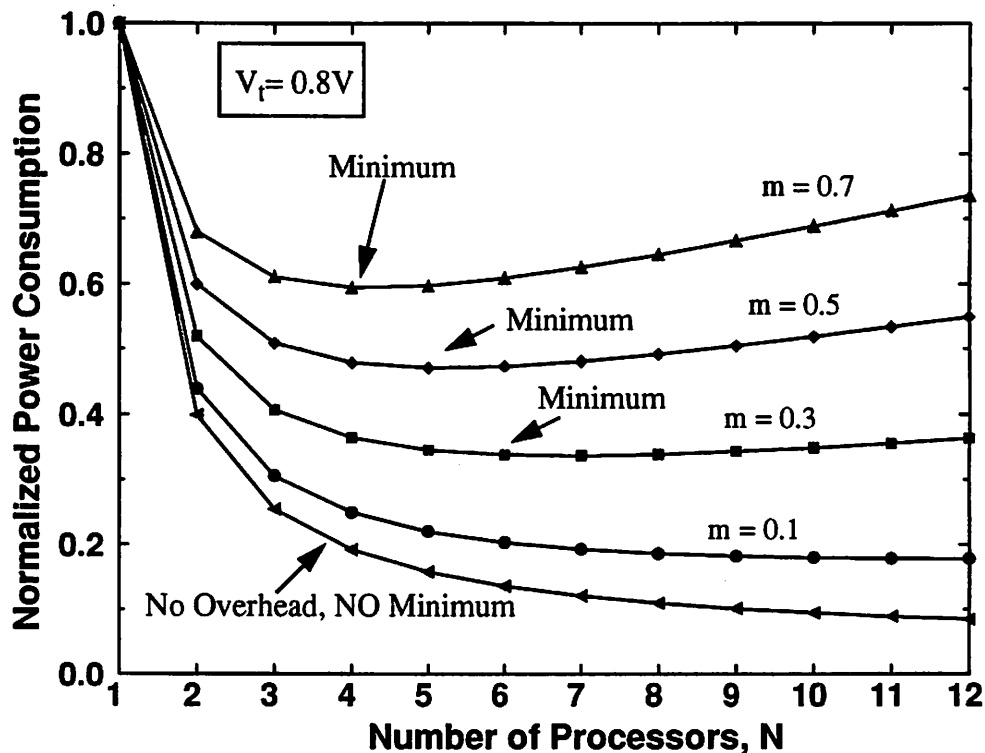


Figure 3-13 : Power vs. N for various levels of overhead capacitance.

Figure 3-14 shows power (normalized to 1 at $V_{dd}=5V$) as a function of V_{dd} for a variety of cases. The lower curve in this figure represents the power dissipation which would be achieved if there were no overhead associated with increased parallelism. For this case, the power is a strictly decreasing function of V_{dd} and the optimum voltage would be set by the minimum value allowed from noise margin constraints. The other curves show the power as a function of the voltage, which is obtained by increasing the level of parallelism. After a certain level of parallelism, the overhead circuitry starts to dominate any gains from voltage reduction and the power starts to increase, and this results in an "optimum" supply voltage for low-power. It is interesting to note all cases have roughly the same optimum value of supply voltage, approximately 1.5V. Even when the overhead is very high ($m=0.5$), the optimal voltage around 2V is found. The intuition for this voltage is that around the sum of the threshold voltages, the delays and therefore the number of processors start to increase rapidly with voltage reduction and this results in a step rise in

overhead circuitry. This rise in overhead dominates and power starts to rise with further reduction of voltage.

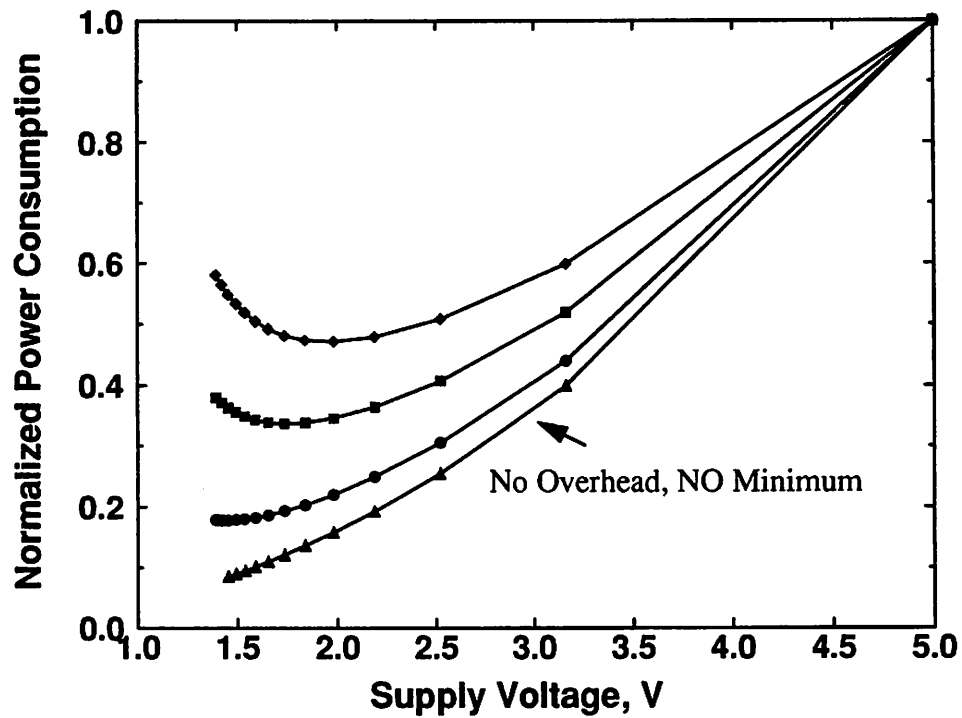


Figure 3-14 : Architecture Driven Optimum Supply Voltage.

Figure 3-15 shows the power as function of number of processors, N , for a fixed value of $m=0.7$, as a function of the threshold voltage. Here, the leakage currents are ignored for reduced threshold voltages. Reducing V_t shifts the minimum point on the curve to the right.

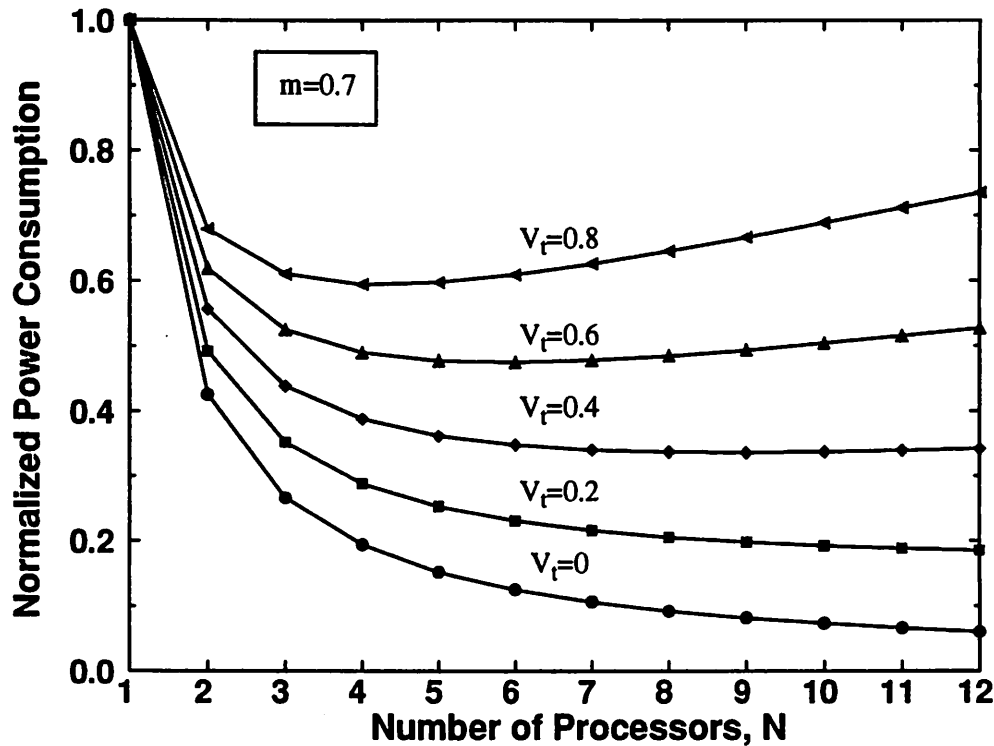


Figure 3-15 : Power vs. N for various threshold voltages.

Trading Area for Lower power through Hardware Pipelining

An architecture driven voltage scaling strategy was presented earlier in which hardware duplication was used to reduce the clock frequency for the processing module which resulted in scaling of the supply voltage. Here another approach will be presented, which involves using pipelining to reduce the critical path (rather than the clock rate) and the supply voltage.

Figure 3-16 shows an N-stage pipelined approach for implementing the generic function shown on the left side. N-1 registers, all clocked at the sample rate, are introduced in the generic function. This has the effect of reducing the critical path. The critical path for an N-stage pipeline is:

$$T_{\text{critical path}} = T_{\text{sample}}/N \quad (\text{EQ 91})$$

Therefore, the logic between the registers could operate N-times slower while maintaining functional throughput. This implies that the supply voltage can be scaled to slow down the circuit

by a factor of N . This voltage at which an N -stage pipelined circuit will have the same throughput as an implementation with no pipelining is derived in Equation 72.

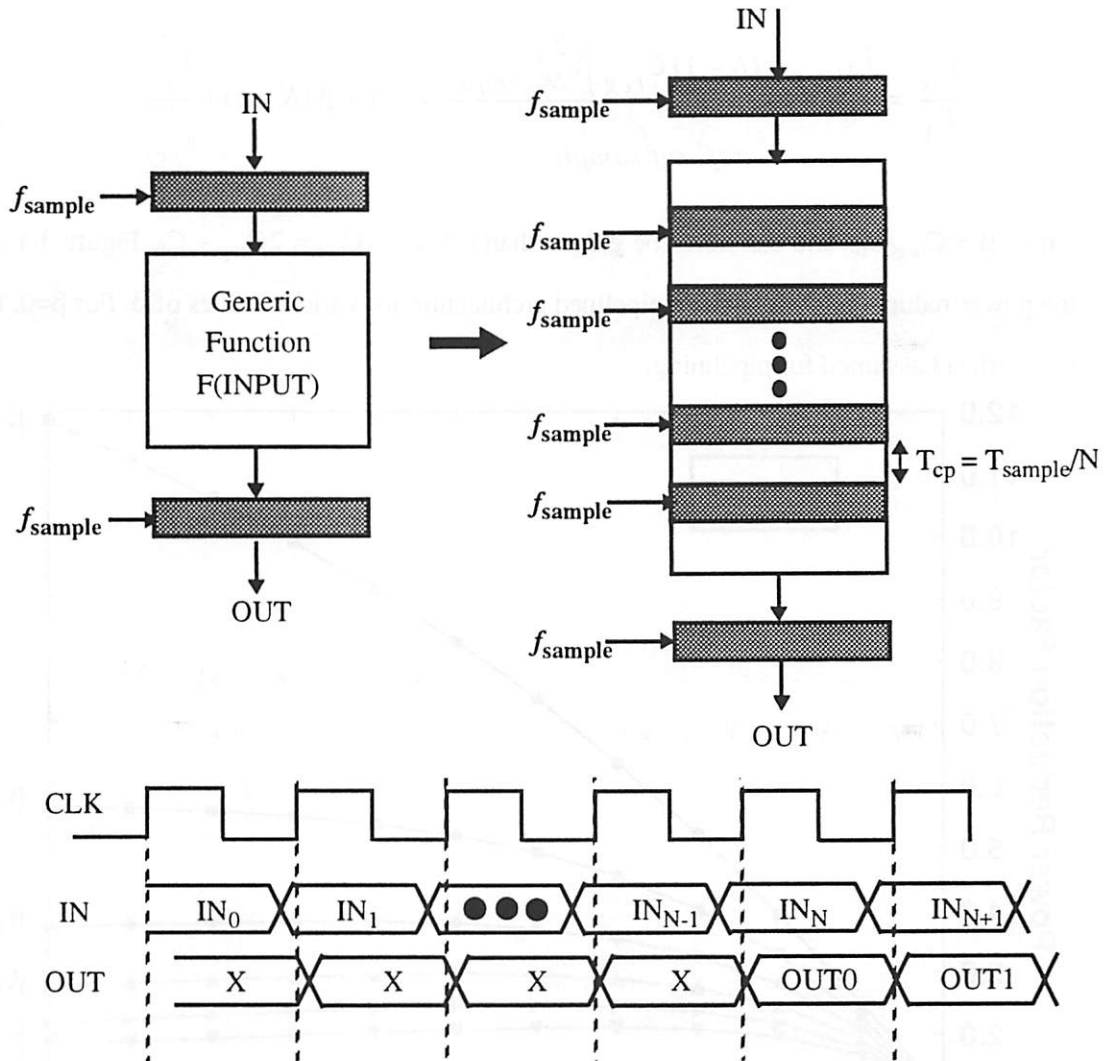


Figure 3-16 : N stage pipelining allows reduction of critical path and voltage.

The power consumption of an N -stage pipelined circuit is approximately given by:

$$P_N = \left(C_{ref} + (N-1)C_{reg} \right) V^2 N f_{sample} \quad (\text{EQ 92})$$

where C_{ref} is the capacitance switched by the original datapath (including the input and output registers) shown on the left of Figure 3-16 and C_{reg} is capacitance switched by each register.

Therefore, the power reduction of a N-stage pipelined implementation versus a non-pipelined implementation is:

$$\frac{P_N}{P_1} = \frac{(C_{ref} + (N-1)C_{reg})V_N^2 f_{sample}}{C_{ref}V_{ref}^2 f_{sample}} = (1 + \beta(N-1)) \frac{V_N^2}{V_{ref}^2} \quad (\text{EQ 93})$$

where $\beta = C_{reg}/C_{ref}$ and can never be greater than 0.5 since $C_{ref} = 2C_{reg} + C_F$. Figure 3-17 shows the power reduction of an N-stage pipelined architecture for various values of β . For $\beta=0$, there is no overhead assumed for pipelining.

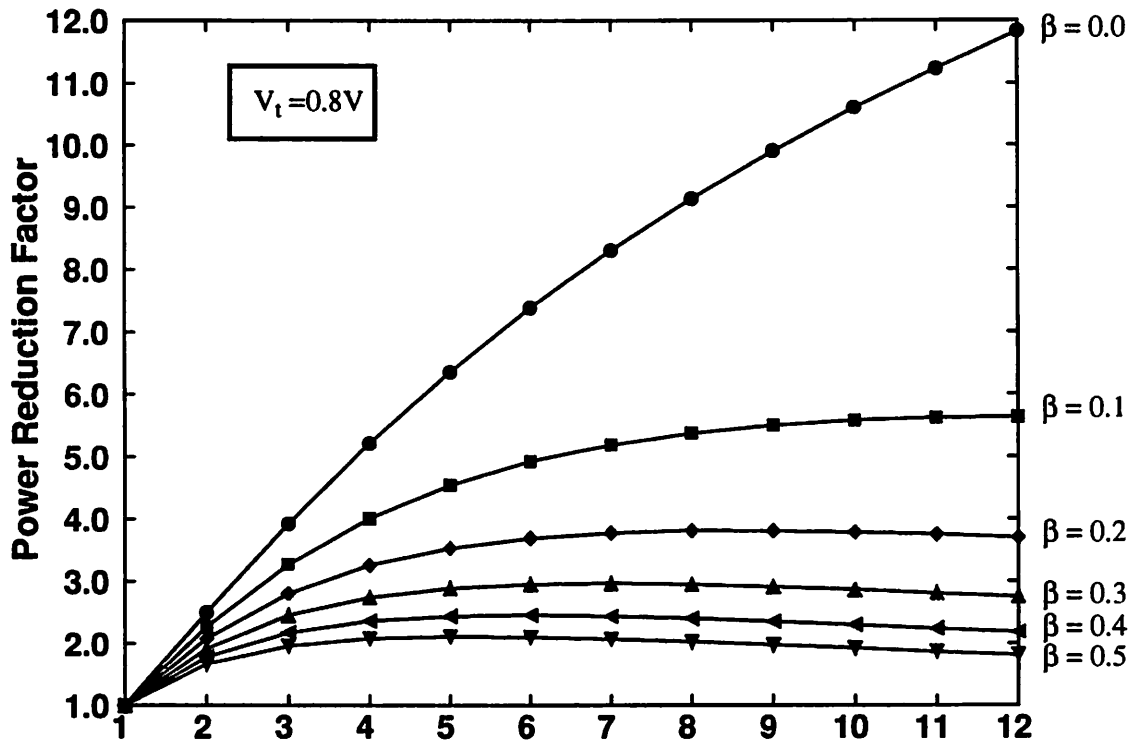


Figure 3-17 : Power reduction for N-stage pipelining for various overhead factors.

Notice that, just like parallelism, the latency is increased for the pipelined implementation and is equal to N clock cycles. However, the overhead for pipelining is much smaller, since only registers have to be added instead of hardware duplication.

Example 1: A Simple Adder-Comparator Example

To illustrate how architectural techniques can be used to compensate for reduced speeds, a simple 8-bit datapath consisting of an adder and a comparator is analyzed assuming a 2.0 μm technology [Chandrakasan92]. As shown in Figure 3-18, inputs A and B are added, and the result compared to input C. Assuming the worst-case delay through the adder, comparator and latch is approximately 25ns at a supply voltage of 5V, the system in the best case can be clocked with a clock period of $T = 25\text{ns}$. When required to run at this maximum possible throughput, it is clear that the operating voltage cannot be reduced any further since no extra delay can be tolerated, hence yielding no reduction in power. We will use this as the reference datapath for our architectural study and present power improvement numbers with respect to this reference. The power for the reference datapath is given by:

$$P_{ref} = C_{ref} V_{ref}^2 f_{ref} \quad (\text{EQ 94})$$

where C_{ref} is the total effective capacitance being switched per clock cycle. The effective capacitance was determined by averaging the energy over a sequence of input patterns with a uniform distribution.

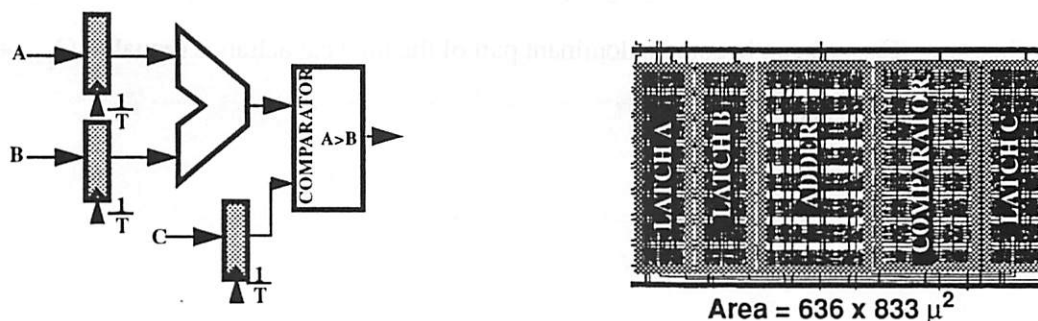


Figure 3-18 : A simple datapath with corresponding layout

One way to maintain throughput while reducing the supply voltage is to utilize a parallel architecture. As shown in Figure 3-19, two identical adder-comparator datapaths are used, allowing each unit to work at half the original rate while maintaining the original throughput. Since the speed requirements for the adder, comparator, and latch have decreased from 25ns to 50ns, the voltage can be dropped from 5V to 2.9V (the voltage at which the delay doubled, from Figure 2-26). While the datapath capacitance has increased by a factor of 2, the operating frequency has correspondingly decreased by a factor of 2. Unfortunately, there is also a slight increase in the total “effective” capacitance introduced due to the extra routing, resulting in an increased capacitance by a factor of 2.15. Thus the power for the parallel datapath is given by:

$$P_{par} = C_{par} V_{par}^2 f_{par} = (2.15C_{ref}) (0.58V_{ref})^2 \left(\frac{f_{ref}}{2}\right) \approx 0.36P_{ref} \quad (\text{EQ 95})$$

This method of reducing power by using parallelism has the overhead of increased area, and would not be suitable for area-constrained designs. In general, parallelism will have the overhead of extra routing (and hence extra power), and careful optimization must be performed to minimize this overhead (for example, partitioning techniques for minimal overhead). As described in Chapter 2, interconnect capacitance will especially play a very important role in deep sub-micron implementations, since the fringing capacitance of the interconnect capacitance ($C_{wiring} = C_{area} + C_{fringing} + C_{wiring}$) can become a dominant part of the total capacitance (equal to $C_{gate} + C_{junction} + C_{wiring}$) and cease to scale.

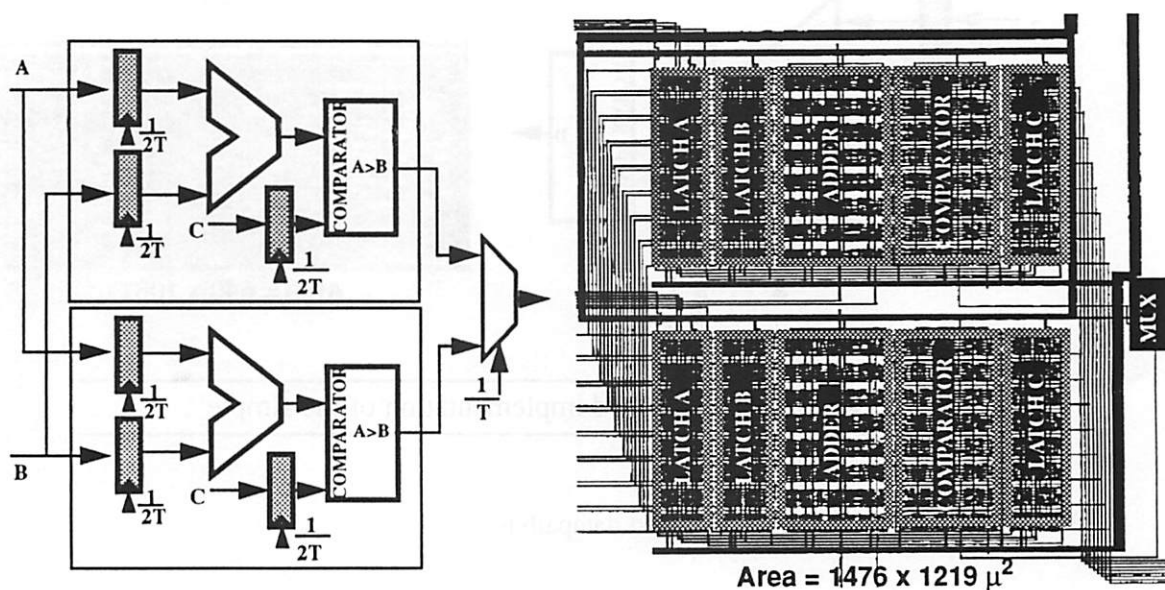


Figure 3-19 : Parallel implementation of the simple datapath

Another possible approach is to apply pipelining to the architecture, as shown in Figure 3-20. With the additional pipeline latch, the critical path becomes the $\max[T_{\text{adder}}, T_{\text{comparator}}]$, allowing the adder and the comparator to operate at a slower rate. For this example, the two delays are equal, allowing the supply voltage to again be reduced from 5V used in the reference datapath to 2.9V (the voltage at which the delay doubles) with no loss in throughput. However, there is a much lower area overhead incurred by this technique, as we only need to add pipeline registers. Note that there is again a slight increase in hardware due to the extra latches, increasing the “effective” capacitance by approximately a factor of 1.15.

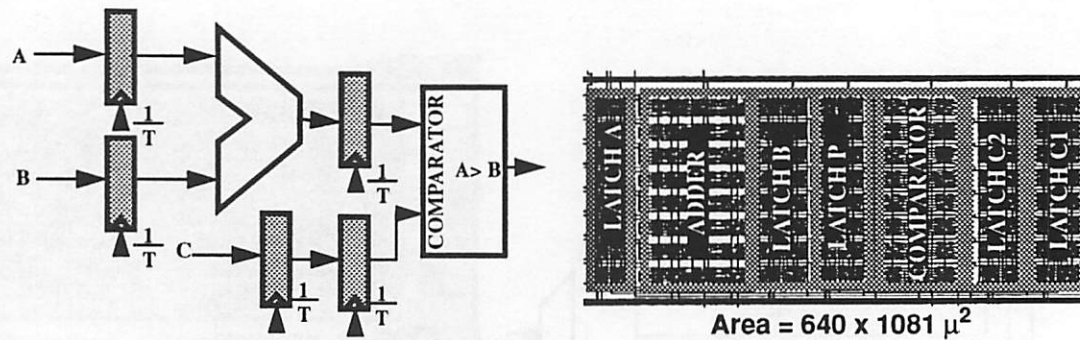


Figure 3-20 : Pipelined implementation of the simple

The power consumed by the pipelined datapath is:

$$P_{pipe} = C_{pipe} V_{pipe}^2 f_{pipe} = (1.15C_{ref}) (0.58V_{ref})^2 f_{ref} \approx 0.39 P_{ref} \quad (\text{EQ 96})$$

With this architecture, power reduces by a factor of approximately 2.5, providing approximately the same power reduction as the parallel case with the advantage of lower area overhead. As an added bonus, increasing the level of pipelining also has the effect of reducing logic depth and hence power contributed due to hazards and critical races.

Furthermore, an obvious extension is to utilize a combination of pipelining and parallelism. Since this architecture reduces the critical path and hence speed requirement by a factor of 4, the voltage can be dropped until the delay increases by a factor of 4. The power consumption in this case is:

$$P_{parpipe} = C_{parpipe} V_{parpipe}^2 f_{parpipe} = (2.5C_{ref}) (0.4V_{ref})^2 \left(\frac{f_{ref}}{2}\right) \approx 0.2P_{ref} \quad (\text{EQ 97})$$

The parallel-pipeline implementation results in a 5 times reduction in power. Table 3-1 shows a comparative summary of the various architectures described for the simple adder-comparator datapath.

Table 3-1 : Results of architecture based voltage scaling

Architecture	Voltage	Area (normalized)	Power (normalized)
Simple	5V	1	1
Parallel	2.9V	3.4	0.36
Pipelined	2.9V	1.3	0.39
Pipelined-Parallel	2.0	3.7	0.2

Figure 3-21 shows the power vs. V_{dd} for the parallel and parallel-pipelined datapaths. Curve 1 in this figure represents the power dissipation which would be achieved if there were no overhead associated with increased parallelism. For this case, the power is a strictly decreasing function of V_{dd} and the optimum voltage would be set by the minimum value allowed from noise margin constraints (assuming that no recursive bottleneck was reached). Curves 2 and 3 are obtained from data from actual layouts and are extensions of the example described earlier in which parallel and parallel-pipeline implementations of the simple datapath were duplicated N times.

In Table 3-2 is a summary of the power reduction and normalized areas that were obtained from layouts. The increase in areas gives an indication of the amount of parallelism being exploited. The key point is that the optimal voltage is found to be relatively independent over the cases considered, and occurred around 1.5V for the 2.0μ technology (with $V_{tn} = 0.7V$ and $V_{tp} = -0.9V$); a similar analysis using a 0.5V threshold, 0.8μ process (with an L_{eff} of 0.5μ) resulted in optimal voltages around 1V, with power reductions in excess of a factor of 10. Further scaling of the threshold would allow even lower voltage operation, and hence even greater power savings.

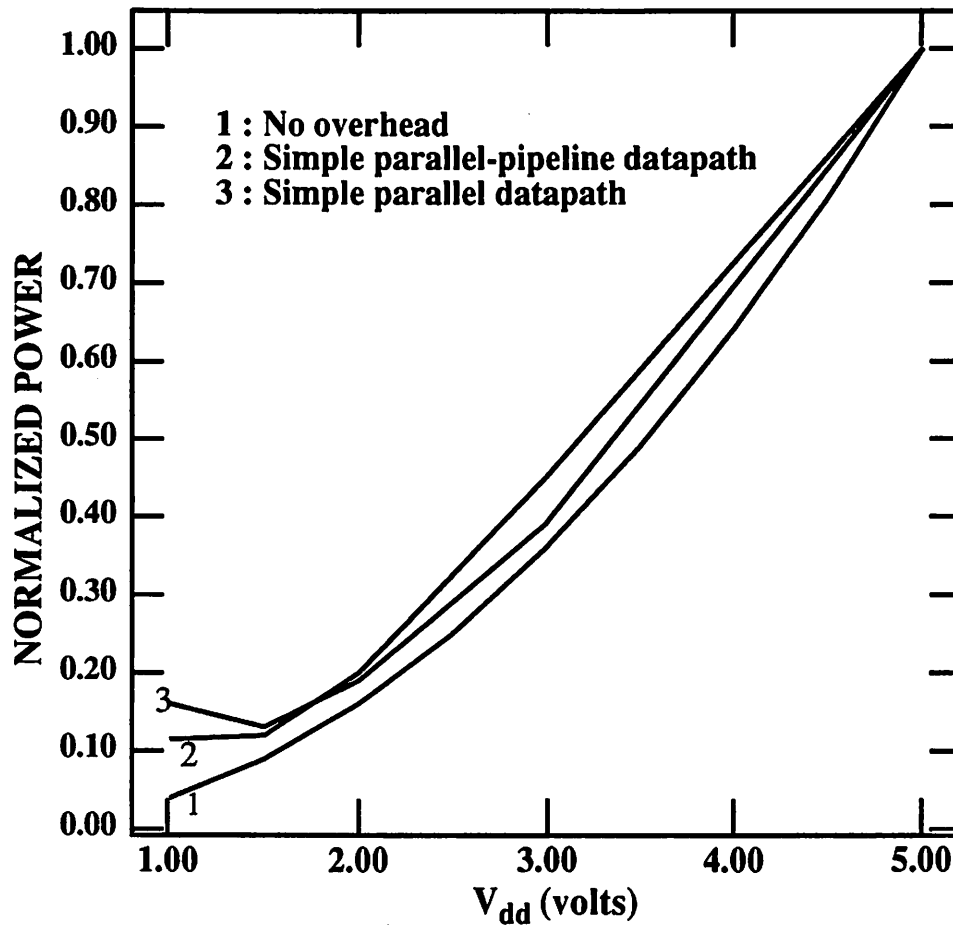


Figure 3-21 : Optimum voltage of operation for the add/compare computation.

Table 3-2 : Normalized Area/Power for various supply voltages for Plots 2,3 in Figure3-21

Voltage	Parallel Area/Power	Parallel -Pipelined Area/Power
5	1 / 1	1 / 1
2	6 / 0.19	3.7 / 0.2
1.5	11 / 0.13	7 / 0.12
1.4	15 / 0.14	10 / 0.11

From the above examples, it is clear that the traditional time-multiplexed architectures, as used in general purpose microprocessors and DSP chips are the least desirable for low-power applications.

This follows since time multiplexing actually increases the speed requirements on the logic circuitry, thus not allowing reduction in the supply voltage.

Example 2: Exploiting Concurrency for Applications with Feedback

So far, we have seen that parallel and pipelined architectures can allow for a reduction in supply voltages to the “optimal” level; this will indeed be the case if the algorithm being implemented does not require feedback. For example, in the simple adder-comparator example which was presented, parallel and pipelined architectures were used to reduce the supply voltage and power. This simple example did not have any feedback and the computation was therefore easily parallelizable. However, there are a wide class of applications inherently recursive in nature, ranging from simple ones, such as infinite impulse response and adaptive filters, to more complex cases such as systems solving non-linear equations and adaptive compression algorithms. There is, therefore, an algorithmic bound on the level to which pipelining and parallelism can be exploited for voltage reduction.

In applications that have feedback (e.g IIR), the computation cannot be easily parallelized and algorithmic transformations are required to alleviate the recursive bottlenecks. In general, exploiting concurrency in an algorithm is not a trivial task and high-level computation transformations are required to exploit concurrency. Many transformations profoundly affect the amount of concurrency in the computation. This includes retiming/pipelining, algebraic transformations and loop transformations. For linear systems, a sequence of computational transformations applied in well defined order can be used to exploit arbitrary parallelism operate circuits at very low supply voltages.

To illustrate the application of speed-up transformations to lower power, consider a first order IIR filter, as shown in Figure 3-22a. A time-multiplexed model is assumed in which each operation is assumed to take once control cycle and therefore this structure has a critical path of 2 (i.e for this example, $T_{\text{sample}}/T_{\text{clock}} = 2$). The goal is to reduce the number of cycles to process one sample

since this will allow a reduction of the power supply. Due to the recursive bottleneck [Messerschmitt88] [Parhi89] imposed by the filter structure, it is impossible to reduce the critical path using retiming or pipelining. Also, the simple structure does not provide opportunities for the application of algebraic transformations and applying a single transformation is not enough to reduce power in this example.

Figure 3-22b shows the effect of applying loop unrolling (where two output samples are computed in parallel based on two input samples) on the initial flowgraph.

$$Y_{N-1} = X_{N-1} + A * Y_{N-2} \quad (\text{EQ 98})$$

$$Y_N = X_N + A * Y_{N-1} = X_N + A * (X_{N-1} + A * Y_{N-2}) \quad (\text{EQ 99})$$

The critical path is doubled, but two samples are processed in parallel; therefore, the effective critical path is unchanged and therefore the supply voltage cannot be altered. The effective capacitance switched does not change either since the number of operations is doubled for processing two input samples in parallel. Since neither the capacitance switched nor the voltage is altered, the power of this implementation remains unchanged. Therefore, Loop unrolling by itself does not reduce the power consumption.

However, loop unrolling enables several other transformations (distributivity, constant propagation, and pipelining) which result in a significant reduction in power dissipation. After applying loop unrolling, distributivity and constant propagation in a systematic way, the output samples can be represented as:

$$Y_{N-1} = X_{N-1} + A * Y_{N-2} \quad (\text{EQ 100})$$

$$Y_N = X_N + A * X_{N-1} + A^2 * Y_{N-2} \quad (\text{EQ 101})$$

The transformed solution has a critical path of 3 (Figure 3-22c) for processing two samples. Therefore, the effective critical path is reduced and therefore the voltage can be dropped. This results in a small reduction of the power consumption (by 20%).

However, pipelining can now be applied to this structure, reducing the critical path further to 2

cycles (Figure 3-22d). Since the final transformed block is working at half the original sample rate (since we are processing 2 samples in parallel), and the critical path is same as the original datapath (2 control cycles), the supply voltage can be dropped to 2.9V (the voltage at which the delays increase by a factor of 2, see Figure 2-26). However, note that the effective capacitance increases since the transformed graph requires 3 multiplications and 3 additions for processing 2 samples while the initial graph requires only one multiplication and one addition to process one sample, or effectively a 50% increase in capacitance. The reduction in supply voltage, however, more than compensates for the increase in capacitance resulting in an overall reduction of the power by a factor of 2 (due to the quadratic effect of voltage on power).

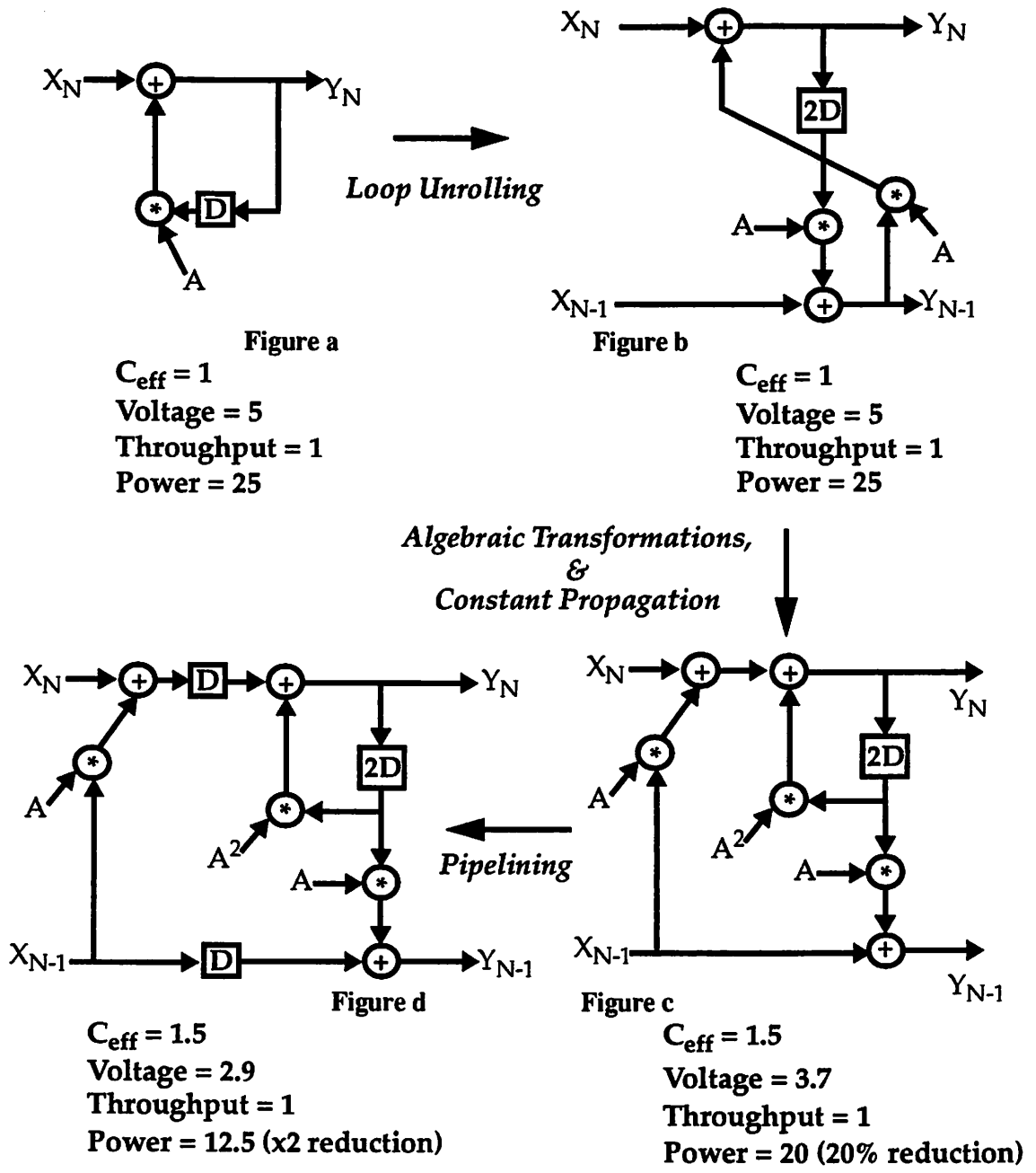


Figure 3-22 : Using speedup transformations to reduce power.

This simple example can be used to illustrate that optimizing for throughput will result in *different* solution than optimizing for power. For this example, arbitrary speedup can be achieved by continuing to apply loop unrolling combined with other transformations (algebraic, constant

propagation, and pipelining). The speedup grows linearly with the unrolling factor, as shown in Figure 3-23. If the goal is to minimize power consumption while keeping the throughput fixed, the speedup can be used to drop the supply voltage. Unfortunately, the capacitance grows linearly with unrolling factor (since the number of operations per input sample increases) and soon limits the gains from reducing the supply voltage. This results in an “optimum” unrolling factor for power of 3, beyond which the power consumption starts to increase again.

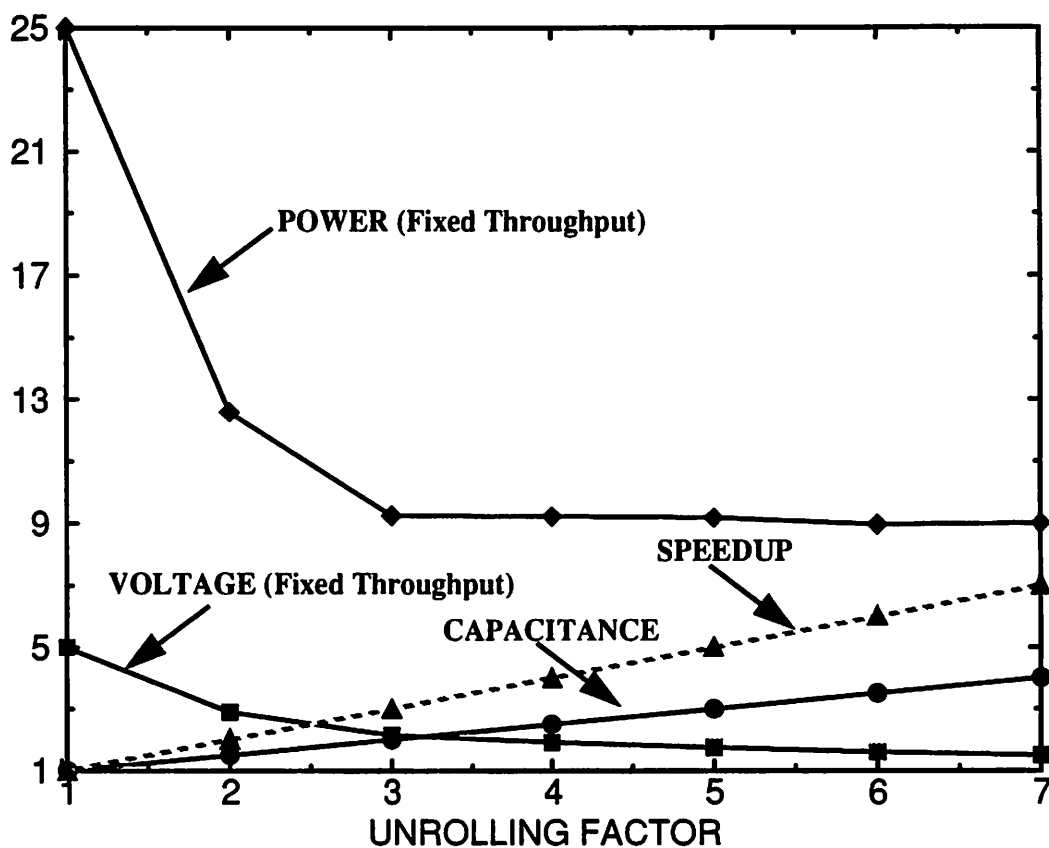


Figure 3-23 : Speed optimizing is *different* than power optimization.

This example brings out two very important points: First, the application of a particular transformation can have conflicting effects on the different components of power consumption. For example, a transformation can reduce the voltage component of power (through a reduction in the critical path) while simultaneously increasing the capacitance component of power. Therefore, while speed-up transformations can be used to reduce power by allowing for reduced supply

voltages, the “fastest” solution is often NOT the lowest power solution. The design environment described later in Chapter 5, can be used to automatically explore the design space using speed-up transformations (as described above) and find the “optimum” solution which trade’s off capacitance and voltage for a fixed throughput. Second, the application of transformations in a combined fashion almost always results in lower power consumption than the isolated application of a single transformation. In fact, it is often necessary to apply a transformation which may temporary increase the power budget, in order to enable the application of transformations which will result in a more dramatic power reduction.

A simple first order IIR filter was presented above to demonstrate that the optimization for throughput is very different than optimization for power. Now consider a bigger and more widely used example, the DCT (Discrete Cosine Transform), to illustrate this point further. We will compare two implementation of the DCT: one proposed by Feig and Winograd [Feig92] and the other, the direct maximally fast form. Feig’s DCT algorithm can be derived from the direct form using the exceptionally sophisticated application of common subexpression elimination/replication and algebraic rules. The direct form, on other hand, can be derived from the Feig’s DCT, by the simple application of the transformation set using the procedure for maximally fast implementation of linear computation [Potkonjak92]. While Feig’s DCT has a critical path of 11 cycles, the maximally fast DCT has a critical path of only 7 cycles. Therefore the V_{dd} can be reduced from 5 V to 3.25 V. While the reduction of the supply was relatively significant, the effective capacitance increased at a rate larger than an order of magnitude. Feig’s form has 54 multiplications and 462 addition while the maximally fast form has 4,096 multiplication and 4,096 additions. The power increases by a factor of more than 20, even after taking into account the voltage reduction due to throughput improvement.

The examples presented in this subsection clearly indicate that there is a sharp difference in the objective function for throughput and power. A number of key transformations for throughput enhancement have the effect of increasing the number of operations or increasing the interconnect

area. For example, it has been shown that common subexpression can yield, for many classes of designs implementation, arbitrarily high throughput at the expense of additional number of operations [Potkonjak92].

3.1.6 Voltage Scaling using Threshold Reduction

From the previous section, it is clear that the optimum voltage is reached when the overhead capacitance from parallelism or pipelining dominates at supply voltages close to the threshold voltage of the devices. Therefore, a key to reducing the “optimum” supply voltage is reducing the threshold voltage of the devices. Reducing the threshold voltage allows the supply voltage to be scaled down (and therefore lower switching power) without loss in speed. For example, a circuit running at a supply voltage of 1.5V with $V_t=1V$ will have approximately the same performance as the circuit running at a supply voltage of 0.9V and a $V_t=0.5V$ using the simple first order theory of Section 2.1.6. Figure 3-24 shows a plot of normalized delay vs. threshold voltage for various supply voltages.

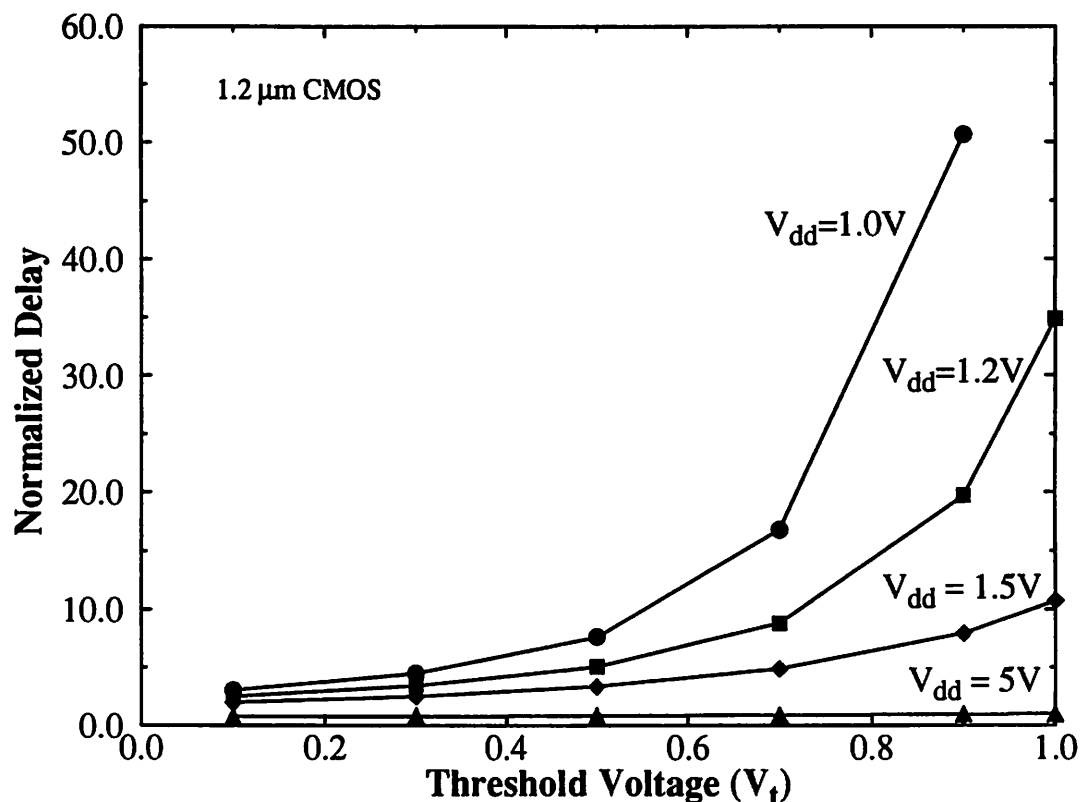


Figure 3-24 : Effect of threshold reduction on the delay for various supply voltages.

Since significant power improvements can be gained through the use of low-threshold MOS devices, the question of how low the thresholds can be reduced must be addressed. The limit is set by the requirement to retain adequate noise margins and the increase in subthreshold currents. Noise margins will be relaxed in low power designs because of the reduced currents being switched, however, subthreshold currents can result in significant static power dissipation.

Figure 3-25 shows a plot of energy vs. threshold voltages for a fixed throughput for a 16-bit datapath ripple carry adder (which essentially represents the power to perform the operation). Here, the power supply voltage is allowed to vary to keep the throughput fixed. For a fixed throughput (e.g. that obtained at a 20Mhz clock rate), the supply voltage and therefore the switching component of power can be reduced while reducing the threshold voltage. However, at some point, the threshold voltage and supply reduction is offset by an increase in the leakage currents, resulting in an optimal threshold voltage for a given level of logic complexity. That is,

the optimum threshold voltage must compromise between improvement of current drive at low supply voltage operation and control of the sub-threshold leakage.

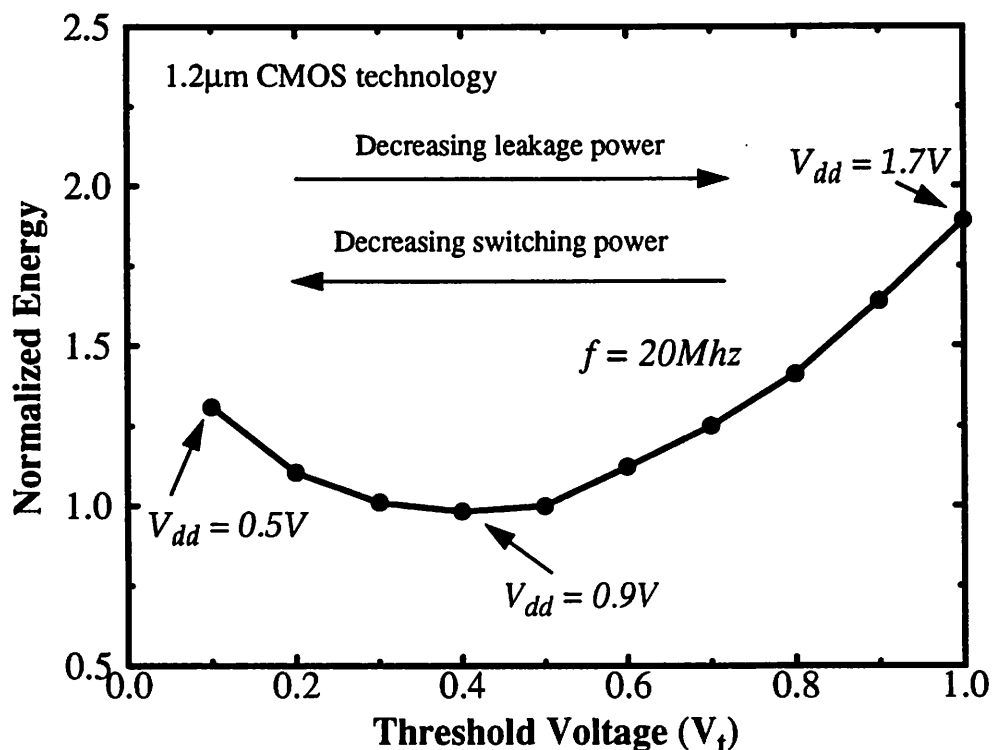


Figure 3-25 : Compromise between dynamic and leakage power dissipation through V_t variation.

3.2 Support Circuitry for Low-voltage Operation

3.2.1 High Efficiency Power Supply Generation [Stratakos94]

The architecture driven voltage scaling strategy that was presented earlier assumes that the supply voltage is a free variable and more importantly that the voltage can be set to any arbitrary level with very high efficiency. In order to realize such portable systems in which different parts of the system could operate at their own “optimum” supply voltage and communicate with each other using the level-conversion circuitry that will be described in the next section, the design of high efficiency low-voltage (in which the voltage can be made programmable) switching regulators

must be considered. Figure 3-26 shows the schematic of a Buck regulator that can be used to generate low-supply voltages with high efficiency.

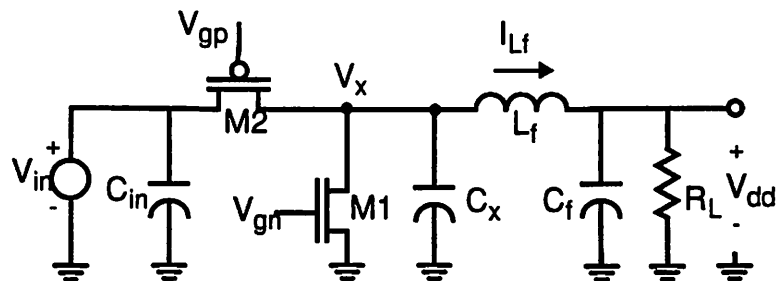


Figure 3-26 : A Buck regulator that be used to generate arbitrary voltages.

The converter works by chopping the input voltage (through control of V_{gn} and V_{gp}) to reduce the average voltage. This produces a square-wave of duty cycle D with frequency f_s at the inverter output node V_x . The chopped signal is filtered by the second-order low-pass filter (L_f and C_f) to reduce the ac component to an acceptable ripple value. Buck converters are capable of a 0% to 100% duty factor and the output voltage is given as:

$$V_{dd} = V_{in} * D \quad (\text{EQ 102})$$

where V_{in} is the unregulated input voltage and D is the duty factor. For example, with $V_{in} = 6\text{V}$ and $D = 25\%$, the output voltage is 1.5V. The output can be set to an arbitrary value by controlling the duty cycle on the inverter output node V_x .

There are three main sources of dissipation that cause the conversion efficiency of this circuit to be less than unity: switching loss ($C_x V_{in}^2 f_s$ loss due to parasitics), conduction loss ($I^2 R$ loss in the power transistors and filter components), and $C_g V_{in}^2 f_s$ gate drive loss for each FET. Four techniques are used to improve the efficiency of the converter at reduced voltages and current levels.

Synchronous Rectification

The free-wheeling diode used in conventional high voltage converters is replaced with a gated

NMOS device (M1). For low output voltage levels, the diode's voltage drop becomes a significant fraction of the output voltage, leading to large conduction loss. A gated NMOS device can achieve the same function more efficiently.

Soft Switching or Zero Voltage Switching of Power Devices

Capacitive switching loss is nearly eliminated by using the filter inductor as a current source to charge and discharge the inverter node; therefore, the power transistors are turned on and off at $V_{DS}=0$ (referred to as Zero Voltage Switching or ZVS). This factor is especially important for high-frequency converters. Figure 3-27 shows the timing for soft-switching of the Buck converter. Soft-switching is accomplished by individual control of the gate voltages of the NMOS and PMOS devices such that there is a dead-time when neither transistors conducts. Consider the 0 \rightarrow 1 transition at node x which is initiated by turning off the NMOS device. By choosing the value of the inductor to be small enough, the ripple current can be made larger than the average current, \bar{I}_{out} , and therefore the current on the inductor reverses direction. The reverse inductor current is then used to move charge from the output filter capacitor to the parasitic capacitance at node x. When node x is equal to V_{in} , the PMOS device is turned on with zero drain to source voltage, eliminating switching loss.

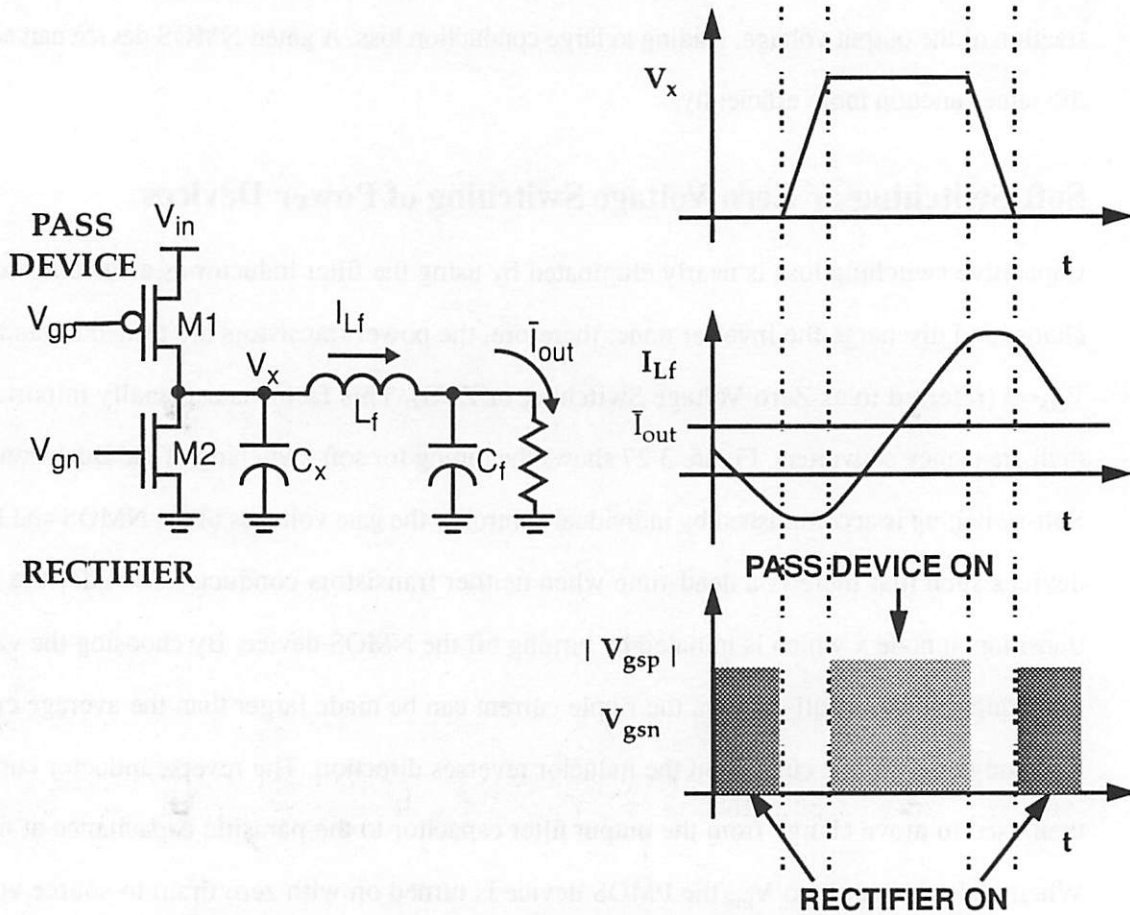


Figure 3-27 : Timing for soft-switching of the Buck converter.

Adaptive Dead Time Control

Zero Voltage Switching (or soft-switching) depends critically on the dead-time, during which neither M1 nor M2 conducts. A fixed dead-time designed assuming an average current load can result in significant losses if operating conditions change.

Figure 3-28 shows the effect of varying load conditions of the waveforms for a high to low transition at node x. When the average load current decreases, as shown on the left, the current to discharge C_x is lower and therefore the discharge time is increased. For a dead-time implementation, the NMOS would turn on before the node x has discharged and would discharge

node x. When the average current increases, the current to discharge the capacitance C_x is larger and the discharge occurs faster. This will cause the source to body diode of the drain to turn on for the NMOS device and causes forward-diode conduction, which is very inefficient for low-voltage operation.

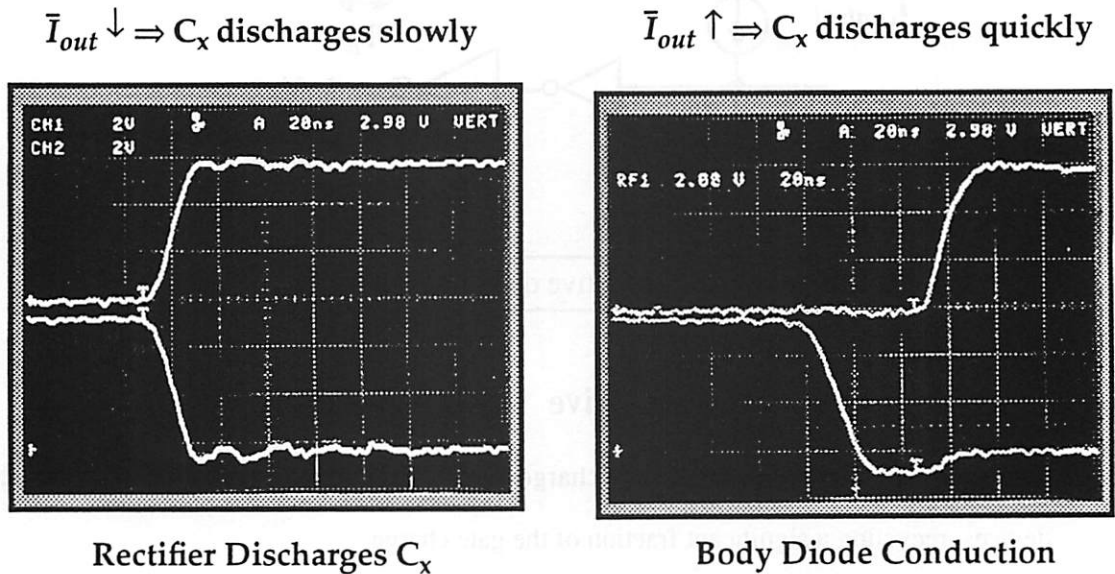


Figure 3-28 : Effects of varying load conditions on waveforms.

From Figure 3-28, it is clear that an adaptive dead-time control through a negative feedback loop is needed to accommodate varying operating conditions and process variations. Figure 3-29 shows the schematic for the dead-time control for the NMOS device. On a clock (CLK) low to high transition, the switch shown in the figure turns on shorting the internal node (int) to ground and this causes the buffered output, which drives the gate of the NMOS device, to go low. On the clock high to low transition, the switch is open and the turn on of the NMOS gate is delayed. The delay is set by setting $I_{control}$ (which is determined from the relative phases of V_x and V_{gn}). This circuit provides zero voltage switching over a wide range of loads [Stratakos94].

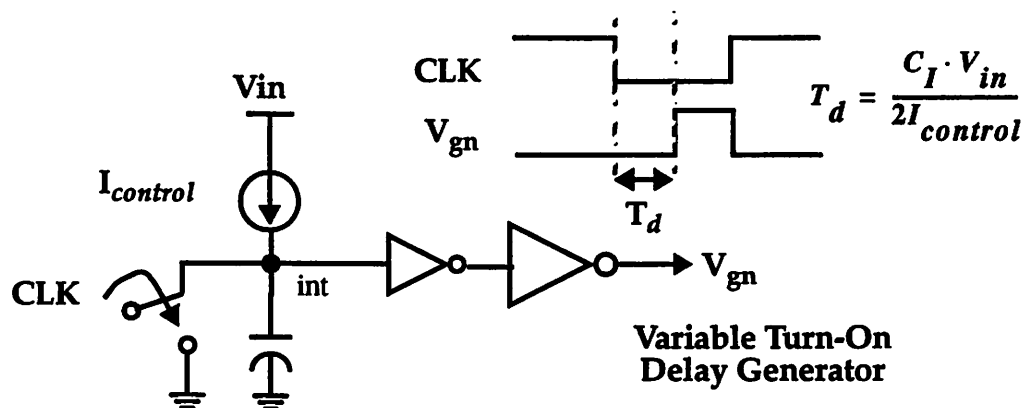


Figure 3-29 : Adaptive dead-time control.

Resonant-transition Gate-drive

A resonant inductor can be used to charge and discharge the gate capacitances of the power devices, recycling a significant fraction of the gate charge.

Transistor Sizing for Minimizing Overall Losses

The switching loss is nearly eliminated using soft-switching and adaptive dead-time control. An important design trade-off involves the selection of the power transistor sizes such that the total losses are minimized. That is, minimize:

$$P_{\text{total}} = P_{\text{conduction loss}} + P_{\text{gate-drive loss}} \quad (\text{EQ 103})$$

The conduction loss is proportional to the resistance of the power transistors and therefore increasing the width will decrease this component. The gate drive loss is proportional to the gate capacitance of the devices and therefore increasing the width will linearly increase this loss. Figure 3-30 shows a plot of the two components of loss and the total power. Through simple algebraic manipulation, the optimum width can be determined to be:

$$W_{\text{opt}} = \sqrt{\frac{b}{a \cdot f_s}} \quad (\text{EQ 104})$$

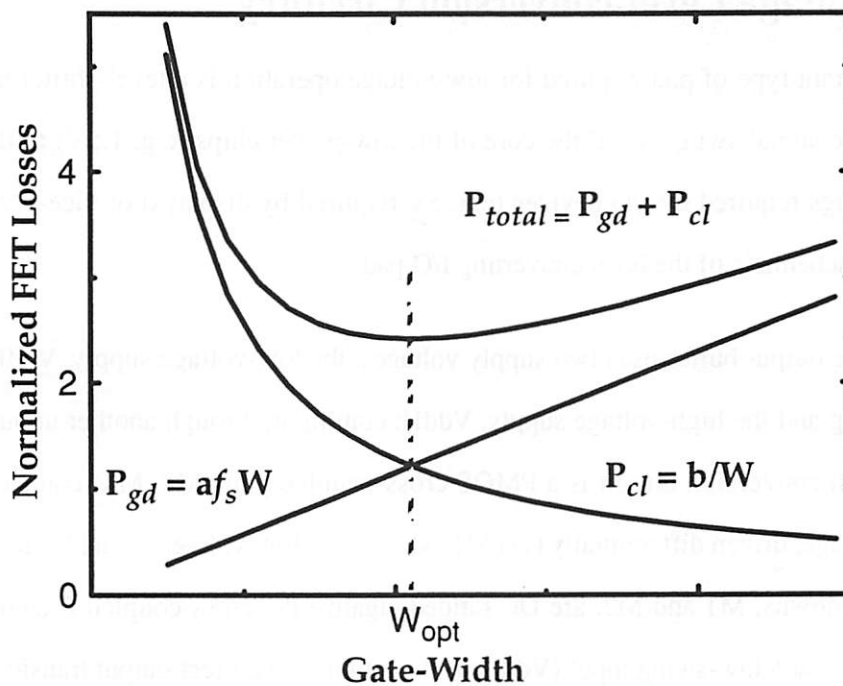
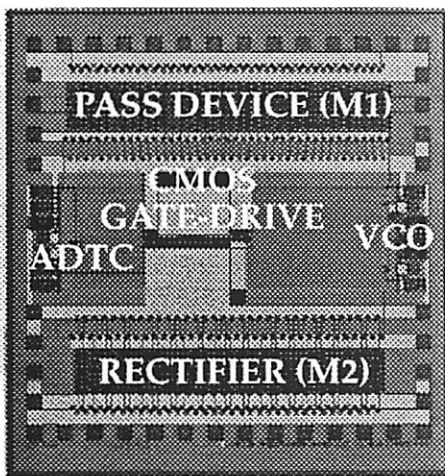


Figure 3-30 : Optimum sizing of power transistors.

Figure 3-31 shows the die photo of a prototype 1.5V regulator that achieves > 90% efficiency.



Frequency: 1Mhz
Battery Voltage: 6V
Output Voltage: 1.5V
Output Power: 750mW
Efficiency: 92%
Technology: 1.2 μ m
M1: 10.2cm / 1.2 μ m
M2: 10.5cm / 1.2 μ m
Size: 4.2mm X 4.2mm

Figure 3-31 : Die photo of a 1.5V Buck regulator.

3.2.2 Voltage Level-conversion Circuitry

One important type of pad required for low-voltage operation is a level-shifter that can convert low-voltage signal swings from the core of the low-power chips (e.g. 1.5V) to the high-voltage signal swings required by I/O devices (e.g. 5V required by displays) or vice-versa. Figure 3-32 shows the schematic of the level converting I/O pad.

This tristate output buffer uses two supply voltages, the low-voltage supply, V_{ddL} , that is tied to the pad-ring and the high-voltage supply, V_{ddH} , coming in through another unbuffered pad. The low-to-high conversion circuit is a PMOS cross-coupled pair (M3, M4) connected to the high supply voltage, driven differentially (via M1, M2) by the low-voltage signal from the core. The N-device pulldowns, M1 and M2, are DC ratioed against the cross-coupled P-device pullups, M3 and M4, so that a low-swing input ($V_{ddL}=1V$) guarantees a correct output transition ($V_{ddH}=5V$). That is, the PMOS widths are sized so that the drive capability of the NMOS can overpower the drive of the PMOS, and reverse the state of the latch. The ratio is larger than just the ratio of mobilities, because the PMOS devices are operating with $V_{GS}=V_{ddH}$, and the NMOS is operating with $V_{GS}=V_{ddL}$. This level-converting pad consumes power only during transitions and consumes no DC power. The remaining buffer stages and output driver are supplied by V_{ddH} . This level-conversion pad will work only with an NWEELL process since it requires isolated wells for the PMOS devices that are connected to the high voltage.

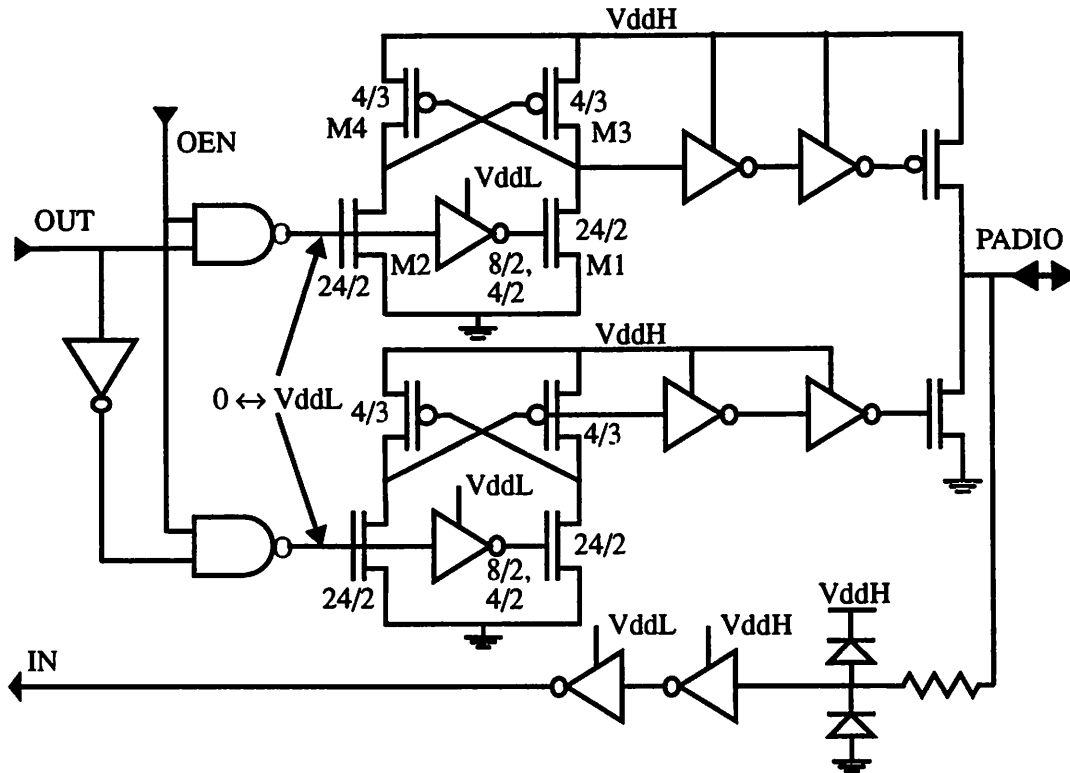


Figure 3-32 : Level-conversion I/O pad buffer.

3.3 Noise Considerations at Reduced Supply Voltages

Reducing the supply voltage raises some issues on noise immunity. There are three important sources of internal noise in integrated circuits: inductive noise due to the simultaneous switching of circuits, resistive noise on the power supply lines, and coupling between neighboring circuits. In this section, the effect of supply voltage scaling on the various components of noise will be analyzed.

Bond wires, package pins, printed circuit board traces, and IC traces all have parasitic inductances. When a CMOS gate switches, a current spike flows through the power bus, which has parasitic inductance from the above mentioned sources. When current changes through an inductor, a voltage is generated across the inductor which is equal to $L \cdot di/dt$. As a result of this voltage drop,

there is a fluctuation in the internal power supply, which can cause soft-errors if the voltage drop is large. This effect is commonly referred to as simultaneously switching noise since it is most pronounced when many circuits (especially many off-chip drivers switch simultaneously). Figure 3-33 is a model for including bond inductance for computing the internal supply voltage.

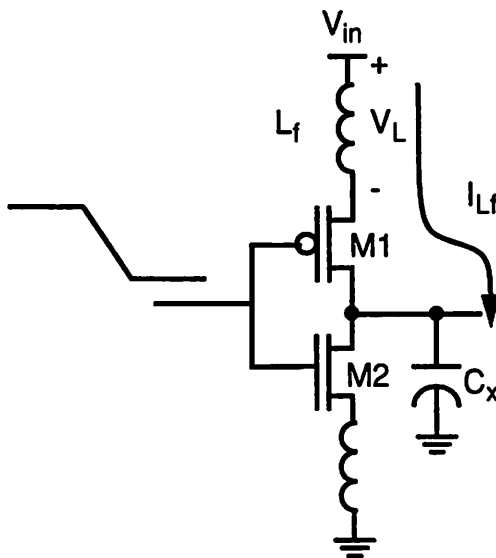


Figure 3-33 : Inductive noise on the power supplies.

Ignoring the threshold effects on current (i.e. assuming that I is proportional to V_{dd}^2 and delay is inversely proportional to V_{dd}), we see that the di/dt scales as V_{dd}^3 (even faster when threshold effects are taken into account), and hence drastically reducing the voltage noise. However, as mentioned earlier, reducing the voltage will require more parallelism (to meet throughput requirements) and will hence increase the number of simultaneously switching elements. Even so, the overall noise reduces in approximately a quadratic fashion with respect to the supply voltage. Also, using advanced packaging can reduce the lead inductances dramatically over conventional packaging (e.g. a MCM package has less than 2nH of lead inductance while a typical wirebond pin grid array package has about 10nH).

Resistive drops occur on the power bus due to its finite resistance. The instantaneous peak resistive drop is equal to $R * I_{peak}$ which is proportional to V_{dd}^2 . This component therefore scales

quadratically. However, due to increased parallelism at lower voltages, this component only scales linearly.

3.4 Summary

A key design methodology for low-power design is aggressive voltage scaling strategy beyond the conventional 5V to 3.3V scaling. Conventional voltage scaling techniques have used metrics such as maintaining reliability and/or maintaining the circuit speed. Unfortunately, this has been driven by the primary figure-of-merit for computers, that use one of the industry standard microprocessors, which has become the clock rate, and has therefore resulted in architectures which are antagonistic to low power operation - uni-processor implementations which require circuits to operate at the fastest possible rate or at the highest possible voltage. On the other hand, parallel architectures, operating at reduced clock rates, can have the *same* effective performance in terms of instructions/second at substantially reduced power levels. Aggressive voltage scaling can result in more than an order of magnitude reduction in power compared to conventional scaling approaches without loss in performance.

The issue of how far this approach can be taken, is interesting to address. One limit is reached, when the noise margin of the devices is degraded (though the reduced clock rates limit ground bounce problems), however, before this limit the overhead in increased parallelism to compensate for the reduced logic speed begins to dominate, increasing the power faster than the voltage reduction decreases it. Through a number of examples, we find that the optimum occurs in the range between 1-1.5volts, resulting in an order of magnitude lower power dissipation than the conventional "low power" 3.3 volt scaling. A minor modification to the technology which involves threshold voltage reduction is required to scale the power supply voltages into the sub 1-V range, which results in even more power savings.

Architecture driven voltage scaling assumes that arbitrary power supply voltages can be generated

with high efficiency. A Buck converter topology has been shown to generate arbitrary voltages with efficiencies greater than 90%. Level conversion circuitry can be used to go between different voltage levels and maintain compatibility with existing “high-voltage” standards like 3.3V!

CHAPTER 4

Approaches to Minimizing Switched Capacitance

In the previous chapter, power dissipation was minimized in CMOS circuits by aggressive supply voltage scaling. Since CMOS circuits do not dissipate power if they are not switching, another approach to low power design is to reduce the switching activity to the minimal level required to perform the computation. This can range from simply powering down the complete circuit or portions of it, to more sophisticated schemes in which the clocks are gated or optimized circuit architectures are used which minimize the number of transitions. The focus of this chapter is on minimizing the effective switching capacitance at all levels of the design. The emphasis will be on signal processing applications (such as video compression, speech recognition, etc.) which have two important attributes that make them *different* from general purpose computation (such as X-server computation, SPICE simulations, etc.): throughput constrained computing and the correlation present in the data being processed.

The throughput constraint implies that once the required sample rate is met by the hardware, there is NO advantage to making the computation faster (unlike general purpose computation where the goal is to compute as fast as possible). This attribute was exploited in the previous chapter to

maintain the throughput at reduced supply voltages by modifying architectures to be more parallel and pipelined, resulting in a dramatic reduction of the power consumption. Here it will be shown that restructuring logic can reduce the overall switching activity without sacrificing the throughput requirements.

The second attribute of signal processing applications which can be used in minimizing the switched capacitance, is the correlation which can exist between values of a temporal sequence of data, since switching should decrease if the data is slowly changing (highly correlated). This is in contrast to general purpose computation, where the data being processed tends to be random. To illustrate the correlation in signal processing applications, consider once again the transition characteristics of a typical DSP signal, human speech.

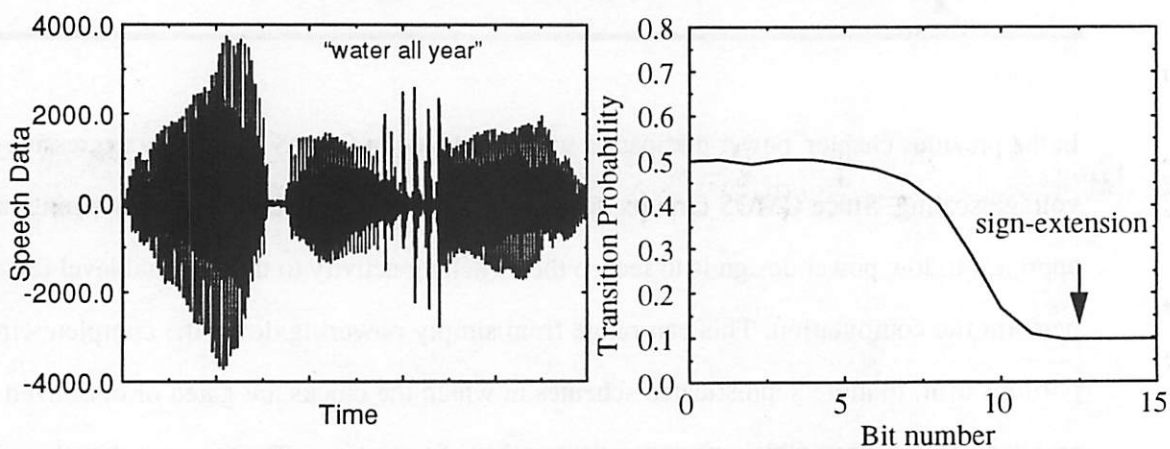


Figure 4-1 : Data in signal processing applications is often correlated.

The left side of Figure 4-1 shows the signal samples for a small window of time. The right side of Figure 4-1 shows the transition probabilities (both the 0->1 and 1-> 0) for each bit (assuming two's complement representation). As described in Chapter 2, there are three important regions in the graphs: lower-order region (the LSB's), the middle region, and the higher-order region. In the lower-order region, the bits are uncorrelated both temporally and spatially and therefore the transition probability is 1/2. The higher order bits represent sign extension operation performed in two's complement representation; i.e., the transition probability on the higher order bits is

determined by the frequency at which the signal changes from positive to negative or from negative to positive. In the middle region, the transition of the bits fall between the transition probabilities of the lower order bits and the higher bits.

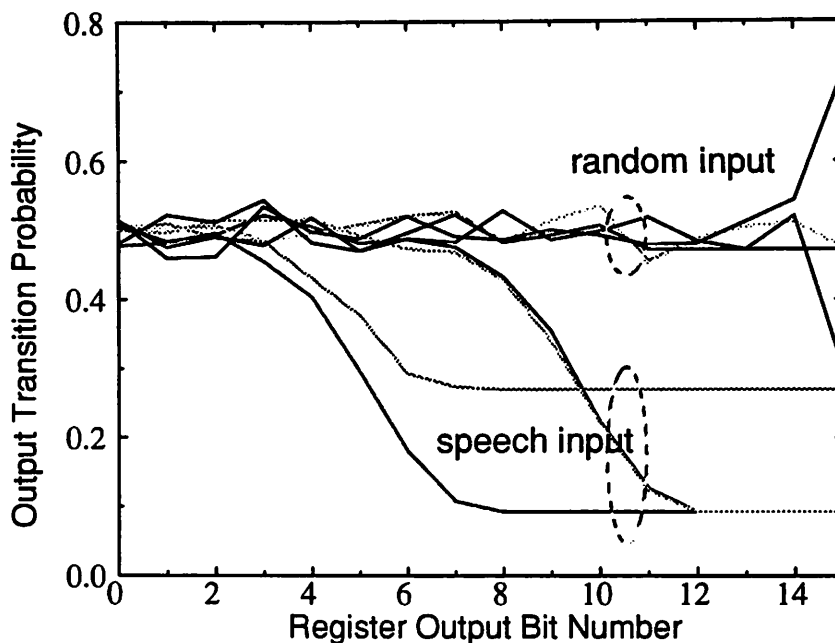


Figure 4-2: Dependence of activity on statistics: correlated vs. random input.

An example of the difference in the number of transitions which can be obtained for a highly correlated data stream (human speech) versus random data is shown in Figure 4-2 - the transition activity for a few registers in a 14th order 16-bit FIR filter design. For an architecture which does not destroy the data correlation, the speech data switches 80% less capacitance than the random input. Certain architectures can destroy signal correlations, as will be described later in this chapter, increasing the switching capacitance. In addition, the sequencing of operations can result in large variations of the switching activity due to these temporal correlations.

The following sections describe a system level approach to minimize switching capacitance which involves optimizing algorithms, architectures, logic design, and circuit design.

4.1 Algorithmic Optimization

The choice of algorithm is the most highly leveraged decision in meeting the power constraints. The ability for an algorithm to be parallelized will be critical and the basic complexity of the computation must be highly optimized.

4.1.1 Minimizing the Number of Operations

Minimizing the number of operations to perform a given function is critical to reducing the overall switching activity. To illustrate the power trade-offs that can be made at the algorithmic level, consider the problem of compressing a video data stream using the vector quantization algorithm. Vector quantization (VQ) is a lossy compression technique which exploits the correlation that exists between neighboring samples and quantizes samples together rather than individually. Detailed description of vector quantization can be found in [Gersho92].

Figure 4-3 shows a block diagram of the VQ encoding/decoding process. On the encoder side, a group of pixels is blocked into a vector and compared (using a metric such as Mean Square Error or Absolute Error) against a set of predetermined reproduction vectors (a set of possible pixel patterns) and the index of the best match is output. The decoder has a copy of all possible reproduction vectors (codebook) and the index of the best codeword is used to reconstruct the image using a simple table lookup operation. For this example, the image is segmented into 4x4 blocks (i.e the vector size is 16) and there are 256 levels in the codebook. In this case, 16:1 compression is achieved since only 8-bits are transmitted (choosing 1 out of 256 levels) instead of 16 x 8 bits for the true data.

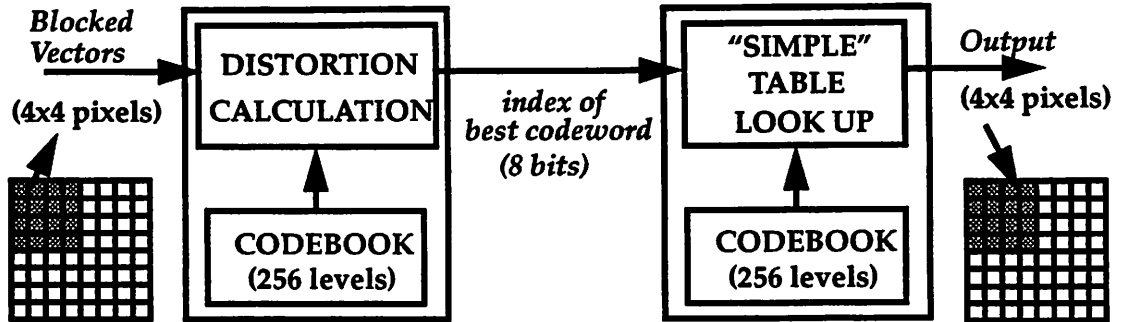


Figure 4-3 : Video compression/decompression using Vector quantization.

In this section, the focus will be on evaluating the computational complexity of encoding algorithms for VQ. Typically, the distortion metric used is mean square error. The distortion metric between an input vector X and a codebook vector C_i is computed as follows:

$$D_i = \sum_{j=0}^{15} (X_j - C_{ij})^2 \quad (\text{EQ 105})$$

Three VQ encoding algorithms will be evaluated: full search, tree search and differential codebook tree-search.

Full Search Vector Quantization

Full search is a brute-force VQ in which the distortion between the input vector and every entry in the codebook is computed. Figure 4-4 shows the organization of the codebook for full-search VQ. Here the distortion as shown in Equation 105 is computed 256 times (for C_0 through C_{255} in Figure 4-4) and the codeindex minimum distortion is determined and sent over to the decoder. For each distortion computation, there are 16 memory accesses (to fetch the entries in the codeword), 16 subtractions, and 16 multiplications and 15 additions. In addition to this, the minimum of 256 distortion values, which involves 255 comparison operations must be determined.

C0	PIX0	PIX1	○ ○ ○	PIX14	PIX15
C1	PIX0	PIX1	○ ○ ○	PIX14	PIX15
C2	PIX0	PIX1	○ ○ ○	PIX14	PIX15
			○ ○ ○		
C254	PIX0	PIX1	○ ○ ○	PIX14	PIX15
C255	PIX0	PIX1	○ ○ ○	PIX14	PIX15

Figure 4-4 : Codebook organization for Full-search Vector Quantization.

Tree-structured Vector Quantization

In order to reduce the computational complexity of an exhaustive full-search vector quantization scheme, a binary tree-search is typically used. The basic idea is to perform a binary search sequence instead of one large search. As a result, the computational complexity increases as $\log N$ instead of N , where N is the number of nodes at the bottom of the tree. Figure 4-5 shows the structure for the tree search. At each level of the tree, the input vector is compared against two codebook entries. If for example at level 1, the input vector is closer to the left entry, then the right portion of the tree is never compared below level 2 and an index bit 0 is transmitted. This process is repeated till the leaf of the tree is reached. The TSVQ will in general have some degradation over the performance of the full search VQ due to the constraint on the search. However, this is typically not very noticeable [Gersho92] and the power reduction relative to the full search VQ is very significant. Here only $2 \times \log_2 256 = 16$ distortion calculations have to be made compared to 256 distortion calculations in the full search VQ. The number of comparison operations is also reduced to 8.

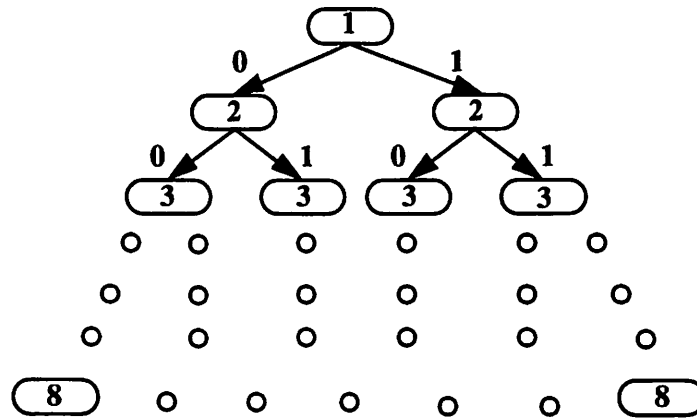


Figure 4-5 : Tree structured Vector Quantization.

Differential Codebook Tree-structure Vector Quantization

Another option is to use the same search pattern as the previous scheme but perform computational transformations to minimize the number of switching events [Fang92]. In the above scheme, at each level of the tree, the distortion difference between the left node and right node needs to be computed. This is summarized by the following equation:

$$D_{left-right} = \sum_{j=0}^{15} (X_j - C_{leftj})^2 - \sum_{j=0}^{15} (X_j - C_{rightj})^2 \quad (\text{EQ 106})$$

This equation can be manipulated to reduce the number of operations.

$$D_{left-right} = \sum_{j=0}^{15} \left((X_j - C_{leftj})^2 - (X_j - C_{rightj})^2 \right) \quad (\text{EQ 107})$$

$$D_{left-right} = \sum_{j=0}^{15} \left(X_j^2 + C_{leftj}^2 - 2X_j C_{leftj} - X_j^2 - C_{rightj}^2 - 2X_j C_{rightj} \right) \quad (\text{EQ 108})$$

$$D_{left-right} = \sum_{j=0}^{15} \left(C_{leftj}^2 - C_{rightj}^2 \right) + \sum_{j=0}^{15} 2X_j \left(C_{rightj} - C_{leftj} \right) \quad (\text{EQ 109})$$

The first term in Equation 109 can be precomputed for each level and stored. By storing and accessing $2 \times (C_{rightj} - C_{leftj})$, the number of memory access operations can be reduced; that is, by changing the contents of the codebook through computational transformations, the number of switching events - number of multiplications, additions/subtractions and memory accesses- can be reduced. Table 4-1 shows a summary of the computational complexity per input vector (16 pixels). This example clearly demonstrates the dramatic reduction in computational complexity that can be achieved through algorithmic optimizations.

Table 4-1 : Computational complexity of VQ encoding algorithms.

Algorithm	# of Memory Accesses	# of Multiplications	# of Additions	# of Subtractions
Full Search	4096	4096	3840	4096
Tree Search	256	256	240	264
Differential Search Tree Search	136	128	128	0

4.1.2 Minimizing Shift-Add Operations for Multiplications with Multiplications by Constants

Multiplications with constant coefficients is a commonly performed operation in signal processing applications. Example applications include FIR filters, IIR filters, Discrete Cosine Transform (DCT) which is used in many image compression algorithms, matrix multiplication operations, etc. For this general class of applications, an important power minimization technique that can be applied is conversion of multiplications into shift-add operations. The basic idea exploited is that a multiplication with a zero is a NOP and therefore can be eliminated. A shift-add multiplication can be implemented using a direct-mapped full parallel approach - in which there is a one to one

correspondence between the shift-add operations on the flowgraph and the operators on the hardware implementation - or a time-multiplexed approach - in which there is one adder and one shifter and a controller which controls the shifter. Figure 4-6 shows the block diagram for the two approaches. In the parallel implementation, there is no control required since the shift operation is hardwired, which means that the shift operation degenerates to routing. In addition to control overhead, the time-multiplexed approach also has the overhead of one register access for each shift-add operation. Therefore, if there are many 1's in the coefficient, the time-multiplexed shift-add implementation can consume more power compared to a full array multiplier.

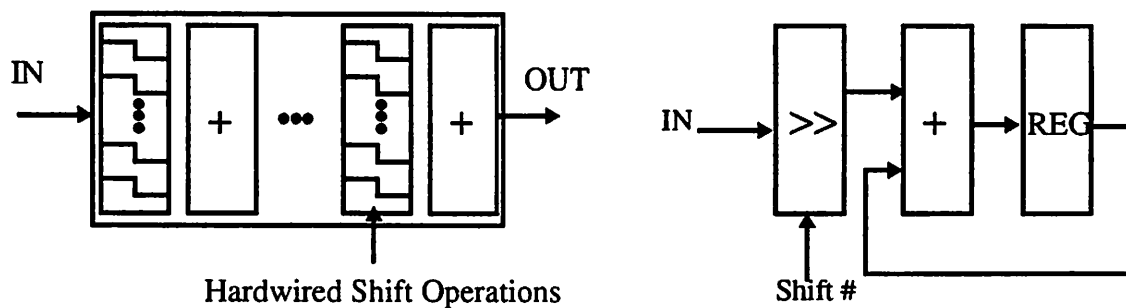


Figure 4-6 : Hardwired shift-add and time-muxed shift-add implementations.

Several algorithms and tools have been developed to minimize the number of shift-add operations for constant coefficients [Jain85].

Another commonly used optimization technique is to scale the coefficients so as to minimize the number of shift-add operations. When multiplication of multiple coefficients with a single input variable is involved, common-subexpression elimination can be combined with scaling to minimize the number of shift-add operations. The basic idea is illustrated in the example shown in Figure 4-7, which involves multiplying an input with two coefficients. The left side shows a brute-force implementation in which the two outputs are computed in parallel. This requires 4 shifts and 3 additions. Another approach is shown on the right, which exploits the common terms in the constants to share some of the shift-add operations; here, A is computed first and is then used in the computation of B. This overall number of shift-add operations that have to be performed

reduced.

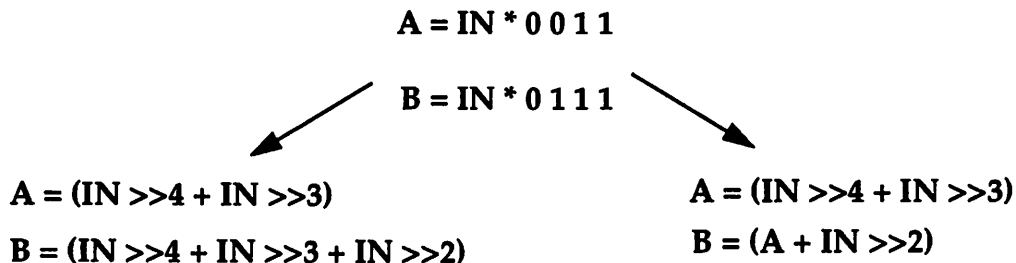


Figure 4-7 : Example of common sub-expression elimination.

Example of Color Space Conversion

To illustrate the application of shift-add conversion on a larger example, consider the color space conversion matrix from RGB \rightarrow YIQ which is used in most compression algorithms. Figure 4-8 shows the basic operation being performed. On the encoder, a digital RGB vector is converted to digital YIQ and sub-sampled. This conversion involves a matrix multiplication operation with constant coefficients. On the decoder, the YIQ vector is converted back to RGB. For this operation, there are two main optimizations that can be performed on the multiplications. First, the coefficients on the encoding matrix can be scaled; for example, C_{21} , C_{22} , and C_{23} can all be scaled by $1/\alpha_1$, as long as the coefficients on the decoder matrix D_{12} , D_{22} , and D_{32} are scaled by α_1 . This gives some opportunity for exploring the design space to minimize the number of shift-add operations. Of course, the scaling operation should not come at the expense of “visual image degradation”. The second optimization is to exploit multiplication of inputs with multiple coefficients; for example, D_{12} , D_{22} , and D_{32} are all multiplied with I' . This allows the application of the common sub-expression elimination to share some of the shift-add terms.

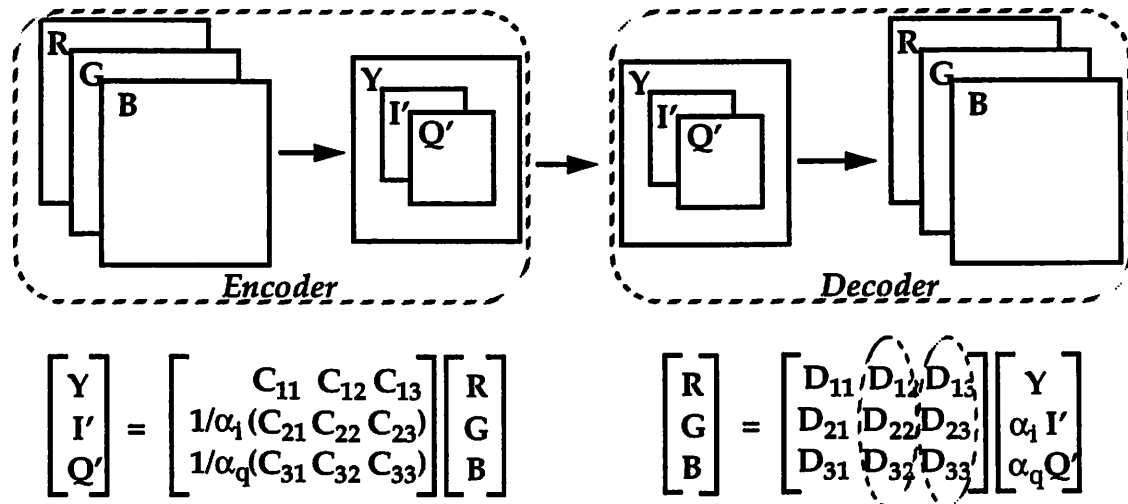


Figure 4-8 : Color conversion matrix for video applications.

Figure 4-9 shows the plot of the number of shift-add operations that have to be performed for the three coefficients D_{12} , D_{22} , and D_{32} which are all multiplied with I' . The top curve indicates the number of operations with just scaling of coefficients and does not include the effect of exploiting common terms in the three coefficients. The curve below indicates the number of operations that has to be performed using a strategy that combines scaling with common sub-expression elimination. It is clear from this plot that slight variations in the coefficients can result in significant reduction of the computational complexity.

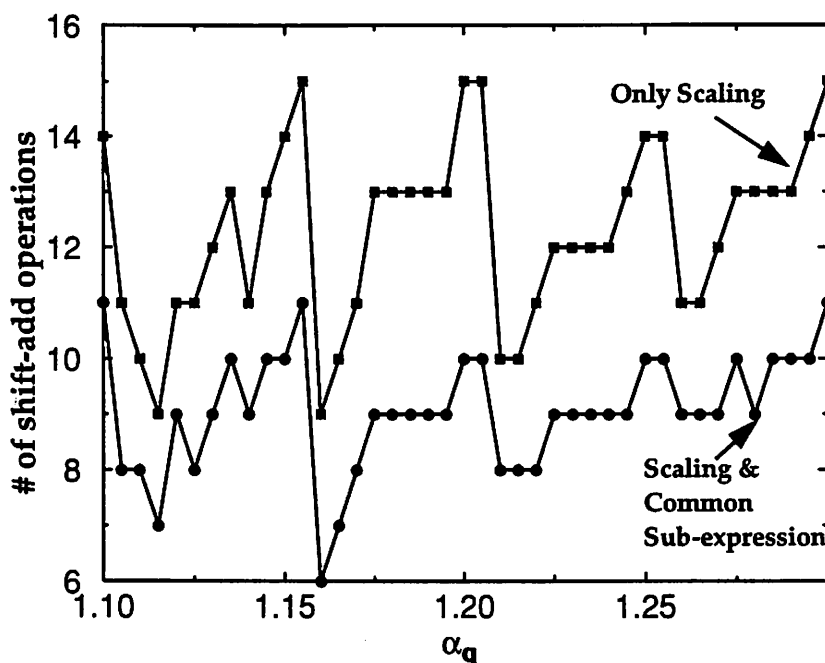


Figure 4-9 : Exploiting common sub-expression to minimize number of operations.

4.1.3 Minimizing Temporal Bit Transition Activity by Choice of Data Representation

Communicating data bits in an appropriately coded form can reduce the switching activity. Coding has long been used in communication systems to control the statistics of the transmitted data symbols, or in other words to control the spectrum of the transmitted signal. The goal of coding has been to either remove undesired correlation among information bits (scrambling, e.g. for encryption) or to introduce controlled correlation (redundancy, e.g. for spectrum shaping of transmitted signal, timing recovery, error correction in unreliable channels). Coding for reduced switching activity falls under the second category - one would like to introduce controlled correlation such that the total number of bit transitions is reduced. Interestingly, this use of coding to reduce the switching activity is just the opposite of the traditional use of some types of coding in communication channels to ensure that the transmitted signal has enough transition activity so that timing recovery and other functions can be reliably performed.

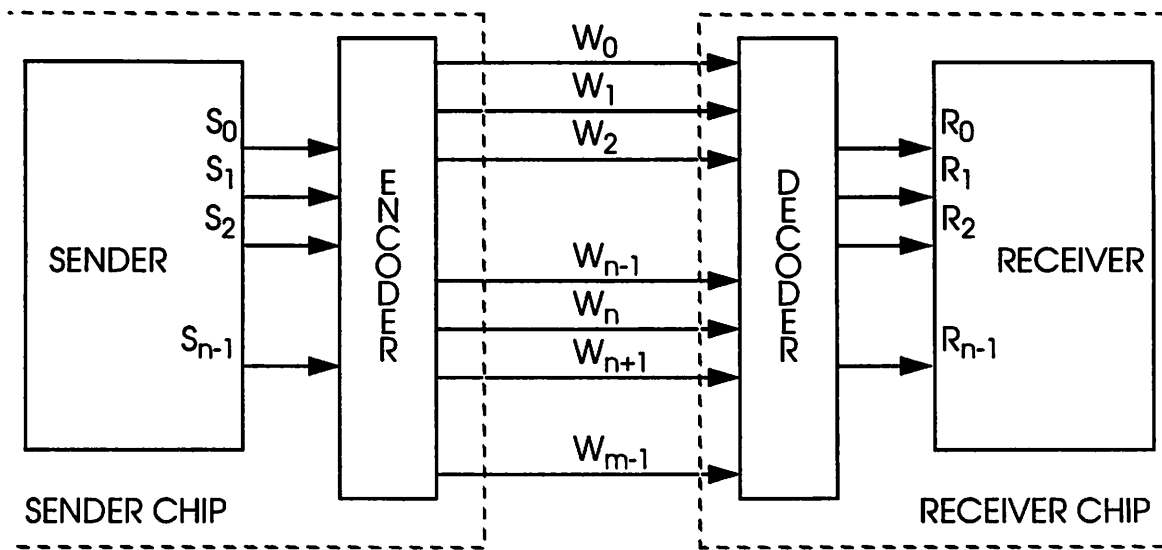


Figure 4-10 : Data coding approach to reduction of switching activity on highly capacitive inter-chip I/O Lines. Note that $m \geq n$

Consider the simple case where one chip needs to send n -bit data words to another chip, and where all the 2^n possible n -bit data words are possible as would be the case in general. The common way to do this inter-chip communication, without using any coding, is to transmit the data bits over a set of n wires. The idea behind using coding to reduce the switching activity is shown in Figure 4-10 where a reduction in the switching activity is obtained at the cost of extra hardware in the form of an encoder on the sender chip, a decoder on the receiver chip, and possibly larger number of wires m (where $m \geq n$) between the two chips. A coding scheme with simple encoder and decoder, and a small number of wires $m (\geq n)$ is obviously desirable.

A coding scheme is non-redundant if $m = n$ the 2^n elements of the set of n -bit data words will be mapped among themselves. It is obvious that one cannot devise a non-redundant coding scheme to reduce the switching activity without knowing the statistics of the sequence of words that is being communicated.

When $m > n$ the coding scheme has redundancy because the 2^n unique data words to be sent get mapped to a larger set of 2^m data words that are transmitted over the wires. This mapping may be

done in many different ways. One simple way is to use a one-to-one mapping so that $2^m - 2^n$ of the m -bit patterns are never transmitted. An advantage of such a scheme is that both the encoder and the decoder are memoryless. Another way to do the mapping is to use a static one-to-many mapping of the 2^n unique data words to be sent to the 2^m data words that are transmitted. The current state is then used to select a specific m -bit data word to transmit out of the various choices corresponding to a n -bit data word that needs to be sent. This scheme would require an encoder with memory, but the decoder is still memoryless. More elaborate schemes where both the encoder and decoder have memory are also possible.

Example #1: One-Hot Coding

Of the many coding schemes that exist in the literature, one which has relevance to switching activity reduction is the *one-hot* coding which is sometimes used in state assignment of finite state machines. In the context of the inter-chip communication problem, one-hot coding is a redundant coding scheme with a one-to-one mapping between the n -bit data words to be sent and the m -bit data words that are transmitted. The two chips are connected using $m = 2^n$ wires, and a n -bit data word is encoded for transmission by placing a '1' on the i -th wire where $0 \leq i \leq 2^n - 1$ is the binary value corresponding to the bit pattern, and '0' on the remaining $m - 1$ wires. Obviously both the encoder and the decoder are memoryless.

One-hot encoding requires an exponential number (2^n) of wires, but guarantees that precisely one $0 \rightarrow 1$ and one $1 \rightarrow 0$ bit transition occurs when a different data word is sent. Assuming the n -bit data words to be sent are independent and uniformly randomly distributed, the reduction in dynamic power consumption due to one-hot coding will be:

$$\frac{P_{\text{one-hot}}}{P_{\text{no coding}}} = \frac{\alpha_{\text{one-hot}} CV^2 f}{\alpha_{\text{no coding}} CV^2 f} = \frac{\alpha_{\text{one-hot}}}{\alpha_{\text{no coding}}} = \begin{cases} 1 & \text{for } n = 1 \\ \frac{4(1 - 2^{-n})}{n} & \text{for } n \geq 2 \end{cases}$$

While the one-hot coding gives a large reduction in switching activity, the exponential number of

wires required by it makes it an impractical scheme. For example, if $n=8$, the one-hot coding gives a 75% reduction in switching activity but requires $m=256$ wires.

Example #2: Gray-coding

Another encoding strategy which has been used for low-power is gray-coding [Su94]. A gray code sequence is a set of numbers in which adjacent numbers only have one bit difference. Gray coding is most useful when the data being transmitted over a bus is sequential in nature. In this case, the number transitions for binary representation over gray-coding approaches 2 since for large values of n , the number of transitions for binary representation will approach 2 while gray-code will always have one transition. Table 4-2 shows the binary representation and gray-code representation for decimal numbers 0 through 15.

Table 4-2 : Binary and Gray-code representation.

Decimal Value	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011

Table 4-2 : Binary and Gray-code representation.

Decimal Value	Binary Code	Gray Code
14	1110	1001
15	1111	1000

The conversion from binary code to gray code representation is defined as follows. Let $B = \langle b_{n-1}, b_{n-2}, \dots, b_1, b_0 \rangle$ be the binary representation and $G = \langle g_{n-1}, g_{n-2}, \dots, g_1, g_0 \rangle$ be the gray code representation for the number. Note that gray code is a non-redundant representation.

$$g_{n-1} = b_{n-1} \quad (\text{EQ 110})$$

$$g_i = b_{i+1} \oplus b_i \quad (i=n-2,0) \quad (\text{EQ 111})$$

For example, a binary code representation $B = \langle 1, 1, 0, 1 \rangle$ is equivalent to a gray code representation $\langle b_3, b_3 \oplus b_2, b_2 \oplus b_1, b_1 \oplus b_0 \rangle$ which is equal to $\langle 1, 0, 1, 1 \rangle$.

The conversion from gray code to binary also uses XOR operation, but is slightly more complex.

For the MSB, once again:

$$b_{n-1} = g_{n-1} \quad (\text{EQ 112})$$

For the rest of the bits, b_i is equal to the XOR of g_i and all the and all the bits of G preceding g_i .

$$b_i = b_{i+1} \oplus g_i \quad (i=n-2,0) \quad (\text{EQ 113})$$

For example, a gray code representation $G = \langle 1, 1, 0, 1 \rangle$ is equivalent to a binary representation $\langle g_3, g_3 \oplus g_2, g_3 \oplus g_2 \oplus g_1, g_3 \oplus g_2 \oplus g_1 \oplus g_0 \rangle$ which is equal to $\langle 1, 0, 0, 1 \rangle$.

Gray coding has been applied to code the address lines for both instruction access and data access to reduce the number of transitions [Su94]. For instruction accesses, it was found that gray coding significantly reduced the switching activity since the temporal transitions were typically sequential. When a branch occurs, the temporal transition is not sequential and therefore there will be more than one bit transition for the gray coding approach. For data access, it was found that the transitions were approximately equal for both representations since the data accesses were not as

sequential. For random data patterns, it was found that binary and gray coding have approximately equal transition activity. Figures 4-11 and 4-12 show the reduction in switching activity for instruction address coding and data address coding. BPI is the number of bit transitions for each instruction.

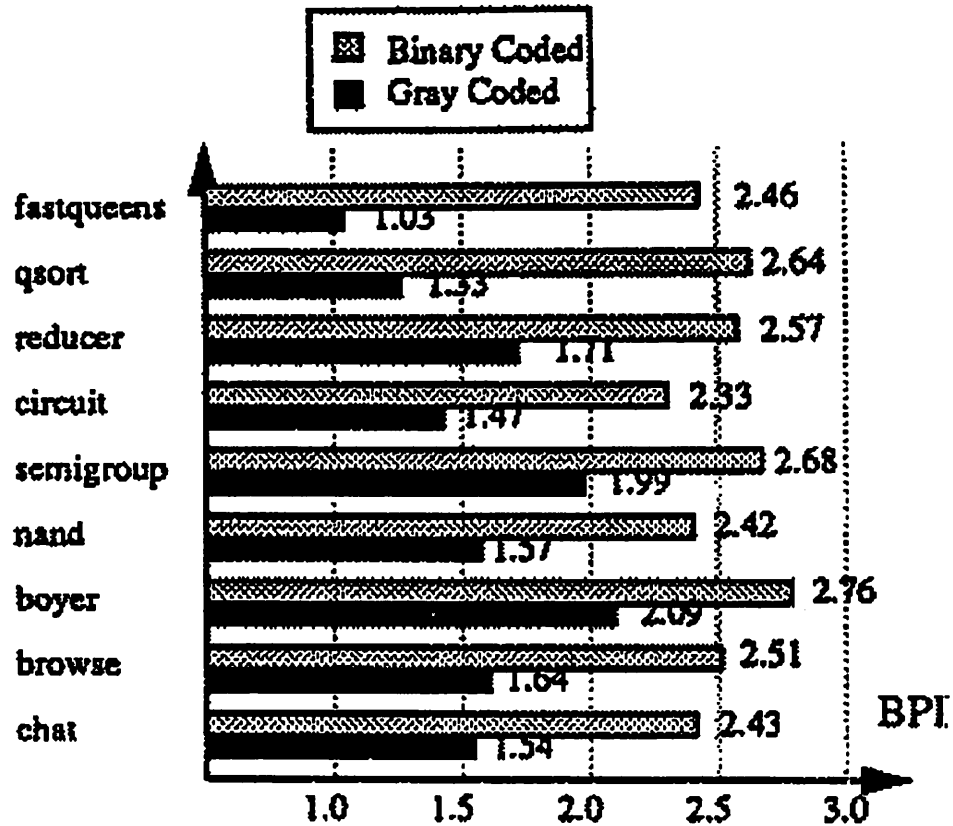


Figure 4-11 : Temporal transition activity comparison for instruction addresses [Su94].

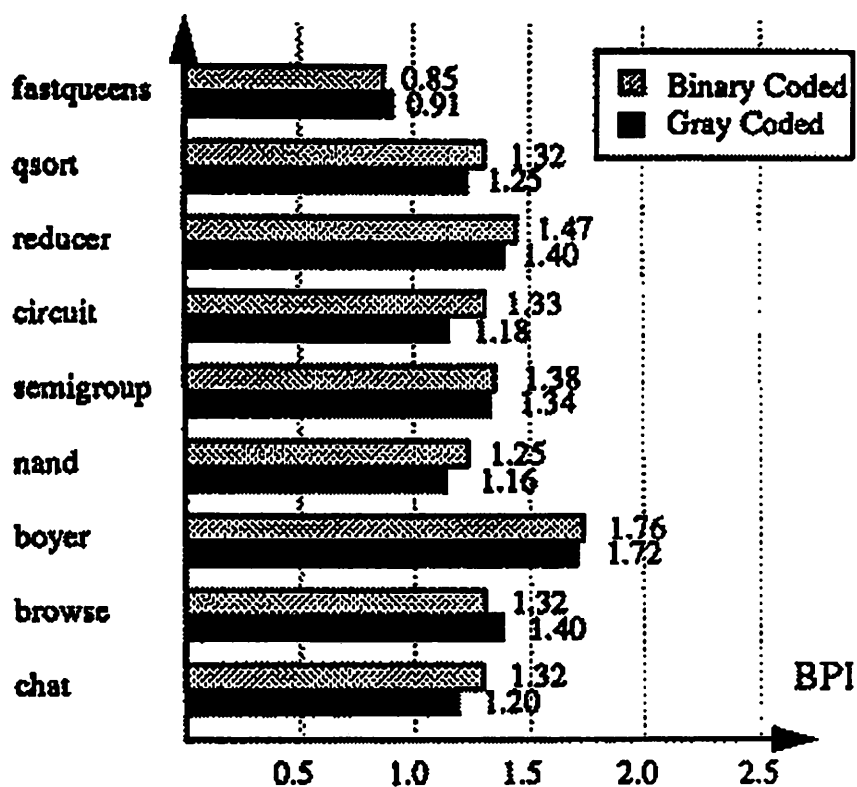


Figure 4-12 : Temporal transition activity comparison for data addresses [Su94].

Example #3: Bus Inversion Coding

Another coding scheme is proposed in [Stan94] - it is a redundant coding scheme with a one-to-many mapping between the n -bit data words that need to be sent, and the m -bit (where $m=n+1$) data words that are transmitted:

If the i -th data word to be transmitted over the wires is S_i , then we either transmit S_i or \bar{S}_i depending on which would cause fewer number of wires to change their values (i.e. fewer bit transitions). Note that \bar{S}_i is the bit-wise inverse of S_i . Further, to tell the receiver as to what is being transmitted - S_i or \bar{S}_i - an extra wire P is used to carry this information. This extra wire thus encodes the polarity of the data word. The possibility of a transition on the polarity wire is taken into account when deciding whether to transmit S_i or \bar{S}_i .

The scheme is simple: the decoder is memoryless, the encoder only uses the current state of the wires for its memory, and both the decoder and the encoder can be implemented with little area and power overhead. The number of extra wires required is one (i.e. $m=n+1$), which the smallest possible number of for any redundant coding scheme.

Analysis shows that for uniformly distributed n -bit data words the reduction in dynamic power consumption due to the above coding scheme is:

$$\frac{P_{\text{coding}}}{P_{\text{no coding}}} = \frac{\alpha_{\text{coding}} CV^2 f}{\alpha_{\text{no coding}} CV^2 f} = \frac{\alpha_{\text{coding}}}{\alpha_{\text{no coding}}} = \left(1 + \frac{1}{n}\right) \left(1 - \frac{{}^n C_{\lceil n/2 \rceil}}{2^n}\right)$$

Figure 4-13 [Srivastava94] is a plot of the above ratio as a function of the number of bits n . As the plot shows, the maximum reduction in switching activity is obtained for $n=2$ for which the reduction is 25%. Even for 32 bit busses, a reduction of 11% is obtained by using the coding scheme at the cost of one extra wire. It is easy to see from the analytic expression for $\alpha_{\text{coding}}/\alpha_{\text{no coding}}$ as well as from the plot in Figure 4-13 that the efficacy of the coding technique disappears for large values of n , as the ratio $\alpha_{\text{coding}}/\alpha_{\text{no coding}}$ converges to 1.

Since the coding technique works better for smaller values of n , one can use the following extension which uses a larger number of extra wires to obtain larger reductions in switching activity: divide the n -bit data bus into smaller groups (of 2 or more bits) and code each group independently by associating a polarity bit with each of the group. For example, a 32-bit bus can be divided into 4 groups of 8-bit which are then coded independently using a total of 4 extra wires, thus giving 18.3% reduction in switching activity as opposed to the 11.3% reduction that is obtained by coding the entire bus together using only one extra wire. In the extreme, by using 16 extra wires, and coding the original 32 bits in 16 groups of 2 bits, one can get a reduction in switching activity of 25%. Since groups have at least 2 bits (coding 1 bit is useless), this extended approach allows one to make use of up to $n/2$ extra wires.

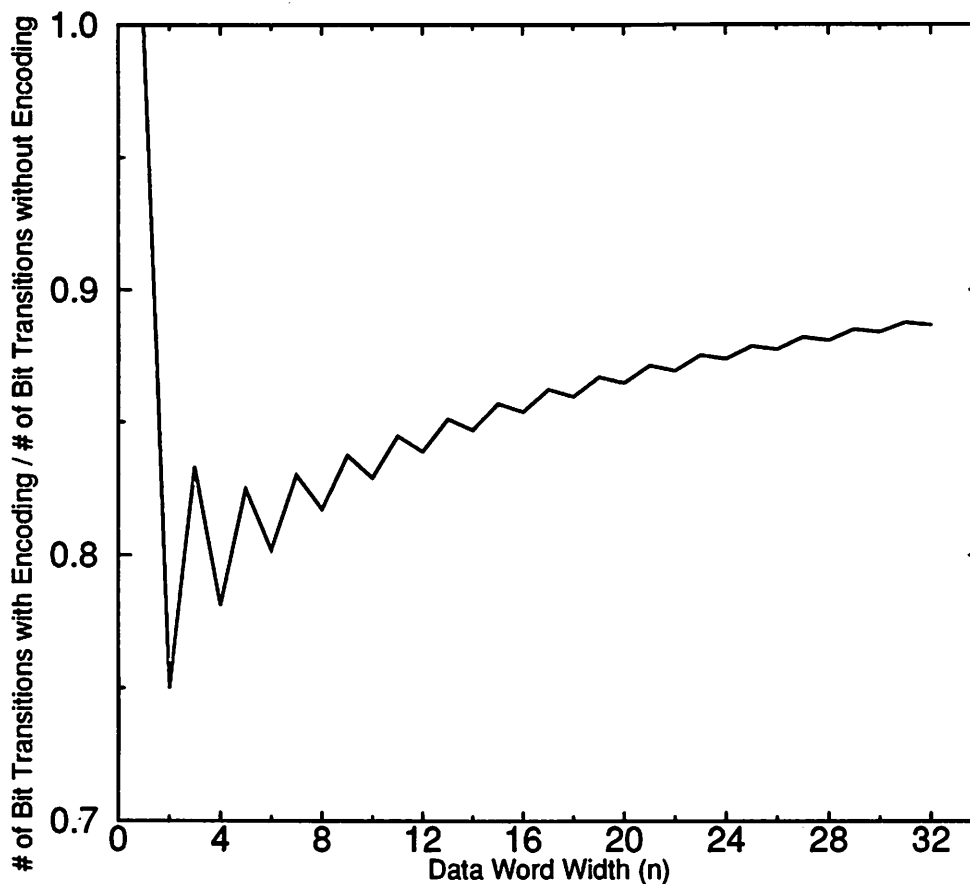


Figure 4-13 : Theoretically predicted reduction in switching activity for a stream of uniformly distributed data words [Srivastava94].

Example #4: Two's Complement vs. Sign Magnitude

Another choice of data representation is two's complement vs. sign magnitude. In most signal processing applications, two's complement is typically chosen to represent numbers since arithmetic operations (addition and subtraction) are easy to perform. One of the problems with two's complement representation is sign-extension, which causes the MSB sign-bits to switch when a signal transitions from positive to negative or vice-versa (for example, going from -1 to 0 will result in all of the bits toggling). Therefore using a two's complement representation can result in significant switching activity when the signals being processed switch frequently around zero and when they do not utilize the entire bit-width (i.e., the dynamic range is much smaller than

the maximum possible value determined from the bit-width) since a lot of the MSB bits will perform sign-extension. Even if a signal utilizes the entire bit-width, arithmetic operations such as scaling can reduce the signal dynamic range.

One approach to minimizing the switching in the MSBs is to use a sign-magnitude representation, in which only one bit is allocated for the sign and the rest for the magnitude. In this case, if the dynamic range of a signal does not span the entire bitwidth, only one bit will toggle when the signal switches sign, as opposed to the two's complement representation where due to sign extension several of the bits will switch. To illustrate this, consider gaussian data applied to a 16-bit data-bus, and let the signal have a mean of 0 and a $3\sigma = 2^{11}$. Figure 4-14a shows the transition probabilities vs. bit position number for two's complement representation as a function of the first order correlation coefficient $\rho = \text{cov}(X_n, X_{n+1}) / \sigma^2$. A $\rho = 0$ indicates that the data is uncorrelated and therefore the transition probability for the MSB's is 1/2 (i.e there is a 50% chance the output is going to transition from positive to negative). A large positive correlation coefficient (e.g +0.99) implies that the signal changes very slowly and therefore switches sign very infrequently (i.e. the transition probability is close to zero). Similarly, a negative correlation coefficient with a large magnitude (e.g -0.99) implies that the signal changes frequently from positive to negative.

The upper breakpoint (which represents the dynamic range of signal) lies at $\log_2(3\sigma) = 11$ bits in this case. Therefore for the two's complement representation, the bits from 11 to 15 indicate the activity of the sign-bit. If a sign-magnitude representation is used (shown in Figure 4-14b), the bits 11-14 will have a low transition activity, and bit 15, whose transitions represent the sign transition probability, will have the same activity as the two's complement representation. Also, the transition region has lower activity for the sign-magnitude representation. Clearly, there is an advantage in using sign-magnitude representation over two's complement to reduce the switching activity.

From the above example, it is clear that the reduction in the number of transitions for sign-

magnitude over two's complement is a function of both the dynamic range of the signal and the signal correlation coefficient. Consider evaluating the reduction in the number of transitions as a function of dynamic range and the correlation coefficient. Let the normalized dynamic range of a signal be defined as $3\sigma / \text{Maximum Amplitude}$. For a 16 bit example, this turns out to be: $3\sigma / 2^{16}$. Let the number of transitions represent the sum of the transitions per clock cycle for all the bits. i.e.

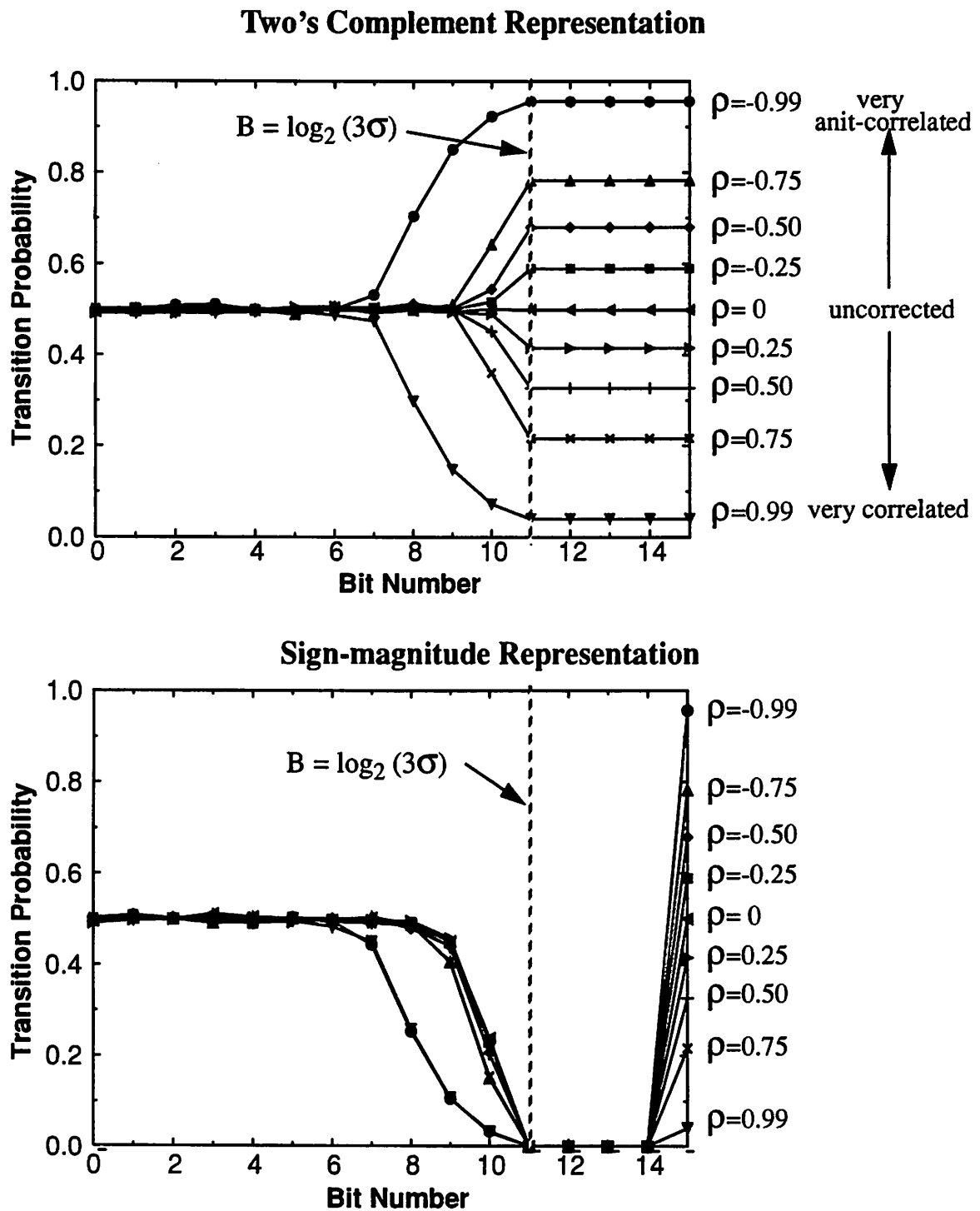
$$\text{Number of Transitions} = \sum_{i=1}^{16} \alpha_i \quad (\text{EQ 114})$$

Figure 4-15 shows a plot of the total number of transition as a function of the normalized dynamic range and the correlation factor for two's complement representation. For a correlation factor of $\rho = 0$, the number of transitions is equal to 8 and is independent of the normalized dynamic range since the data is random and each bit has a 50% probability of transitioning (\Rightarrow 8 transitions for a 16-bit bus). For very correlated data (e.g. $\rho = 0.99$), the MSB's don't switch very often while the LSB's switch 50% of the time. Therefore, for a small normalized dynamic range, the average number of transitions is very small and for a large dynamic range it approaches the value of 8 (since the lower region with activity of 50% extends out to more bits). Similarly, for very anti-correlated data (e.g. $\rho = -0.99$), the MSB switch very frequently. Therefore, a small normalized dynamic range will result in a lot of transitions (since many bits are allocated to sign-extension) while the number of transition approaches 8 for high values of the normalized dynamic range.

Figure 4-16 shows the ratio of the number of transitions required in the two's complement representation to the sign-magnitude representation as a functions of signal correlation and normalized dynamic range (plotted on a log axis). From this plot, it is clear that the biggest win for sign-magnitude representation is when the dynamic range is smallest and the signal is very anti-correlated.

The above analysis suggests that sign-magnitude has some advantages in terms of the number of the transitions on busses. However, addition and subtraction computation are difficult to implement in sign-magnitude representation. Sign-magnitude is therefore most useful for cases where

large busses have to be driven (for example external memory access), where the overhead for converting back to two's complement (which is basically a subtracter) is quite insignificant compared to the reduction in capacitance switched in the large busses. That is, a small overhead capacitance is added to the system to reduce the *overall* switched capacitance.



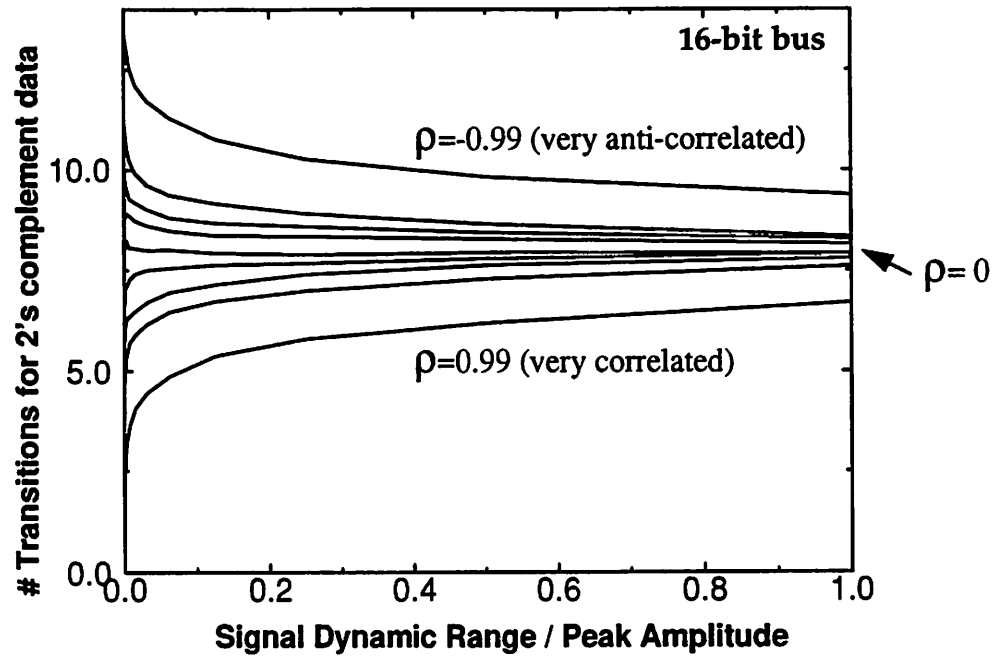


Figure 4-15 : Transition activity for two's complement representation.

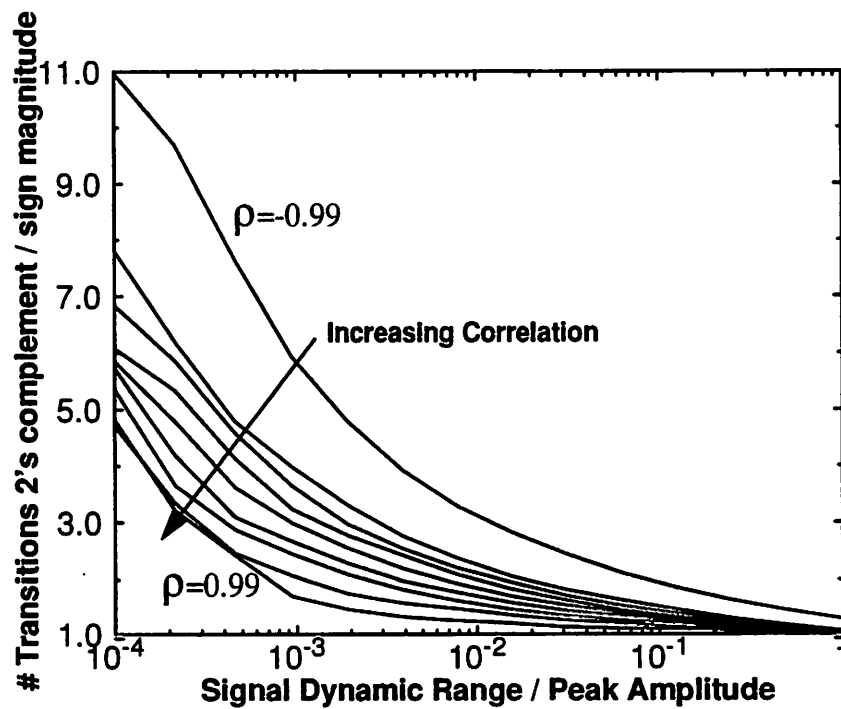


Figure 4-16 : Activity reduction for sign-magnitude over two's complement.

4.2 Architecture Optimization

Architecture optimization can also be used to significantly reduce the switching activity by optimizing the number representation, optimizing the ordering of operations, optimizing resource utilization and minimizing glitching activity.

4.2.1 Optimizing Data Representation for Arithmetic Computation

As mentioned in the previous section, sign-magnitude has some advantages in terms of reducing the number of the transitions on busses, but addition/subtraction operations are difficult to implement. However, in several applications that require multiple additions inside a sample period (e.g. Multiply Accumulate Units of DSP filters), an architecture optimization can be used that trades silicon area for lower switching activity without altering the throughput. To illustrate this consider a correlator example in which the correlation length is 1024; the samples, whose values range from -7 to +7, are accumulated at 64MHz.

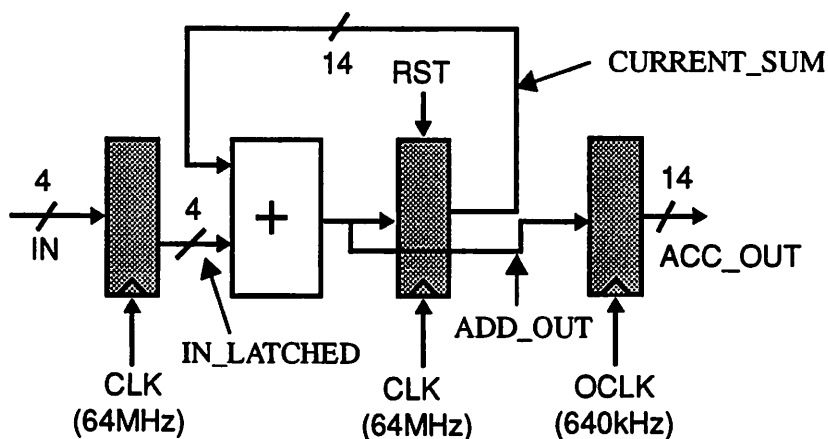


Figure 4-17 : Two's complement implementation of an accumulator.

Figure 4-17 shows a conventional architecture for an accumulator that uses two's complement representation. The accumulated result (adding 1024, 4-bit numbers) is transferred to an accumulator register at 640kHz. In this architecture, the MSB of the input, bit 3 (assuming that the

LSB is bit 0), is tied to bits 4-13 of the adder input for sign-extension and therefore anytime the input switches sign, the MSB bit (which indicates the sign) will switch, resulting in all of the higher order input bits to the adder switching.

Figure 4-18 shows the transition probabilities for three signals in the two's complement datapath assuming uniformly distributed inputs. For this distribution, the input is equally likely to be positive or negative, and therefore the sign-extension bits 3-13 have a transition probability close to 1/2. Since the accumulator acts as a low-pass filter (i.e. $CURRENT_SUM_N = CURRENT_SUM_{N-1} + IN_N$), the higher order bits have little switching activity even when the input is rapidly varying; that is, the accumulator smoothes the input signal. This is shown in Figure 4-19 which shows the $CURRENT_SUM$ output and the input value for 1024 samples (here the input is random and varying from -7 to +7). Although the $CURRENT_SUM$ has low switching activity, the adder output (before the latch) has significant switching activity due to glitching, as seen from Figure 4-18. The glitching activity arises since all of the input bits to the adder switch each time the input changes sign, and this results in high switching activity of the adder (even though the final adder output at the end of the cycle does not change relative to its value at the beginning of the cycle).

Another approach for implementing the accumulator is to use a sign-magnitude representation whose datapath is shown in Figure 4-20. Here two accumulator datapaths are used, one that sums all positive numbers and one that sums all negative numbers. The latched sign-bit from the input register is used to generate gated clocks that enables the positive datapath latch or the negative datapath latch; this ensures that this scheme does not increase effective clock load. At the end of 1024 cycles, the positive and negative accumulated values are transferred to separate registers. A subtract operation is then performed at the lower frequency and therefore is quite negligible in terms of capacitance overhead. The key to low-power is that there is no sign-extension is being performed and therefore the adder has a low switching activity in the higher order bits. In fact, the higher order bits only need an incrementer (as opposed to a full-adder required by the two's

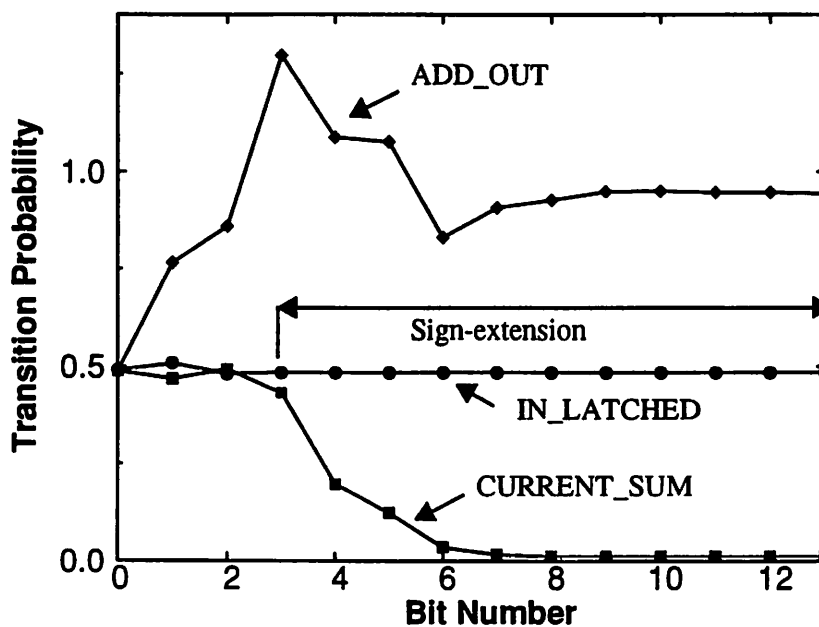


Figure 4-18 : Signal statistics for two's complement implementation of the accumulator datapath assuming random inputs.

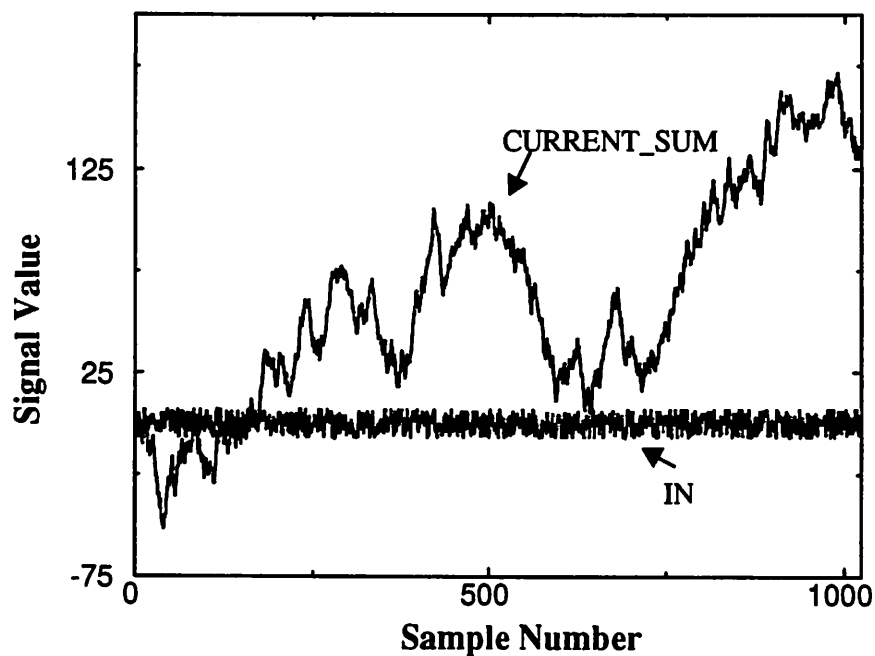


Figure 4-19 : Signal Value for two's complement implementation for random inputs.

complement implementation), reducing the number of gates that are switched in the accumulator.

Figure 4-21 shows the transition probabilities for the output of the adders in both datapaths. Also

shown in the figure is the transition activity for the two's complement implementation and the sum of the transition activities for the sign-magnitude implementation; we can see that the glitching activity is significantly reduced in the sign-magnitude implementation. The sign-magnitude implementation, however, requires control circuitry (overhead capacitance) to generate the timing signals for the various latches in the implementation.

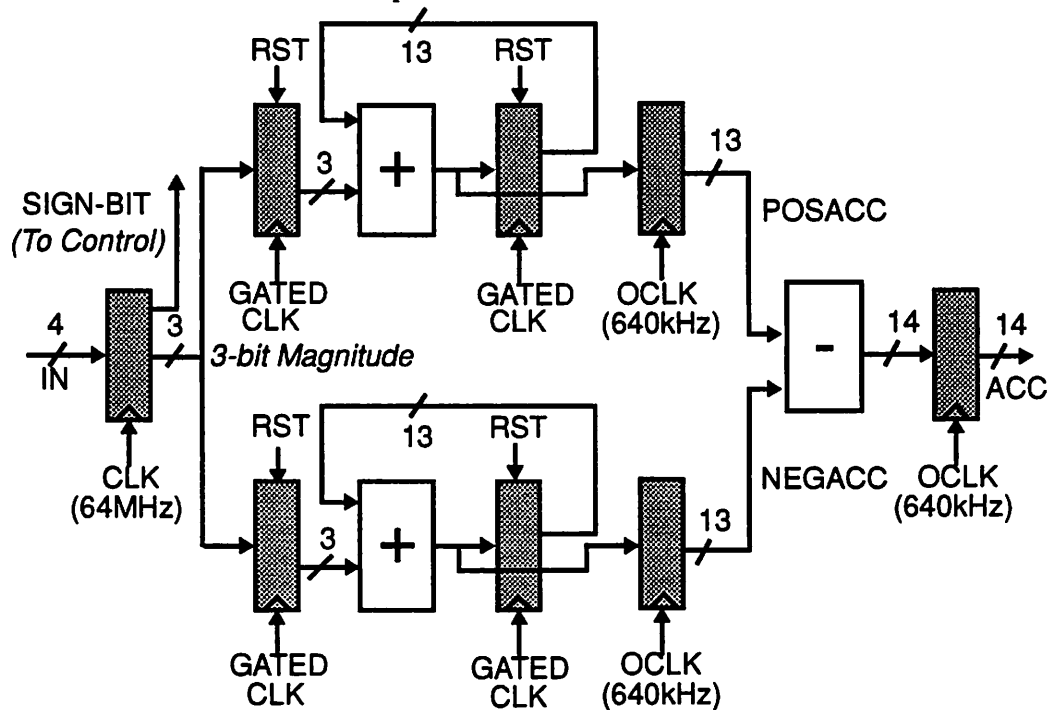


Figure 4-20 : Sign magnitude implementation of an accumulator.

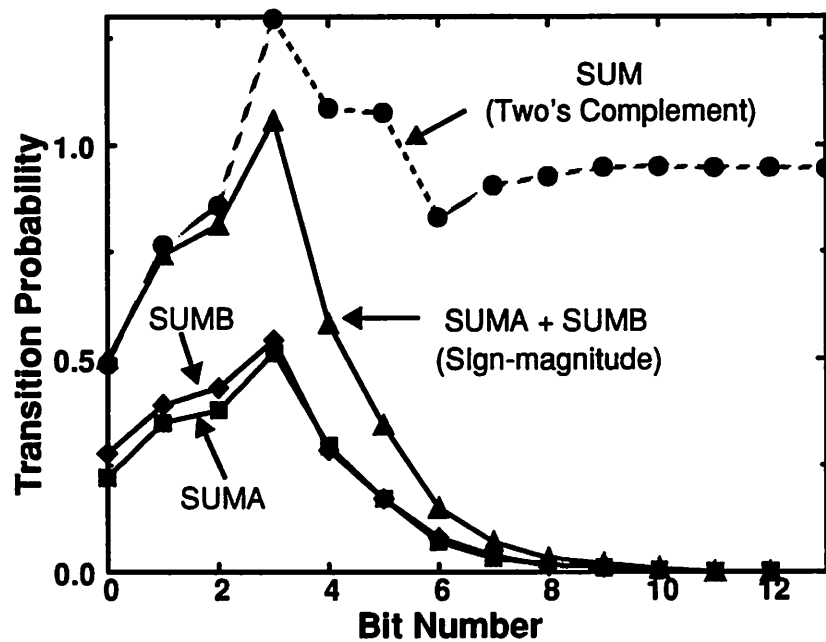


Figure 4-21 : Signal statistics for Sign Magnitude implementation of the accumulator datapath assuming random inputs.

By keeping the computation for positive data and negative data separate, the power is not very sensitive to rapid fluctuations in the input data. Table 4-3 shows the power estimates for various input patterns. Again, the biggest advantage is when the sign toggles frequently. For the case when the input changes very slowly, the sign-magnitude implementation consumes more power (15%) due to the capacitance switched by the control overhead circuitry.

Table 4-3 : Number representation trade-off for arithmetic.

Input Pattern (1024 cycles)	Two's Complement Power, 3V	Sign Magnitude Power, 3V
Constant (IN= 7)	1.97 mW	2.25 mW
Ramp (-7,-6,...7,-7)	2.13 mW	2.43 mW
Random	3.42 mW	2.51 mW
Min -> Max -> Min (-7, +7,-7,+7,...)	5.28 mW	2.46 mW

4.2.2 Ordering of Input Signals

The switching activity can be reduced by optimizing the ordering of operations in a design. To illustrate this, consider the problem of multiplying a signal with a constant coefficient, which is a very common operation in signal processing applications. Multiplications with constant coefficients are often optimized by decomposing the multiplication into shift-add operations and using the canonical sign digit representation. Consider the example in which a multiplication with a constant is decomposed into $IN + IN \gg 7 + IN \gg 8$. The shift operations represent a scaling operation, which has the effect of reducing the dynamic range of the signal. This can be seen from Figure 4-22 which shows the transition probability for the 3 signals IN , $IN \gg 7$ and $IN \gg 8$. In this example, IN has a large variance and almost occupies the entire bit-width. The shifted signals are scaled and have a much lower dynamic range.

Now consider the two alternate topologies of implementing the two required additions. In the first implementation, IN and $IN \gg 7$ are added in the first adder and the sum ($SUM1$) is added to $IN \gg 8$ in the second adder. In this case, the $SUM1$ transition characteristics is very similar to the characteristics of the input IN since the amplitude of $IN \gg 7$ is much smaller than IN and the 2 inputs have identical sign-bits (since a shift operation does not change the sign). Similarly $SUM2$ is very similar to $SUM1$ since $SUM1$ is much larger in magnitude than $IN \gg 8$. In the second implementation (obtained by applying associativity and commutativity), the two small numbers $IN \gg 7$ and $IN \gg 8$ are summed in the first adder and the output is added to IN in the second adder. In this case, the output of the first adder has a small amplitude (since we are adding 2 scaled numbers of the same sign) and therefore lower switching activity. The second implementation switched 30% less capacitance than the first implementation. This example demonstrates that ordering of operation can result in reduced switching activity.

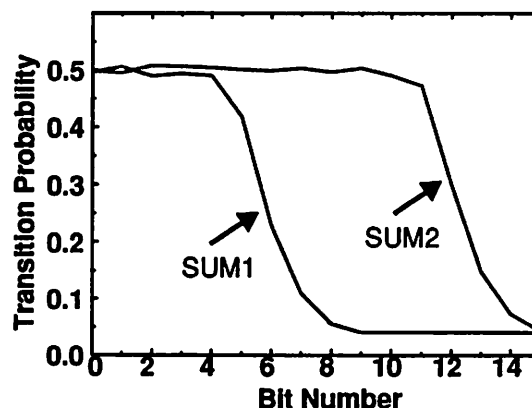
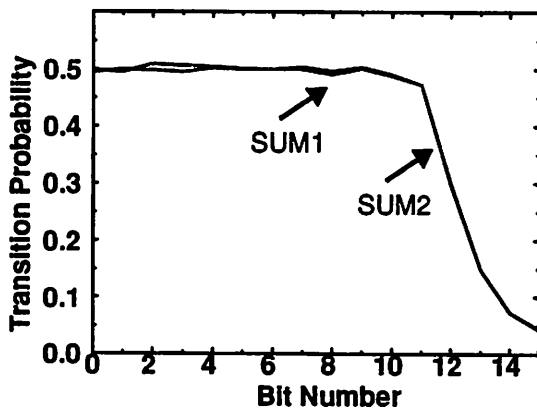
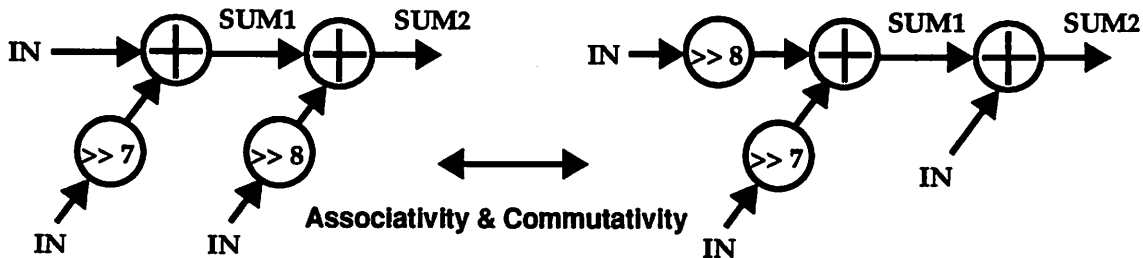
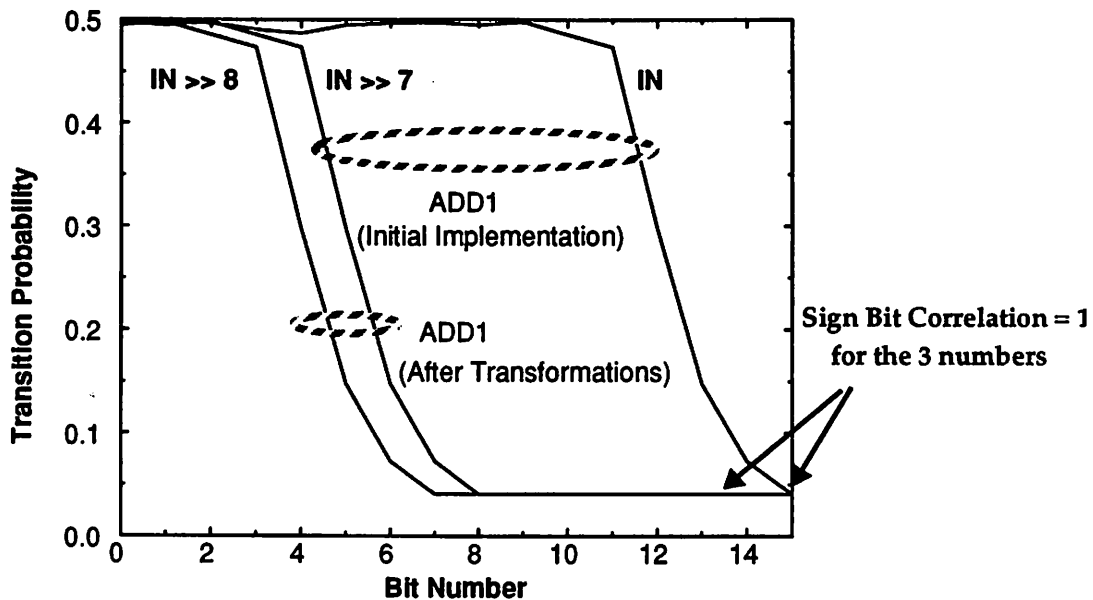


Figure 4-22 : Reducing activity by re-ordering inputs.

4.2.3 Reducing Glitching Activity

As described in Chapter 2, static designs can exhibit spurious transitions due to finite propagation delays from one logic block to the next (also called critical races and dynamic hazards), i.e. a node can have multiple transitions in a single clock cycle before settling to the correct logic level. To minimize the “extra” transitions and power in a design, it is important to balance all signal paths and reduce the logic depth. For example, consider the two implementations for adding four numbers shown in Figure 4-22 (assuming a cascaded or non-pipelined implementation). Assume that all primary inputs arrive at the same time. Since there is a finite propagation delay through the first adder for the chained case, the second adder is computing with the new C input and the previous output of $A + B$. When the correct value of $A + B$ finally propagates, the second adder recomputes the sum. Similarly, the third adder computes three times per cycle. In the tree implementation, however, the signal paths are more balanced and the amount of extra transitions is reduced. The capacitance switched for a chained implementation is a factor of 1.5 larger than the tree implementation for a four input addition and 2.5 larger for an eight input addition. The above simulations were done on layouts generated by the LagerIV silicon compiler using the IRSIM switch-level simulator over 1000 uncorrelated random input patterns.

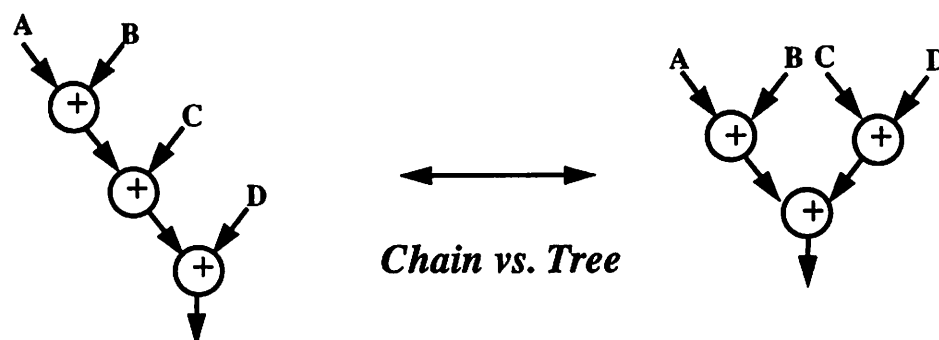


Figure 4-23 : Reducing the glitching activity

The results presented above indicate that increasing the logic depth (through more cascading) will increase the capacitance due to glitching while reducing the logic depth will increase register power. Hence the decision to increase or decrease logic depth is based on a trade-off between

glitching capacitance vs. register capacitance. Also note that reducing the logic depth can reduce the supply voltage while keeping throughput fixed.

4.2.4 Degree of Resource Sharing

A signal processing algorithm has a required number of operations that have to be performed within the given sample period. One strategy for implementing signal processing algorithms is the direct mapping approach, where there is a one to one correspondence between the operations on the signal flow graph and operators in the final implementation. Such an architecture style is conceptually simple and requires a small or no controller. Often, however, due to area constraints or if very high-throughput is not the goal (e.g. speech filtering), time-multiplexed architectures are utilized in which multiple operations on a signal flowgraph can be mapped onto the same functional hardware unit.

Given that there is a choice between time-multiplexed and fully-parallel architectures, as is the case in low to medium throughput applications, an important question arises which is what is the architecture that will result in the lowest switching activity. To first order, it would seem that the degree of time-multiplexing would not affect the capacitance switched by the logic elements or interconnect. For example, if a data flowgraph has five additions to be performed inside the sample period, it would seem that there is no difference between an implementation in which one physical adder performs all five additions or an implementation in which there are five adders each performing one addition per sample period. It would seem that the capacitance switched should only be proportional to the number of times the additions were performed. However, this turns out not to be the case. To understand this trade-off, consider two examples of resource sharing: sharing busses and sharing execution units.

Example 1: Time-sharing Busses (Output of two counters)

Consider an example of two counters whose outputs are sent over parallel and time-multiplexing

busses as shown in Figure 4-24. In case 1, we have two separate busses running at some frequency f while case 2 has one bus running twice as fast. Therefore for a fixed voltage, the power consumption for the parallel implementation is given by:

$$P_{no-bussharing} = 1/2(\sum\alpha_i) C_{nbs} V^2 f + 1/2(\sum\alpha_i) C_{nbs} V^2 f = (\sum\alpha_i) C_{nbs} V^2 f \quad (\text{EQ 115})$$

where C_{nbs} is the physical capacitance per bit of BUS1 and BUS2 (assume for simplicity that all bits have the same physical capacitance) and α_i is the transition activity for bit i (for both 0->1 and 1->0 transitions and therefore there is a factor of 1/2 in Equation 115). In this case, since the output is coming from a counter, the activity is easy to compute. $\sum\alpha_i = 1 + 1/2 + 1/4 + \dots + 1/128$ since the LSB switches every cycle, the 2nd LSB switches every other cycle, etc. This means that each bus will have on the average a total approximately two transitions per clock cycle.

For the time-multiplexed implementation, we have one bus C_{bs} running at twice the frequency. Typically C_{bs} will be smaller than C_{nbs} since with fewer interconnects, it is easier to route the chip and hence the wire lengths are smaller. The goal here is to show that the activity α' is modified due to time-multiplexing and the power consumption for this implementation is given by:

$$P_{bussharing} = 1/2 \sum\alpha' C_{bs} V^2 2f = \sum\alpha' C_{bs} V^2 f \quad (\text{EQ 116})$$

Figure 4-24 shows the plot of total number of transitions per cycle (i.e. $\sum\alpha'$) for the time-shared case as a function of the skew between the counter outputs. The figure also shows the number of transitions for the non-time-shared implementation which is independent of skew and is equal to 4 (for 2 busses). As seen from this figure, except for one value of the counter skew (when the skew is = 0), the "parallel" implementation has the lower switching activity. This example shows that time-sharing can significantly modify the signal characteristics and cause an increase in switching activity.

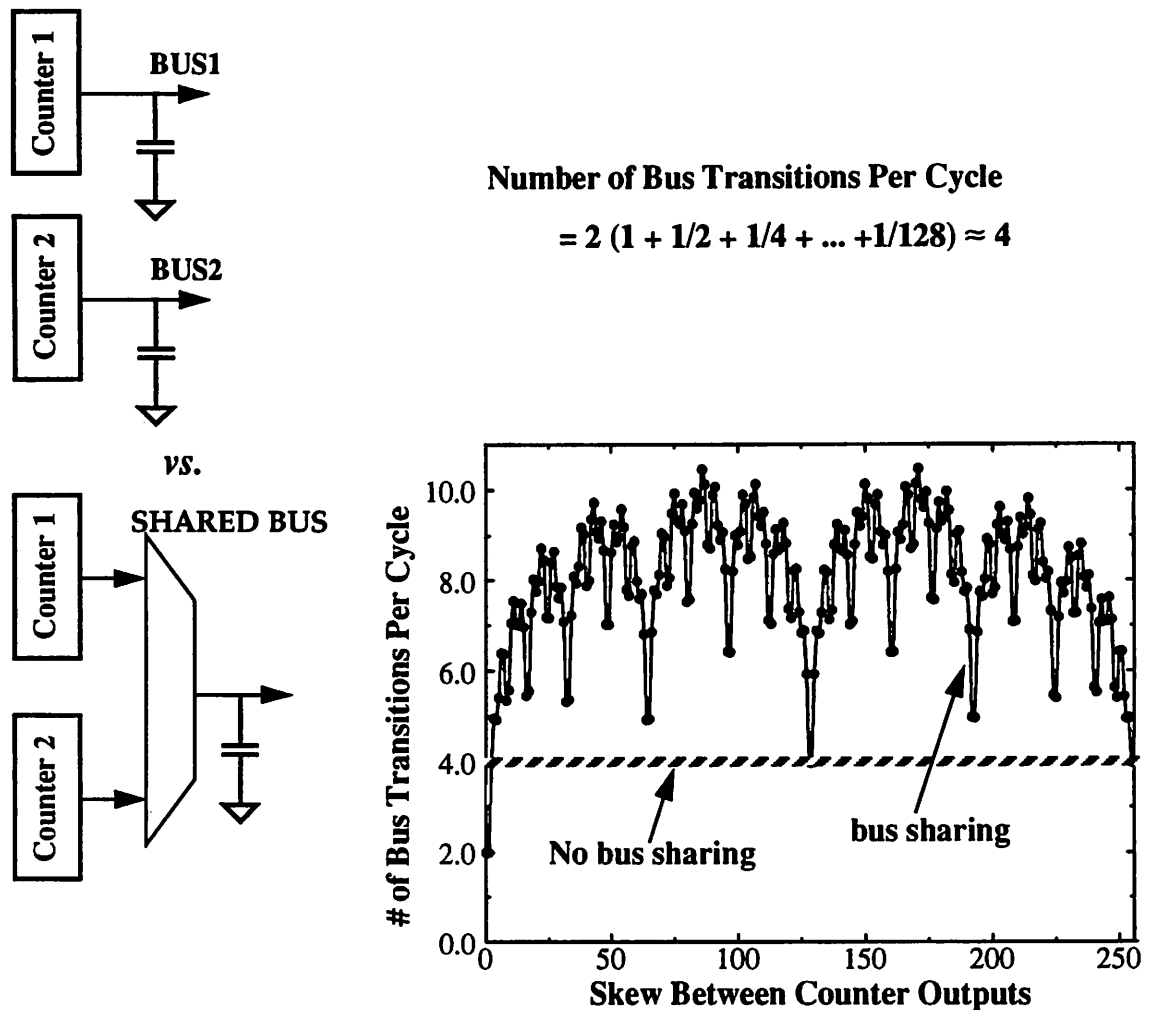


Figure 4-24 : Activity trade-off for time-multiplexed hardware: bus-sharing example.

Example 2: Time-sharing Execution Units

The second example is a simple second order FIR filter which will demonstrate that time-multiplexing of execution units can increase the switched capacitance. The FIR filter is described as follows:

$$Y = a_0 \cdot X + a_1 \cdot X@1 + a_2 \cdot X@2 = A + B + C \quad (\text{EQ 117})$$

where @ represents the delay operator and $A = a_0 \cdot X$, $B = a_1 \cdot X@1$ and $C = a_2 \cdot X@2$. Also let $O1 = A + B$. The value of the coefficients are: $a_0 = 0.15625$, $a_1 = 0.015625$, and $a_2 = -0.046875$.

One possible implementation might be to have two physical adders, one performing $A + B = O1$ and the other performing $O1 + C$. An alternate implementation might be to have a single time-multiplexed adder performing both additions. So, in cycle 1, $A + B$ is performed, and in cycle 2, $O1 + C$ is performed. The topologies for the two cases are shown in Figure 4-25. In the first implementation, the adders can be chained and therefore the effective critical path can be reduced (chaining two ripple carry adders of N bits has a delay $= (N + 1) \cdot \text{Delay of one bit}$); it is clear that this will result in extra glitching activity but since the objective here is to isolate and illustrate the activity modification only due to time-multiplexing, the first implementation is assumed to be registered. The IRSIM simulator was used to determine the switching activity for the adders in the two implementations assuming speech input data. The transition probability for the two adders for the parallel implementation and the average transition probability per addition for the time-multiplexed implementation are shown in Figure 4-25.

The results once again indicate that time-multiplexing can increase the overall switching activity. The basic idea is that in the fully-parallel implementation, the inputs to the adders change only once every sample period and the source of inputs to the adders are fixed (for example, $IN1$ of adder1 always comes from the output of the multiplier $a_0 * X$). Therefore, if the input changes very slowly (i.e the input data is very correlated), the activity on the adders become very low. However, in the time multiplexed implementation, the inputs to the adder change twice during the sample period and more importantly arrive from different sources. For example, during the first cycle, $IN2$ of the time-multiplexed adder is set to B and during the second cycle, $IN2$ of the time-multiplexed adder is set to C . Thus, even if the input is constant (which implies that the sign of X , $X@1$ and $X@2$ is the same) or slowly varying, the sign of B and C will be different since the sign of the coefficients a_1 and a_2 are different. This will result in the input to adder switching more often and will therefore result in higher switching activity. That is, even if the input to the filter does not change, the adder can still be switching. For this example, the time multiplexed adder switched 50% extra capacitance per addition compared to the parallel case (even without including

the input changes to adders or the multiplexor overhead).

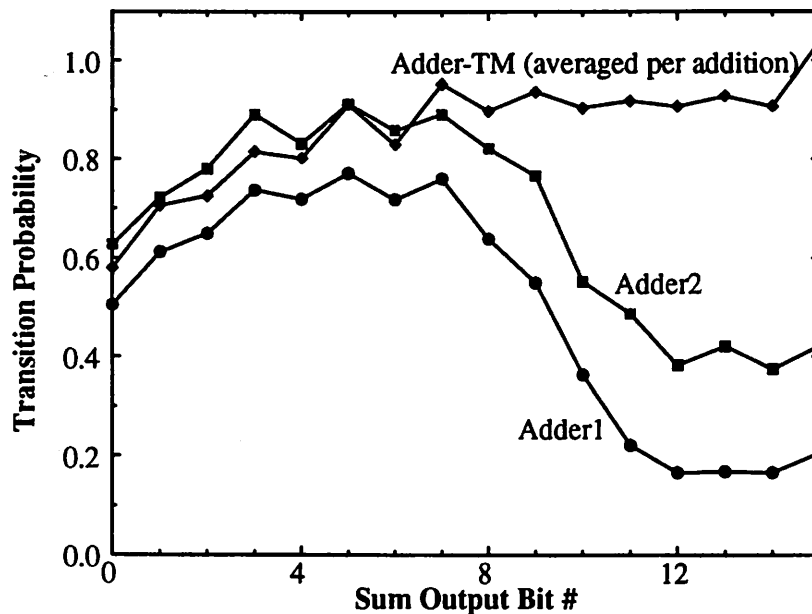
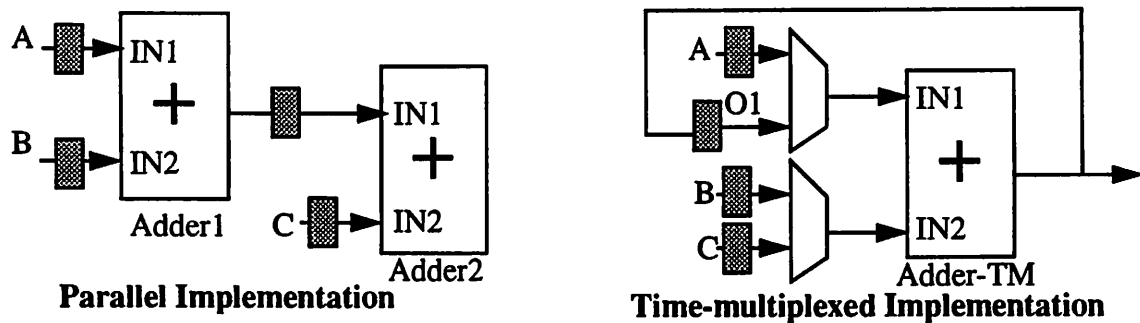


Figure 4-25 : Activity trade-off for time-multiplexed hardware: adder example.

4.3 Logic Optimization

4.3.1 Logic Minimization and Technology Mapping

Recently some effort has gone into changing the optimization criteria for synthesizing logic structures (both combinational and sequential) to one that addresses low power. Typical reduction in power consumption that can be achieved by optimizing at this level for random logic is on the order of 25%. Techniques have been proposed which choose logic to minimize switching [Shen92], position registers through re-timing to reduce glitching activity [Montero93] and to

decrease power during the technology mapping phase [Tiwari93], choosing gates from a library which reduces switching [Tsui93]; for example a three input AND gate can be implemented as a single 3-input gate (i.e NAND followed by an inverter) or two 2-input AND gates with different power results.

4.3.2 Activity Trade-off for Various Logic Structures

It has been shown in Chapter 3 that the choice of logic topology can have a strong influence on the total transition activity, which will directly impact the switching activity. A logic function such as an adder can be implemented using various different approaches. For example, an adder can be implemented using several topologies including ripple-carry, carry-select, carry-lookahead, conditional sum, and carry skip. Various circuit topologies for adders and multipliers have been investigated for impact on the transition activity [Callaway92]. Table 4-4 summarizes the results of the number of transitions per addition for different topologies assuming a randomly distributed input pattern. Though these simulation results were not obtained from layout and the capacitance switched (which is equal $\sum \alpha_i C_i$ and is the important metric) was not published, they give an idea of the switching activity for various topologies. They used a metric of $1/(\text{Number of Transitions} * \text{Delay})$ to evaluate the various adders and concluded that a lookahead topology was the best based on this metric.

Table 4-4 : Average number of gate transitions per addition [Callaway92].

Adder Type	16 bits	32 bits	64 bits
Ripple Carry	90	182	366
Carry Lookahead	100	202	405
Carry Skip	108	220	437
Carry Select	161	344	711
Conditional Sum	218	543	1323

Figure 4-26 shows the probability distributions of the number of gate transitions for various 32bit

logic adders obtained from gate level simulations.

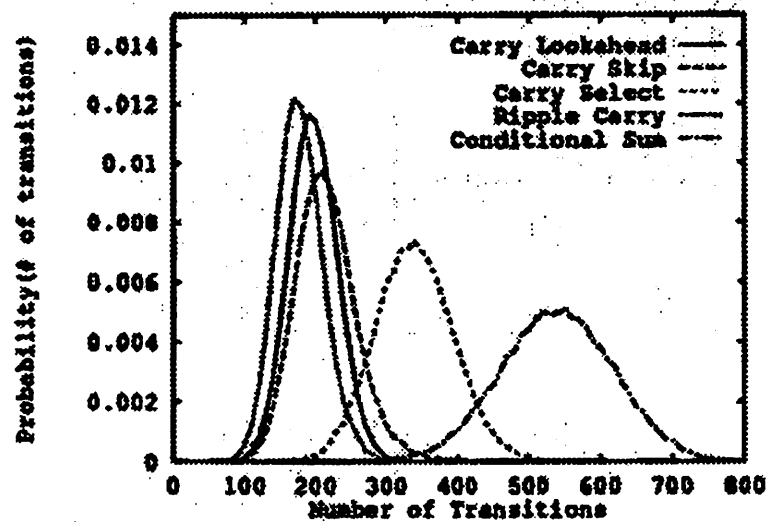


Figure 4-26 : Histogram of transition activity for various adder topologies [Callaway92].

4.3.3 Logic Level Power Down

Powering down has traditionally been applied only at the chip and module levels, however, the application to the logic level can also be very beneficial to reduce the switching activity at the expense of some additional control circuitry [Alidina94]. Assume a pipelined system for comparing the output of two numbers from a block of combinational logic as shown in Figure 4-27; the first pipeline stage is a combinational block and the next pipeline stage is a comparator which performs the function $A > B$, where A and B are generated in the first stage (i.e., from the combinational block). If the most significant bits, $A[N-1]$ and $B[N-1]$, are different then the computation of $A > B$ can be performed strictly from the MSB's and therefore the comparator logic for bits $A[N-1:0]$ and $B[N-2:0]$ is not required (and hence the logic can be powered down). If the data is assumed to be random (i.e there is a 50% chance that $A[N-1]$ and $B[N-1]$ are different), the power savings can be quite significant. One approach to accomplish this is to gate the clocks as shown in Figure 4-27. The XNOR output of the $A[N-1]$ and $B[N-1]$ is latched by a special register to generate a gated clock. This gated clock is then used to clock the lower order registers. Since the

latch to the lower bits is conditional, they must be made static. The standard TSPC register [Yuan89] is shown in Figure 4-28 as modified to support clock gating. As shown in the timing on the right side of Figure 4-28, with the extra PMOS device M1, the output can be forced to a ZERO during the low phase of the clock. Without this device, it will not be possible to generate rising edges for the gated clock on two consecutive rising edges of the system clock. Gated clocks have been used extensively in the system presented in Chapter 7.

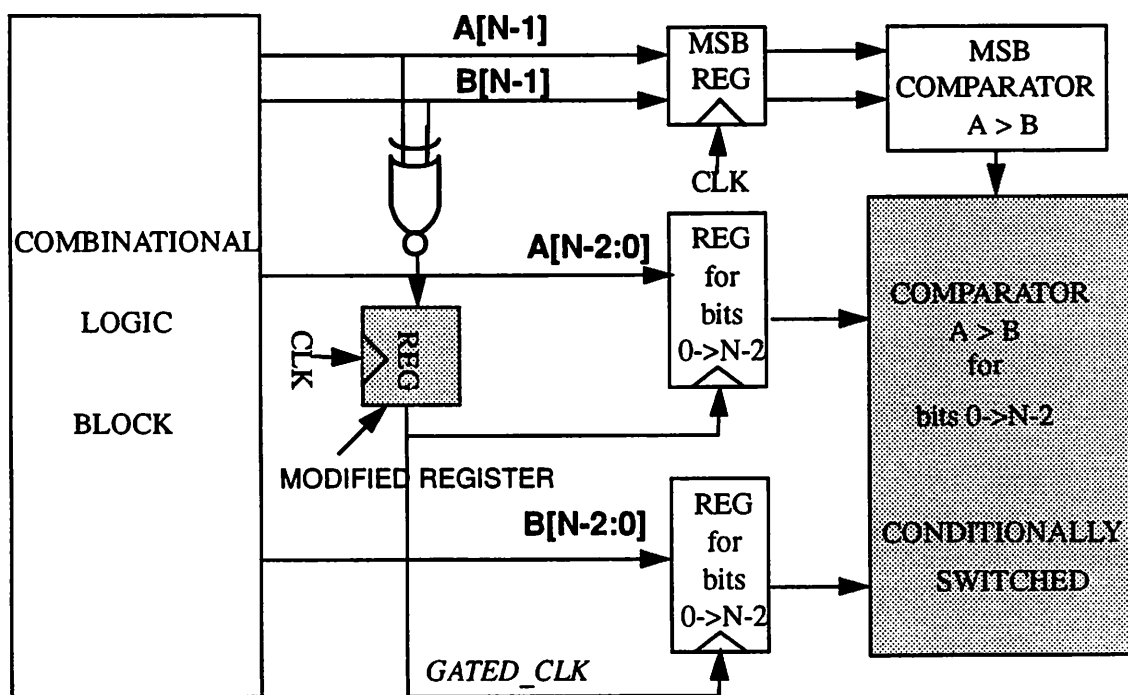


Figure 4-27 : Data dependent logic level shutdown.

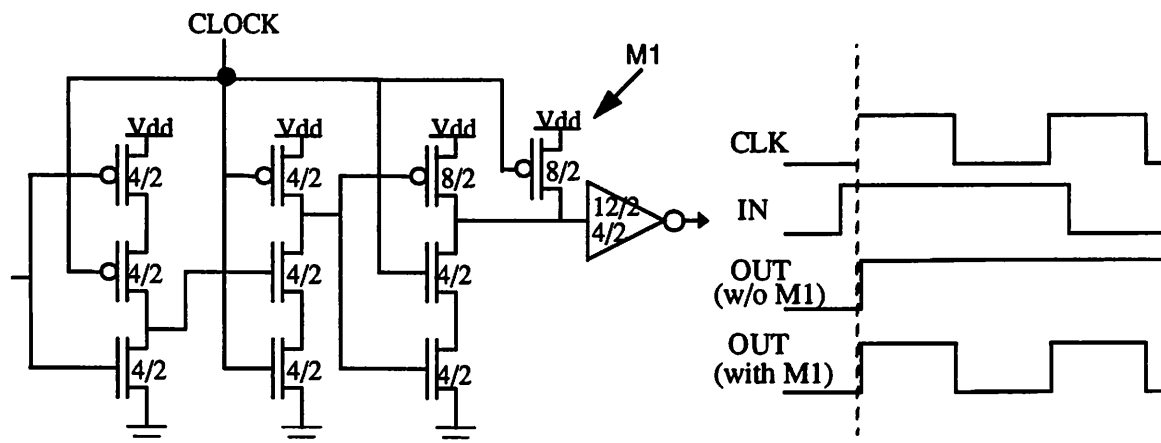


Figure 4-28 : Schematic of a modified TSPC latch that is used to generate gated clocks.

4.4 Circuit Optimization

There are a number of options available in choosing the basic circuit approach and topology for implementing various logic and arithmetic functions. Choices between static vs. dynamic implementations, passgate vs. conventional CMOS logic styles, and synchronous vs. asynchronous timing are just some of the options open to the system designer. At another level, as mentioned previously, there are also various structural choices for implementing a given logic function; for example, to implement an adder module one can utilize a ripple-carry, carry-select, or carry-lookahead topology. In this section, the trade-offs with respect to low power design between a selected set of circuit approaches will be discussed, followed by a discussion of some general issues and factors affecting the choice of logic family.

4.4.1 Dynamic Logic vs. Static Logic

The choice of using static or dynamic logic is dependent on many other criteria than just its low power performance, e.g. testability and ease of design. However, if only the low power performance is analyzed it would appear that dynamic logic has some inherent advantages in a number of areas including reduced switching activity due to hazards, elimination of short-circuit

dissipation and reduced parasitic node capacitances. Static logic has advantages since there is no pre-charge operation and charge-sharing does not exist. Below, each of these considerations will be discussed in more detail.

A. Spurious Transitions

As mentioned earlier static designs can exhibit spurious transitions due to finite propagation delays from one logic block to the next; i.e. a node can have multiple transitions in a single clock cycle before settling to the correct logic level. These spurious transitions dissipate extra power over that strictly required to perform the computation. The number of these extra transitions is a function of input patterns, internal state assignment in the logic design, delay skew, and logic depth. To be specific about the magnitude of this problem, an 8-bit ripple-carry adder with an uniformly distributed set of random input patterns, will typically consume an extra 30% in energy. Though it is possible with careful logic design to eliminate these transitions, dynamic logic intrinsically does not have this problem, since any node can undergo at most one power-consuming transition per clock cycle.

B. Short-circuit currents

Short circuit (direct-path) currents are found in static CMOS circuits. However, by sizing transistors for equal rise and fall times, the short-circuit component of the total power dissipated can be kept to less than 20% [Veendrick84] (typically < 5-10%) of the dynamic switching component. Dynamic logic does not exhibit this problem, except for those cases in which static pull-up devices are used to control charge sharing or when clock skew is significant.

C. Parasitic capacitance

Dynamic logic typically uses fewer transistors to implement a given logic function, which directly reduces the amount of capacitance being switched and thus has a direct impact on the power-delay product [Hodges88][Shoji88]. However, extra transistors may be required to insure that charge-

sharing does not result in incorrect evaluation.

D. Switching activity

The one area in which dynamic logic is at a distinct disadvantage is in its necessity for a precharge operation. Since in dynamic logic every node must be precharged every clock cycle, this means that some nodes are precharged only to be immediately discharged again as the node is evaluated, leading to a higher activity factor. As described in Chapter 2, a dynamic NOR gate has an activity of 0.75 assuming uniform inputs, while the activity factor for the static NOR counterpart will be only 3/16, excluding the component due to glitching. In general, gate activities will be different for static and dynamic logic and will depend on the type of operation being performed and the input signal probabilities. In addition, the clock buffers to drive the precharge transistors will also require power that is not needed in a static implementation.

E. Power-down modes

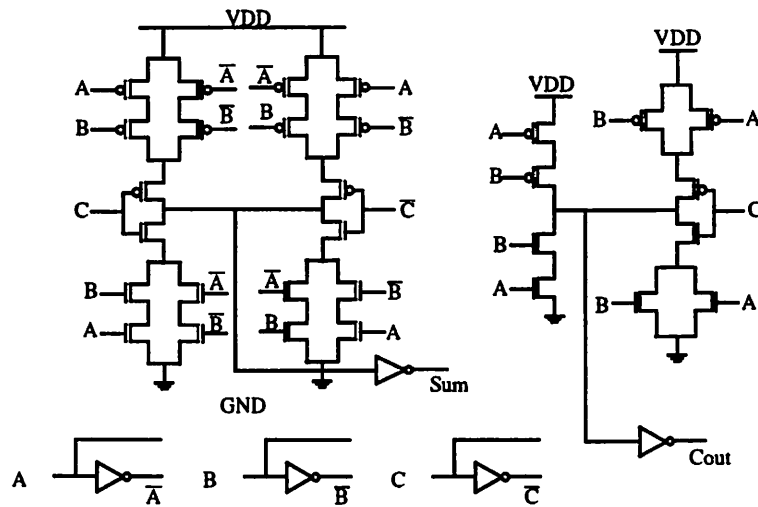
Lastly, power-down techniques achieved by disabling the clock signal have been used effectively in static circuits, but are not as well-suited for dynamic techniques. If the logic state is to be preserved during shut-down, a relatively small amount of extra circuitry must be added to the dynamic circuits to preserve the state, resulting in a slight increase in parasitic capacitance and slower speeds.

4.4.2 Pass Transistor Logic vs. Conventional CMOS Logic

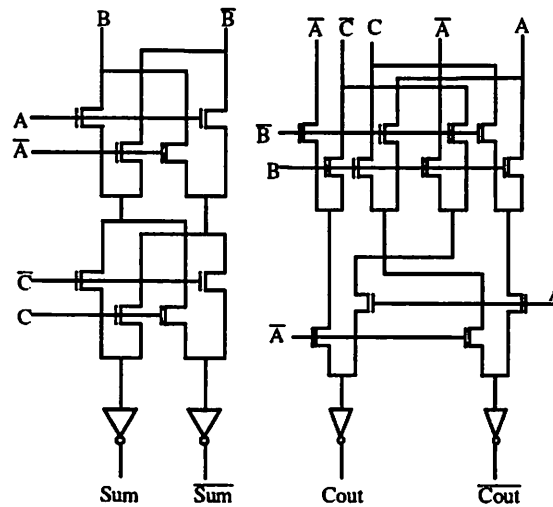
The type of logic style used (static vs. dynamic, pass gate vs. conventional CMOS) affects the physical capacitance in the circuit. The physical capacitance is a function of the number of transistors that are required to implement a given function. For example, one approach to reduce the physical capacitance is to use transfer gates over conventional CMOS gates to implement logic functions, as is used in the CPL (Complementary Passgate Logic) family. In Figure 4-29, the schematic of a typical static CMOS logic circuit for a full adder is shown along with a static CPL

version [Yano90]. The passgate design uses only a single transmission NMOS gate, instead of a full complementary passgate to reduce node capacitance. Passgate logic is attractive as fewer transistors are required to implement important logic functions, such as XOR's which only require 2 pass transistors in a CPL implementation. This particularly efficient implementation of an XOR is important since it is key to most arithmetic functions, permitting adders and multipliers to be created using a minimal number of devices. Likewise, multiplexers, registers, and other key building blocks are simplified using passgate designs.

However, a CPL implementation as shown in Figure 4-29 has two basic problems. First, the threshold drop across the single channel pass transistors results in reduced current drive and hence slower operation at reduced supply voltages; this is important for low-power design since it is desirable to operate at the lowest possible voltages levels. Second, since the "high" input voltage levels at the regenerative inverters is not V_{dd} , the PMOS device in the inverter is not fully turned off, and hence direct-path static power dissipation could be significant. To solve these problems, reduction of the threshold voltage has proven effective, although if taken too far will incur a cost in dissipation due to subthreshold leakage and reduced noise margins. The power dissipation for a passgate family adder with zero-threshold pass transistors at a supply voltage of 4V was reported to be 30% lower than a conventional static design due to the reduced node capacitance, with the difference being even more significant at lower supply voltages due to the reduction in voltage swing [Yano90].



Transistor count (conventional CMOS) : 40



Transistor count (CPL) : 28

Figure 4-29 : CMOS vs. Pass Gate Logic: adder example.

4.4.3 Synchronous vs. Asynchronous

In synchronous designs, the logic between registers is continuously computing every clock cycle based on its new inputs. To reduce the power in synchronous designs, it is important to minimize switching activity by powering down execution units when they are not performing “useful”

operations. This is an important concern since logic modules can be switching and consuming power even when they are not being actively utilized.

While the design of synchronous circuits requires special design effort and power-down circuitry to detect and shut down unused units, self-timed logic has inherent power-down of unused modules, since transitions occur only when requested. However, since self-timed implementations require the generation of a completion signal indicating the outputs of the logic module are valid, there is additional overhead circuitry. There are several circuit approaches to generate the requisite completion signal. One method is to use dual-rail coding, which is implicit in certain logic families such as the DCVSL [Jacobs90][Chu87]. The completion signal in a combinational macrocell made up of cascading DCVSL gates consists of simply ORing the outputs of only the last gate in the chain, leading to small overhead requirements. However, for each computation, dual-rail coding guarantees a switching event will occur since at least one of the outputs must evaluate to zero. We found that the dual rail DCVSL family consumes at least two times more in energy per input transition than a conventional static family. Hence self-timed implementations can prove to be expensive in terms of energy for datapaths that are continuously computing.

4.5 Physical Design

4.5.1 Silicon-On-Insulator Technology

Insulating substrates such as sapphire provide an alternate approach to conventional silicon substrate CMOS. An example of Silicon-on-sapphire CMOS is shown in Figure 4-30 [Uyemura92].

Sapphire is typically chosen since it is an excellent insulator and can be lattice-matched to silicon. An epitaxial layer of silicon is first grown on sapphire. Devices are formed on the epi layer and most interconnects are obtained from subsequent poly or metal patterning. The transistors are

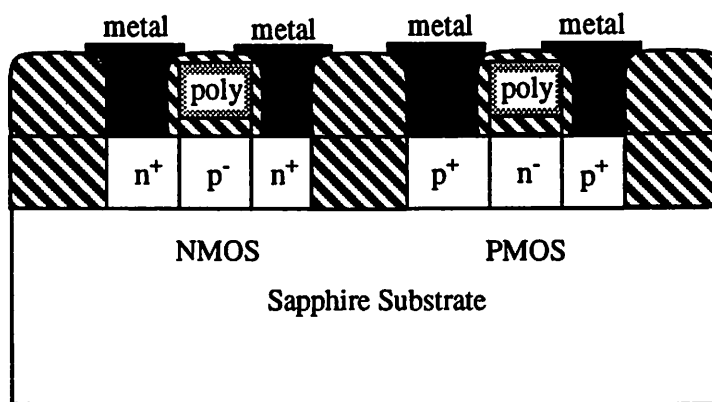


Figure 4-30 : Silicon-on-Sapphire CMOS.

physically separated, eliminating parasitic conduction and latchup. The key advantage of this technology for low-power applications is that the parasitic capacitances in the circuit layout are reduced, reducing the physical capacitance that has to be switched. The interconnect capacitance per unit area is

$$C_{int} = \frac{\epsilon_{int}}{x_{int}} \quad (\text{EQ 118})$$

where x_{int} is the thickness of the sapphire insulator, which is typically on the order of a millimeter. Using $\epsilon_{int} \cong 10.5\epsilon_0$ for sapphire gives $C_{int} = 100\text{pF}/\text{cm}^2$. This is much smaller than silicon substrates where $x_{int} \approx x_{FOX} \sim 0.1\mu\text{m}$.

The main drawbacks of Silicon-on-Sapphire which affect the cost and complexity are the need for a high quality sapphire substrate and the requirement for the silicon epitaxial layer. Recent research has demonstrated the feasibility of using oxide as the insulator.

4.5.2 Layout Optimization

Optimizing the layout to minimize the parasitics is important to minimizing the physical capacitance. Figure 4-31 three different approaches to laying out a large device that might be used for large buffers and clock drivers[Shoji88].

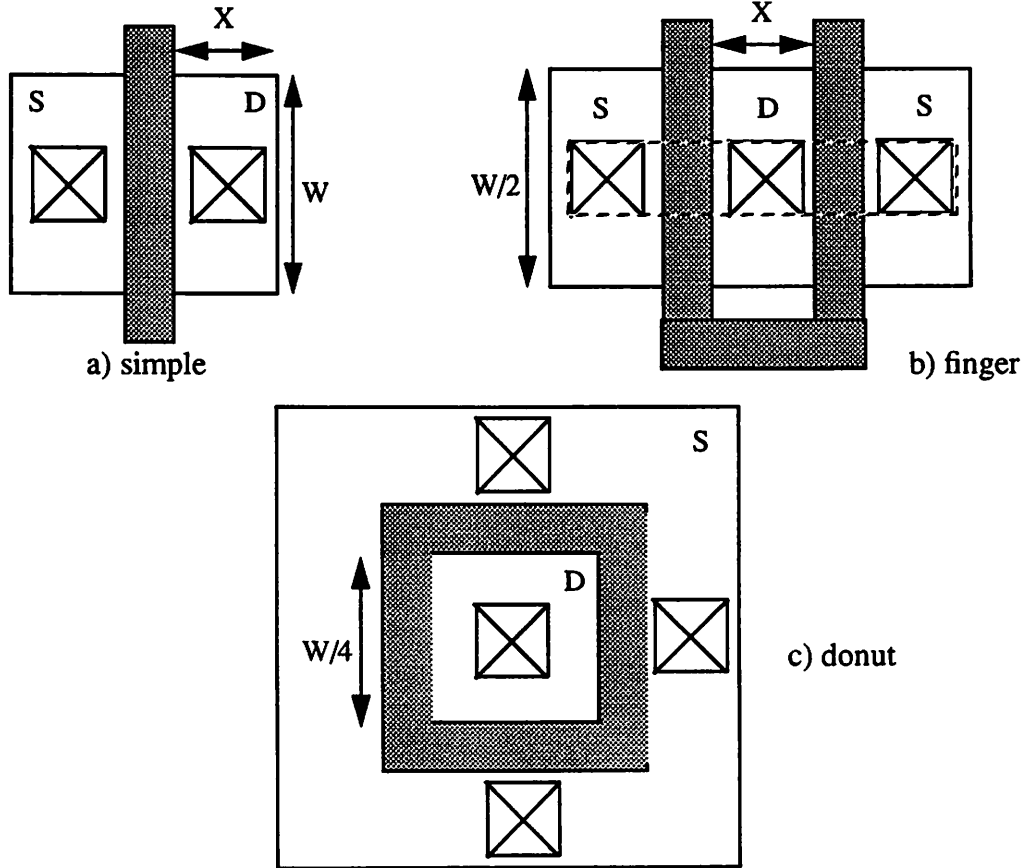


Figure 4-31 : Three layout approaches for a large W/L transistor.

These structures have the same gate capacitance but different drain diffusion capacitances. The first case is a straight forward layout and the capacitance is given from Equation 18,

$$C_{\text{drain}} = C_{\text{area}} WX + C_{\text{jsw}} (W+2X) \quad (\text{EQ 119})$$

where C_{area} is the area capacitance per unit area and C_{jsw} is the side-wall capacitance per unit length.

The second layout structure is a finger type structure that is commonly used to reduce the diffusion capacitance. Here, the side wall capacitance and the drain area are reduced compared to the simple layout. The capacitance is given by:

$$C_{\text{drain}} = C_{\text{area}} \cdot W/2 \cdot X + C_{\text{jsw}} (2X) \quad (\text{EQ 120})$$

This example layout shows only two fingers, but for large sized devices several fingers could be used.

Finally, the third structure is a ring type structure which eliminates any side-wall capacitance in the drain of the device. The drain capacitance is given by

$$C_{\text{drain}} = C_{\text{area}} \cdot W^2/16 \quad (\text{EQ 121})$$

Table 4-5 shows the results of drain capacitance for a device with $W/L = 20$ in a $1.2\mu\text{m}$ technology (i.e $W = 24\mu\text{m}$). For this technology $C_{\text{area}} = 0.3\text{fF}/\mu\text{m}^2$ and $C_{\text{jsw}} = 0.8\text{fF}/\mu\text{m}$ for an NMOS device. Also the minimum contact size is 4λ . As seen from the table, significant reduction in physical capacitance is possible through layout optimization.

Table 4-5 : The influence of layout optimization on physical capacitance.

Layout Style	Area μm^2	Perimeter contributing to sidewall in μm	C_{drain}
Simple	$5\lambda \times 24 = 72$	30	45.6fF
Finger	$6\lambda \times 12 = 43.2$	7.2	18.72fF
Ring	$6 \times 6 = 36$	0	10.8fF

Another optimization at the layout level is for regular tilable data-path structures, where the cells are of parameterized bit-widths. Here, all the data-signals are buses interconnecting the various cells in the datapath. To minimize the physical capacitance, it is important to minimize both the cell size, and the routing channels needed for the buses. The approach used in the low-power cell-library [Burd94] used for the designs described later in Chapters 6 and 7 was to route all buses in metal 2, and to not use any metal 2 for routing within a cell. Unfortunately, this forces some of the modules, such as the adder, to use poly for intracell routing, which has two to three times the capacitance per unit area. However, the inter-block bus routing becomes much more efficient, and the capacitance increase from using poly is small compared to the overall decrease attributable to

the more compact layout and smaller global buses. Each cell is 64λ high, which allows seven metal 2 feedthroughs over the cell. These are used both as I/O ports to the cell, and for the global data-path routing over cells. Utilizing this strategy reduced the area of several sample datapaths, on average, by 35% over the previously used data-path library not optimized for low-power. The area decrease of the datapath modules translates into a reduction of global routing wire length, and their associated parasitic capacitances.

4.5.3 Place and Route

At the layout level, the place and route should be optimized such that signals that have high switching activity (such as clocks) should be assigned short wires and signals with lower switching activities can be allowed progressively longer wires. Current design tools typically minimize the overall area or wire lengths given a timing constraint, which does not necessarily reduce the overall capacitance switched. Recently CAD efforts are starting to address the problem of physical design for low-power in which signal transition activities are used to drive place and route [Hirendu93][Chao94]. An example of the benefits of this optimization is presented in Chapter 6.

4.6 Summary

In the previous chapter, an aggressive voltage scaling strategy was used to minimize the energy to perform a given function. Another approach to low power design is to reduce the switching activity to the minimal level required to perform the computation. In CMOS circuits, since energy is only consumed when capacitance is being switched, power can be reduced by minimizing this capacitance at all levels of the design including algorithms, architectures, logic design, circuit design, and physical design. One key attribute exploited is that signal processing applications, unlike general purpose applications, exhibit a lot of temporal correlation in data and this is used at all the levels to minimize the capacitance switched.

At the algorithmic level, the basic computational complexity must be optimized. The switched capacitance can be minimized through reducing the number of operations to perform a given function (e.g. using hardwired shift-add operations instead of real multiplications or using algebraic transformations to minimize complexity), optimized data representation (binary vs. gray-coding, two's complement vs. sign-magnitude, etc.), and minimizing bitwidths (as will be discussed in the next chapter). Algorithmic optimization typically has the greatest impact on minimizing the switched capacitance.

At the architecture level, there are various degrees of freedom in minimizing the switched capacitance. This includes using sign-magnitude for multiply-accumulate architectures to minimize the number of transitions, ordering of operations to change dynamic range, balancing signal paths at the module level to minimize glitching transitions, and optimizing resource assignment by keeping correlated data on the same hardware and uncorrelated data on different hardware units.

At the logic level, logic minimization and logic-level power down are used to minimize the switched capacitance. At the circuit level, pass transistor logic can be used to minimize the physical capacitance of the circuit to implement function like adders, multiplier, etc. Also, at the circuit level, dynamic vs. static logic or synchronous vs. self-timed styles can be explored. At the physical design level, the place and route can be optimized such that signal that have high switching activity can be assigned short wires while signals that have low switching activity can be allowed to have long wires.

CHAPTER 5

Computer Aided Design Tools

In the previous chapters, approaches were presented to minimize the power consumption through supply voltage scaling and through the reduction of switched capacitance. The focus of this chapter is on automatically finding computational structures that result in the lowest power consumption for DSP applications that have a specific throughput constraint given a high-level algorithmic specification. The basic approach is to scan the design space utilizing various algorithmic flowgraph transformations, high-level power estimation, and efficient heuristic/probabilistic search mechanisms. While algorithmic transformations have been successfully applied in high-level synthesis with the goal of optimizing speed and/or area, they have not addressed the problem of minimizing power.

There are two approaches taken to explore the algorithmic design space to minimize power consumption. First, is the exploitation of concurrency which enables circuits to operate at the lowest possible voltage without loss in functional throughput. Second, computational structures are used that minimize the effective capacitance that is switched at a fixed voltage: through reductions in the number of operations, the interconnect capacitance, the glitching activity, and

internal bit widths and using operations that require less energy per computation.

5.1 Previous and Concurrent Work

5.1.1 Power Estimation

Most of the previous or concurrent work done in power estimation falls under four categories: circuit-level estimation, switch-level estimation, gate-level probabilistic estimation, and architectural power estimation. The primary trade-off between these approaches is the computational complexity vs. accuracy. Circuit-level approaches result in the most accurate estimates, while being the most computationally intensive.

Continuous Time Circuit-Level Power Estimation

At the lowest level of design, power can be estimated using a circuit simulator such as SPICE. The design is described at a very low-level and the simulator accounts for short-circuit and leakage components of power. The major problem with this approach is the long simulation time which limits the number of patterns that can be applied. Therefore, this approach is only viable for analyzing small circuits (e.g. when characterizing library cells).

Figure 5-1 shows a circuit that can be used to measure the energy consumed by a circuit using the SPICE simulator. The current drawn by the circuit under test from the power supply is monitored by the current controlled current source and integrated on the capacitor C . The resistance R is only provided for DC-convergence reasons and should be chosen as high as possible to minimize leakage. Through proper choice of parameters for the capacitor, C and the gain for the current controller current source, k , the voltage on the capacitor can be made equal to the average energy drawn from the supply.

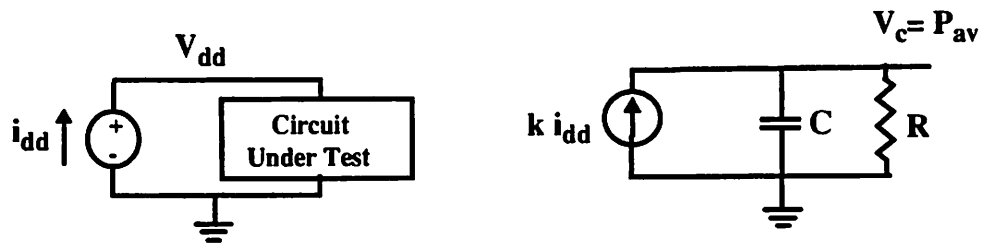


Figure 5-1 : Circuit for measuring the power-delay product.

The voltage across the capacitor is given below (under the assumption that the initial voltage on the capacitor C is zero):

$$C \frac{dV_c}{dt} = k i_{dd}(t) \quad (\text{EQ 122})$$

$$V_c = \frac{k}{C} \int_0^T i_{dd}(t) dt \quad (\text{EQ 123})$$

The energy consumed by the circuit is:

$$E = V_{dd} \int_0^T i_{dd}(t) dt \quad (\text{EQ 124})$$

Therefore, from Equations 123 and 124, by setting $V_{dd} = k/C$, the voltage across the capacitor will equal the energy consumed in the interval $[0, T]$.

Figure 5-2 shows an example of estimating the energy consumed for a 1-bit full adder circuit. The integrating capacitor value is chosen to be 1pF and k is chosen to be V_{dd} (=1.5V for this example). Therefore, the voltage across the capacitor represents the cumulative energy (in pJ) for the 1-bit full-adder circuit. The graph in Figure 5-2 shows the voltage across the capacitor for a sequence of input patterns (IN0 IN1 CIN). The jumps in the voltage curve represent the amount of energy drawn for that particular input transition. For example, for the input transition 111 \rightarrow 000, the change in voltage across the capacitor is 0.84V (the energy drawn for this transition is 0.84pJ).

From this, the capacitance switched for this transition can be determined as $0.84\text{pJ}/(1.5)^2 = 0.37\text{pF}$. The dependence of power on data sequencing is obvious from this figure as different transitions draw different amounts of energy; for example, 100 \rightarrow 010 does not draw very much energy since neither output switches, while 111 \rightarrow 000 draws a lot of energy since both outputs switch. The average energy drawn per transition is $3.64\text{pJ}/8$ (total energy/number of input transitions) = 0.44pJ .

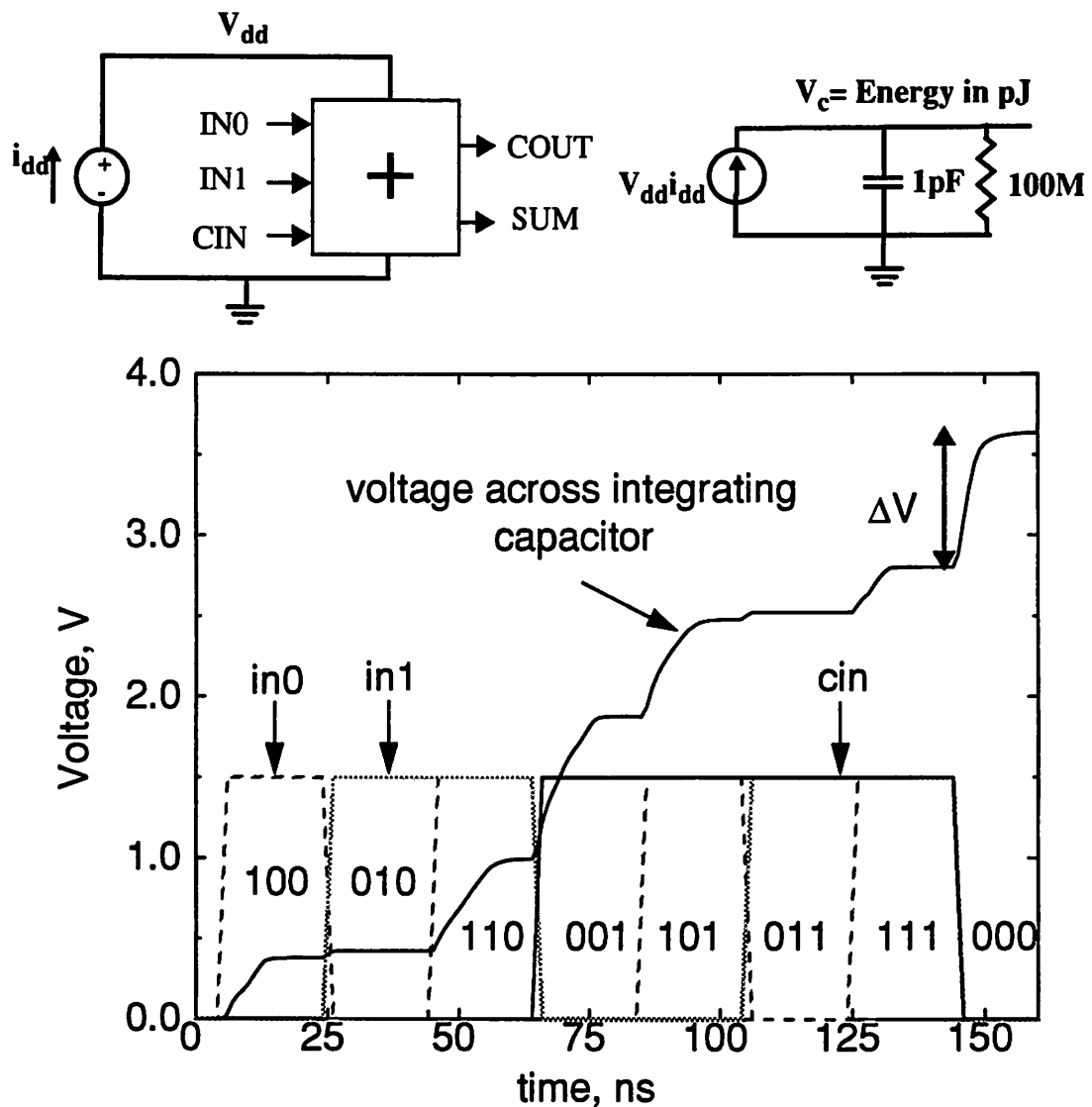


Figure 5-2 : Measuring capacitance switched for a 1-bit full-adder example.

Switch-level Power Estimation

An approach for estimating the power consumption in CMOS circuits using a switch-level simulator is presented in [Kimura91]. The basic idea is to monitor the number of times each node in the circuit transitions during the simulation period. C_{avg} is given by $\sum N_i / N C_i$, where N_i is the total number of power consuming transitions for node i , N is the number of simulation cycles, and C_i is the physical capacitance of node i . This approach using the IRSIM simulator [Salz89] was used estimate power at the layout level. This analysis ignores short-circuit and leakage power components. The results from a few fabricated chips, indicate that the predicted power (with calibrated models from test chips) from IRSIM is within 30% of the measured power. A commercial version, called PowerMill from Epic Design, also exists that takes into account the extra power due to finite rise/fall times [Deng94].

An important parameter for power estimation through simulation is the number of simulation cycles that are needed for accurate estimation of the switched capacitance. In the case of SPICE simulation, only a small number of patterns can be applied due to the time consuming nature of the simulator. Also, only simulation of small circuits is possible, restricting the application to the characterization of cell-libraries. Switch-level simulation is significantly faster and large circuits (whole chips) can be simulated. Figure 5-3 shows the average capacitance switched per cycle as a function of the number of sample periods simulated for an FIR filter. For this example which has no feedback and for the input data pattern used, a good indication of the average power consumption can be obtained by simulating for 50-100 cycles. In general, determining the number of cycles is a non-trivial task, strongly dependent on the topology (signal correlations) and data patterns. A methodology and tool to determine the minimum number of simulation cycles required has been developed [VanOostende93].

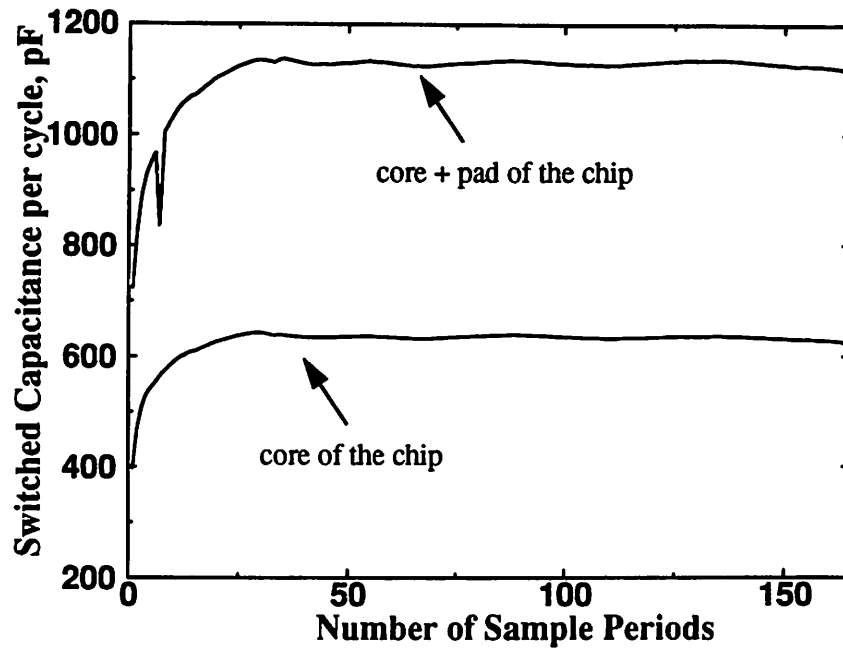


Figure 5-3 : Switched capacitance vs. number of simulated periods.

Gate-level Probabilistic Power Estimation

Gate-level probabilistic approaches estimate the internal node activities of a network given the distribution of the input signals [Cirit87][Najm90][Ghosh92]. Once the signal probability for each node in the network is determined, the total average capacitance switched is then estimated as

$$C_{total} = \sum_{i=1}^{numnodes} (p_{i0})(1-p_{i0}) \cdot C_i \quad (\text{EQ 125})$$

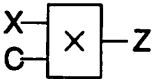
where $numnodes$ is the total number of nodes in the network, p_{i0} is the probability that node i will be in the *ZERO* state, and C_i is the physical capacitance associated with node i . The total power is then estimated as $C_{total} * V_{dd}^2 * f_{clk}$. Most gate-level simulators typically ignore the interconnect component of loading capacitance and model the capacitance as the sum of the output capacitance of the driver plus the input gate capacitance all the gates being driven. Two other major problems with these simulators are in modeling reconvergent fanout (which introduces

correlations between signals) and glitching activity (which arises due to signal skew). Significant efforts to address these problems have been recently made [Ghosh92].

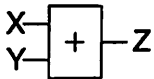
Architectural Power Estimation

Using concepts abstracted from the gate level, an architectural estimation technique has been developed based on high level statistics such as mean, variance, and autocorrelation [Landman93]. While, the above mentioned gate level tools focus on the power consumed by boolean logic gates (NOT, AND, OR, etc.) as a function of their input bit probabilities, the architectural tool considers module (adder, multiplier, register, etc.) power consumption as determined by input word statistics.

As described in Chapter 2, there exists a direct relationship between bit level probabilities and word level statistics. The transition probability as a function of the bit number can be represented as a simple piecewise linear model with breakpoints BP0 and BP1. The important features of this model - the values of the breakpoints and the signal and transition probabilities - can all be extracted from three statistical parameters: the mean, μ ; the variance, σ^2 ; and the lag one correlation coefficient, $\rho_1 = \text{cov}(X_t, X_{t+1})/\sigma^2$. Similar to gate level techniques, given the statistics of the module inputs, the statistics of the outputs are calculated by statistical propagating; this way the model parameters (μ , σ^2 , and ρ_1) for each bus in the architecture can be derived. The top half of Figure 5-4 presents the appropriate propagation equations for the case of an addition and a constant multiplication (the two key operations of linear, time-invariant systems). Once again, the issue of reconvergent fan-out tends to complicate matters by introducing correlations between module inputs which is not handled by these equations; however, this is overcome by abstracting heuristic techniques similar to those applied at the gate level. For example, the revised equations at the bottom of Figure 5-4 account for signal correlations.



$$\begin{aligned} \mu_z &= C\mu_x \\ \sigma_z^2 &= C^2\sigma_x^2 \\ \rho_1^z &= \rho_1^x \end{aligned}$$



$$\begin{aligned} \mu_z &= \mu_x + \mu_y \\ \sigma_z^2 &= \sigma_x^2 + \sigma_y^2 \\ \rho_1^z &= (\rho_1^x\sigma_x^2 + \rho_1^y\sigma_y^2)/\sigma_z^2 \end{aligned}$$

} if $X \perp Y$

Revised Adder Equations for Correlated Inputs:

$$\begin{aligned} \mu_z &= \mu_x + \mu_y \\ \sigma_z^2 &= \sigma_x^2 + 2\rho^{xy}\sigma_x\sigma_y + \sigma_y^2 \\ \rho_1^z &= (\rho_1^x\sigma_x^2 + [\rho_1^{xy} + \rho_1^{yx}]\sigma_x\sigma_y + \rho_1^y\sigma_y^2)/\sigma_z^2 \end{aligned}$$

Figure 5-4 : Statistical parameter propagation [Landman93].

The approaches mentioned above estimate power consumption from a low-level of abstraction. To use these approaches in a high-level synthesis framework, the high-level representation of the algorithm has to be mapped to a low-level description (gate or transistor level), which is very time consuming. Even going to the architecture level (which involves allocation/assignment/scheduling) is too time consuming if the goal is to explore 100's or 1000's of possible structures in a time efficient manner. The estimation time for each new topology is time-consuming itself. Hence, power must be estimated efficiently from an even higher level of abstraction. In this chapter, power estimation techniques will be described which estimate power from an algorithmic level so that the design space can quickly and efficiently explored.

5.1.2 High-level Transformations

Over the last few years, several high-level synthesis systems have incorporated comprehensive sets of transformations, coupled with powerful optimization strategies. Example systems with elaborate applications of transformations are Flamel [Trickey87], SAW [Walker89], SPAID [Haroun89], and HYPER [Rabaey91a].

Among the set of transformations used by the Flamel design system are loop transformations, height reduction and constant propagation. SAW uses among other transformations in-line

expansion, dead code elimination, four types of transformations for conditional statements and pipelining as supporting steps during the behavioral and structural partitioning. The SPAID system set of transformations includes retiming and pipelining, interleaving, substitution of multiplications with constants by addition and shifts and algebraic transformations. HYPER uses more than 20 different transformations whose application is supported by several probabilistic optimization algorithms. All systems provide interactive frameworks where the designer explores the influence of the transformation mechanism or the optimization algorithms for a specific transformation.

The systems described above use transformations to optimize design parameters such as area and throughput. The system presented in this chapter will address the problem of how high-level flowgraph transformations can be used to reduce the power consumption of the VLSI implementation.

5.2 Application of Transformations to Minimize Power

Transformations are changes to the computational structure in a manner that the input/output behavior is preserved. The number and type of computational modules, their interconnection and their sequencing of operation are optimized. The use of transformations makes it possible to explore a number of alternative architectures and to choose those which result in the lowest power. We will use the control-data flow graph format to represent computation [Rabaey91a], in which nodes represent operations, and edges represent data and control dependencies. Transformations have primarily been used until now to optimize either the implementation area or the system throughput. The goal here is to optimize a different function, namely the power dissipation of the final circuit while meeting the functional throughput of the system. Two key approaches are used to reduce power for a fixed throughput: reducing the supply voltage by utilizing speed-up transformations (section 5.2.1) and reducing the effective capacitance being switched (sections 5.2.2 - 5.2.5).

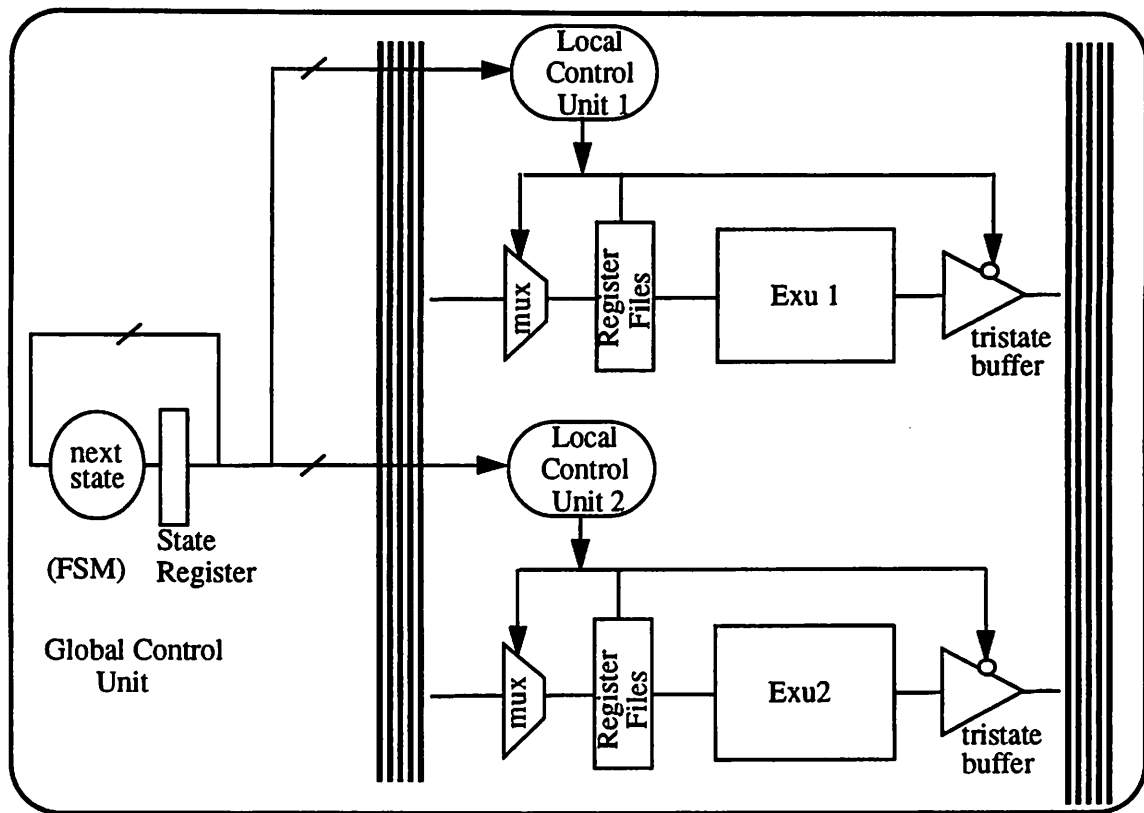


Figure 5-5 : Hardware model used in HYPER.

Figure 5-5 shows the hardware model used in HYPER. A time-multiplexed hardware model is used in which multiple operations on a data-control flowgraph can be mapped to the same hardware unit. A datapath module contains execution units, register files, multiplexors, and buffers. Data stored in the register files is read out to the execution unit whose result is written back into register files in the same datapath or in other datapaths. The register file may be preceded by multiplexors depending on whether the execution unit receives data from different sources at different control cycles in the sample period. The execution units on different datapaths communicate through global busses and the outputs of the execution units are tri-stated if two different units share a common global bus. A distributed control approach (which contains a global and local controller) is used to sequence through the operations in the control cycles. This section addresses the estimation the capacitance components and the supply voltage. The critical path will be stated in the number of control steps, which is the number of cycles required to implement the

given function.

5.2.1 Speed-up transformations

This is probably the single most important type of transformation for power reduction. It is not only the most common type of transformation, but often has the strongest impact on power. The basic idea is to reduce the number of control steps, so that slower control clock cycles can be used for a fixed throughput, allowing for a reduction in supply voltage. Several examples of using this technique were described in Chapter 3. The reduction in control step requirements is most often possible due to the exploitation of concurrency. Many transformations profoundly affect the amount of concurrency in the computation. This includes retiming/pipelining, algebraic transformations and loop transformations.

5.2.2 Operation Reduction

The most obvious approach to reduce the switching capacitance, is to reduce the number of operations (and hence the number of switching events) in the data control flow graph. While reducing the operation count typically has the effect of reducing the effective capacitance, the effect on its critical path is case dependent. To illustrate this trade-off, consider evaluating second and third order polynomials. Computation of polynomials is very common in digital signal processing, and Horner's scheme (the final structure in our examples) is often suggested in filter design and FFT calculations when very few frequency components are needed [Goertzel68].

First we will analyze the second order polynomial $X^2 + AX + B$. The left side of Figure 5-6 shows the straightforward implementation which requires two multiplications and two additions and has a critical path of 3 (assuming that each operation takes one control cycle). On the right side of Figure 5-6, a transformed version having a different computational structure (obtained using algebraic transformations) is shown. The transformed graph has the same critical path as the initial solution and therefore the two solutions will have the same throughput at any given supply

voltage. However, the transformed flowgraph has one less multiplication, and therefore has a lower capacitance and power.

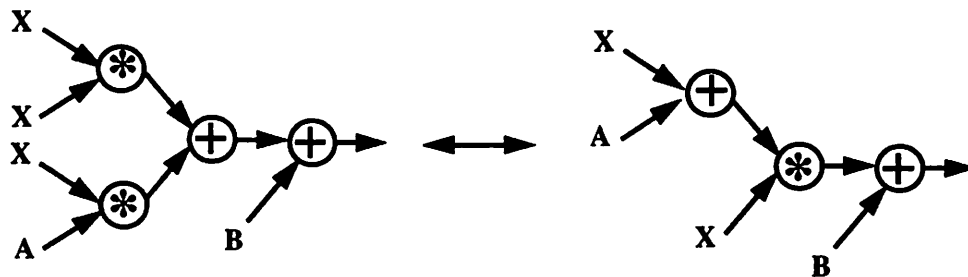


Figure 5-6 : Reducing capacitance while maintaining throughput.

Figure 5-7 illustrates a situation where a significant reduction in the number of operations is achieved at the expense of a longer critical path. This example once again involves the computation of a polynomial, this time of the form X^3+AX^2+BX+C . Again, by applying algebraic transformations we can transform the computation to the Horner's scheme. The number of multiplications reduces by two, resulting in a reduction of the effective capacitance. However, the critical path is increased from 4 to 5, dictating a higher supply voltage than in the initial flowgraph for the same computational throughput. Once again, this example shows that a transformation can have different effects on capacitance and voltage, making the associated power minimization task a difficult optimization problem.

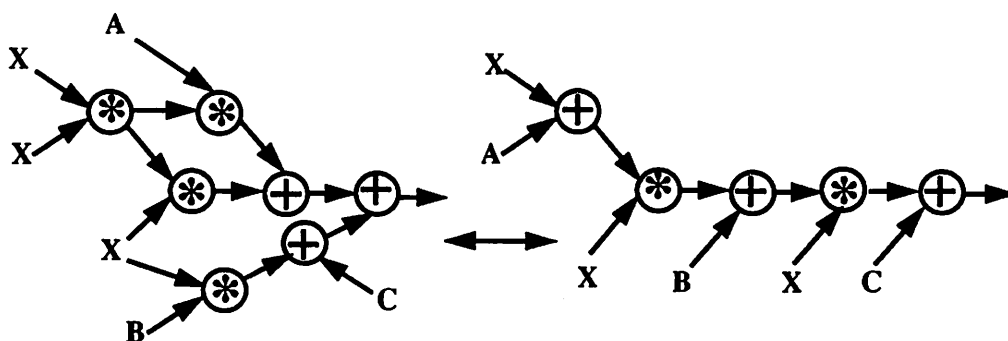


Figure 5-7 : Reducing capacitance at the expense of a higher supply voltage.

Transformations which directly reduce the number of operations in a data control flow graph include: common subexpression elimination, manifest expression elimination, and distributivity.

5.2.3 Operation Substitution

Certain operations inherently require less energy per computation than other operations. A prime example of transformation which explores this trade-off is strength reduction, often used in software compilers, in which multiplications are substituted by additions [Aho77]. Although this situation is not as common as the ones presented in the previous section, sometimes it is possible to achieve significant savings using this type of trade-off.

Unfortunately, this type of transformation often comes at the expense of an increase in the critical path length. This point is illustrated in Figure 5-8, which shows the frequently used application of redundancy manipulation, distributivity and common subexpression in complex number multiplication. A_i and A_r are constants. While the second implementation has a lower effective capacitance, it has a longer critical path. Fixing the available time to be a constant, we see that the second implementation requires at least three control cycles (assuming each operation takes one control cycle), while the first one requires only two. This implies that the voltage in the first implementation can be dropped lower than the voltage in the second implementation, since the same computational throughput can be met with a 50% slower clock rate.

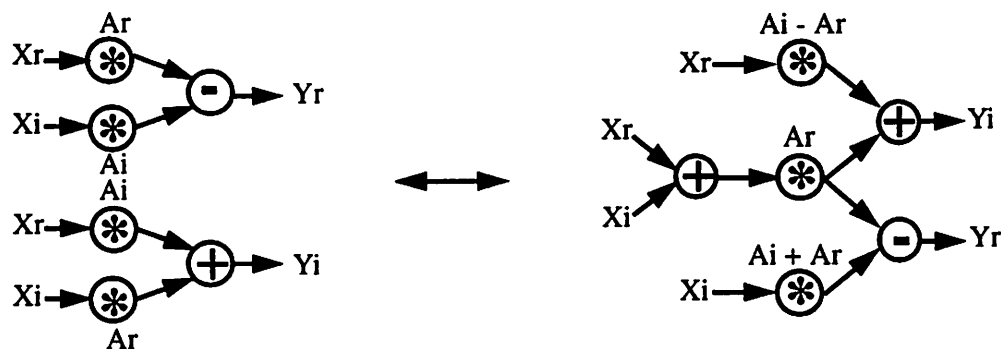


Figure 5-8 : Trading multiplication for an addition.

A powerful transformation in this category is conversion of multiplications with constants into shift-add operations (also described in Chapter 4). Table 5-1 and Table 5-2 shows the power breakdown for a 11 tap FIR filter before and after this transformation. The power consumed by the execution units is reduced to one-eighth of the original power and the power consumed in the registers

is also reduced. A small penalty is paid in the controller but there is a gain in the interconnect power due to a reduction in the length of the bus lines (since the final implementation has a smaller area), resulting in a total power savings of 62%.

Table 5-1 : Breakdown of power consumed for a 11 tap FIR filter before constant multiplications.

Component	Switched Capacitance, pF	Energy @ 2.5V, nJ	%
Exu	739.65	4.62	64.80
Registers & Clock	179.57	1.12	15.73
Control	65.45	0.41	5.73
Bus	156.69	0.98	13.74
Total	1141.36	7.13	100.00

Table 5-2 : Breakdown of power consumed for a 11 tap FIR filter after constant multiplications.

Component	Switched Capacitance, pF	Energy @ 2.5V, nJ	%
Exu	93.07	0.58	21.63
Registers & Clock	161.4	1.00	37.50
Control	83.79	0.52	19.47
Bus	92.10	0.58	21.40
Total	430.36	2.69	100.00

5.2.4 Resource Utilization

It is often possible to reduce the required amount of hardware, while preserving the number of control steps [Potkonjak91]. This is possible because after certain transformations, operations are more uniformly distributed over the available time, resulting in a denser schedule. These transformations include retiming for resource utilization, associativity, distributivity and commutativity. Figure 5-9 shows the result of applying retiming on a second order IIR filter. Both

the transformed graphs, 1 and 2, are obtained from retiming and have a critical path of 3; however, the transformed graph 2 can be scheduled with only 2 multipliers while the first needs 4 multipliers (since all the four multiplications can be performed only in the 3rd control step).

Given that there is a degree of freedom in choosing the amount of resources used for a fixed throughput (i.e. at a fixed supply voltage), the question then becomes is the solution with the minimal amount of resources, as chosen for the minimal area implementation, the “best” for low-power. On one hand, reducing the amount of resources can reduce the wiring capacitance since there are fewer interconnects and/or fewer functional elements and registers, which are obstacles during floorplanning and routing. However, the amount of multiplexors and control logic circuitry will typically increase with more time-sharing of resources. Therefore the optimization strategy (and hence the power estimation model) must take into account the trade-off between interconnect capacitance and control circuitry (which determines the effective capacitance being switched).

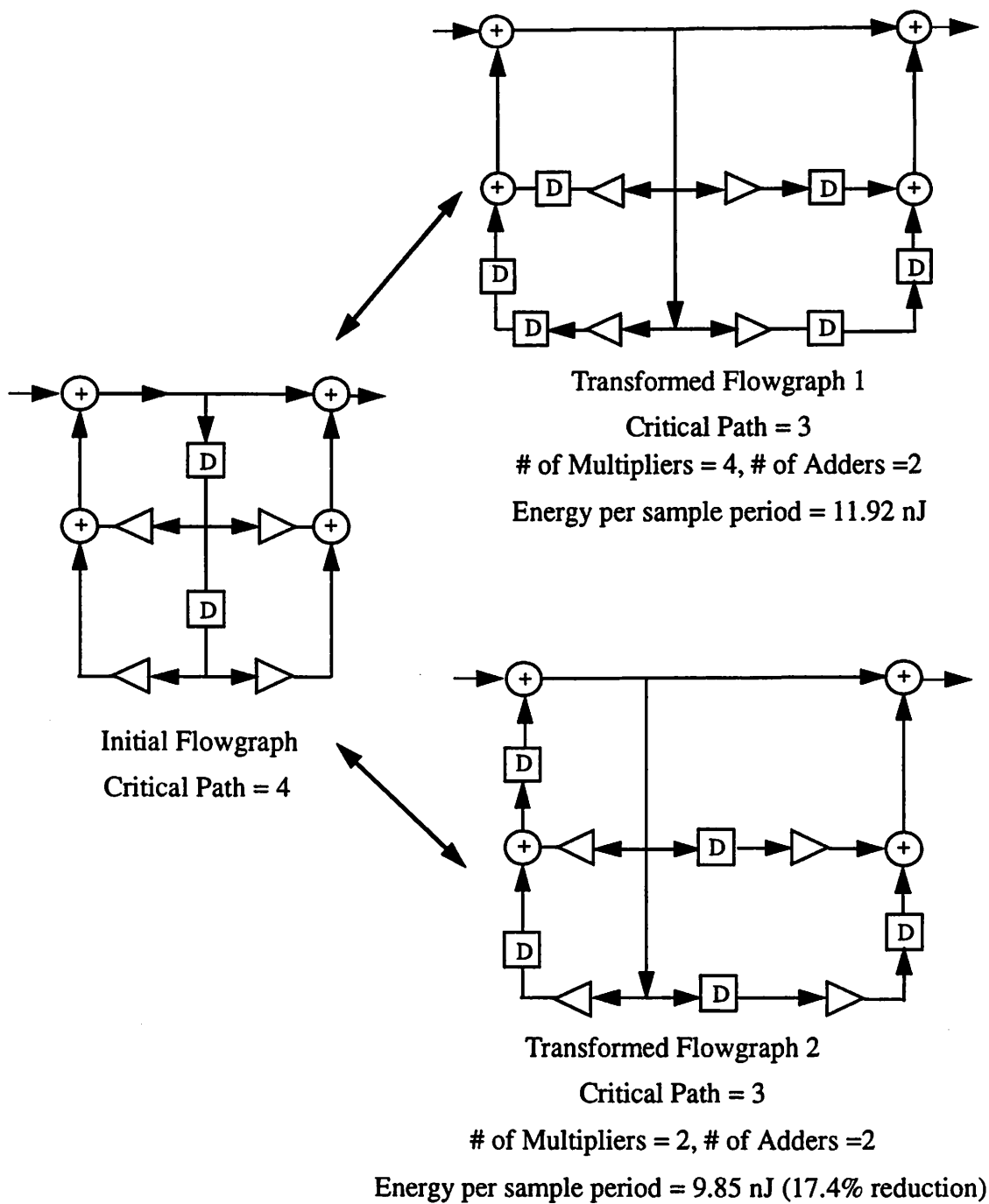


Figure 5-9 : Retiming for improving resource utilization: 2nd order IIR filter.

5.2.5 Wordlength Reduction

The number of bits used strongly affects all key parameters of a design, including speed, area and power. It is desirable to minimize the number of bits during power optimization for at least three reasons:

- fewer bits result in fewer switching events and therefore lower capacitance.
- fewer bits imply that the functional operations can be done faster, and therefore the voltage can be reduced while keeping the throughput constant.
- fewer bits not only reduce the number of transfer lines, but also reduce the average interconnect length and capacitance.

The influence of various transformation on numerical stability (and therefore the required wordlength) varies a lot. While some transformations, for example retiming, pipelining and commutativity, do not affect wordlength, associativity and distributivity often have a dramatic influence [Golberg91]. In some cases, it is possible to reduce both the number of power expensive operations and the required wordlength. In other cases, however, a reduction in wordlength comes at the expense of an increased number of operations.

To illustrate the importance of wordlength optimization, consider the direct form and parallel form implementations of an 8th order Avenhaus bandpass filter (as shown in Figure 5-10a and 5-10b). The critical path of the direct form, after multiplication to shift and addition substitution is 20 clock cycles while the critical path of the parallel form is 28 cycles, assuming that each operation, in both cases, takes one cycle. However, the numerical stability of the parallel form is significantly higher, resulting in wordlength requirements of only 11 bits, while the direct form solution requires 23 bits. The effective critical path for the direct form solution is 980 ns, while parallel form has a critical path of only 610 ns. Therefore, taking wordlength requirements into consideration, changes the situation from one where the critical path in the parallel form is almost 50% longer, to one where the critical path in the direct form is more than 50% longer. Similarly, a

detailed analysis of these two alternatives shows that the final implementation of the parallel filter is reduced by a factor of four for a given throughput.

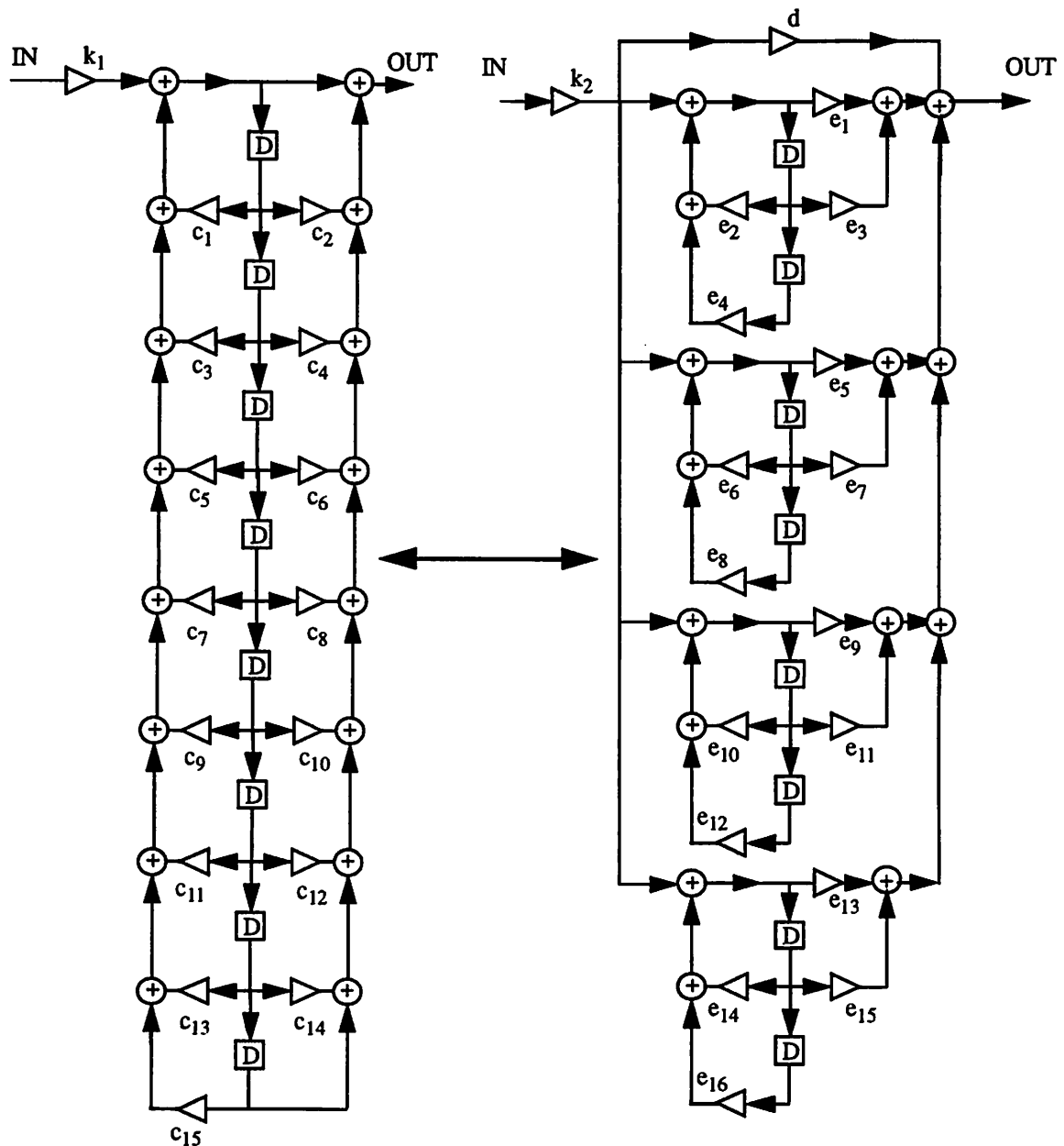


Figure 5-10 : Direct form structure vs. Parallel form structure: 8th order Avenhous filter.

The results from section 5.2 can be summarized in the following requirements for the application of transformations for power reduction:

-
- Efficient implementation of known and new transformations so that power is an explicit part of the cost function and the development of the cost function itself.
 - Development of a transformation framework and search mechanism which will determine the order and extent to which the transformations are applied so that final result is globally optimal.

5.3 Cost Function

The goal is to develop an objective function that is highly correlated to the final (and unknown) power dissipation of the circuit. The objective function should be very easy to compute since it has to be evaluated many times during the optimization process. A detailed estimation, while being accurate, will require hardware mapping and compilation steps to convert a flowgraph to layout, making it impractical during the optimization process. Hence a model correlated to the power must be developed strictly from the flowgraph level. This involves a statistical study of the effects of various high level parameters on interconnect capacitance, control capacitance, etc.

The power consumption of a circuit (represented in a control data flowgraph) is given by:

$$P_{\text{total}} = C_{\text{total}} * V^2 * f_{\text{sampling}} \quad (\text{EQ 126})$$

The goal of power optimization in this work is to keep the throughput constant (i.e. f_{sampling} is fixed) by allowing the supply voltage to vary. For a fixed sample period, power optimization is equivalent to minimizing the total energy switched, $C_{\text{total}} V^2$, where V is appropriate voltage required to meet the throughput rate.

5.3.1 Capacitance estimate

The total capacitance switched depends on four components:

$$C_{\text{total}} = C_{\text{exu}} + C_{\text{registers}} + C_{\text{interconnect}} + C_{\text{control}} \quad (\text{EQ 127})$$

The capacitance estimation is built on top of an existing estimation routine in HYPER that determines bounds and activity of various execution, register and interconnect components as well

as the implementation area [Rabaey91b, Schultz92]. The details of the capacitance estimation routines are described below.

Execution Units

The capacitance switched by the various execution units is estimated by multiplying (over all types of units utilized) the number of times the operation performed by the unit occurs per sample period with the average capacitance per access of the unit type. The total capacitance is hence given by:

$$C_{exu} = \sum_{i=1}^{numtypes} N_i \cdot C_i \quad (\text{EQ 128})$$

where *numtypes* is the total number of operation types, N_i the number the times the operation of type *i* is performed per sample period (or the activity), and C_i is average capacitance per execution of operation type *i*. The average capacitance per execution has been characterized for the various modules (through SPICE and IRSIM simulations, using models that were calibrated with results from experimental measurements) for a uniformly distributed set of inputs. In general the probabilities are not uniform, however, this assumption is made to simplify the cost evaluation. The capacitance values are parameterized as a function of bit-width and are accessed when computing the power contributed due to the execution units. Table 5-3 shows the high-level capacitance models for various modules in the hardware library.

Table 5-3 : Capacitance models for some of the library hardware units for a 1.2 μ m CMOS technology.

Component	Capacitance Model (in fF)	Parameters
ripple adder	$-46 + 151 N$	N: bitwidth
carry select	$-158 + 214 N$	N: bitwidth
comparater	$181 N$	N:bitwidth
multiplier	$253 * N1 * N2$	N1,N2: input bitwidths
shifter	$N * 28.7 + \log(M+1) * (43.8 + 12.2N + .06 N^2 + 0.24MN - 0.18 SN)$	M: Maximum shift allowed N: bitwidth S: Shift

Table 5-3 : Capacitance models for some of the library hardware units for a 1.2 μ m CMOS technology.

Component	Capacitance Model (in fF)	Parameters
register file	$87 + 51 R + 35 N + 8 R N$	N: bitwidth R: number of register

The capacitance per access for a ripple carry adder and a carry-select adder as a function of the bitwidth are shown in Figure 5-11.

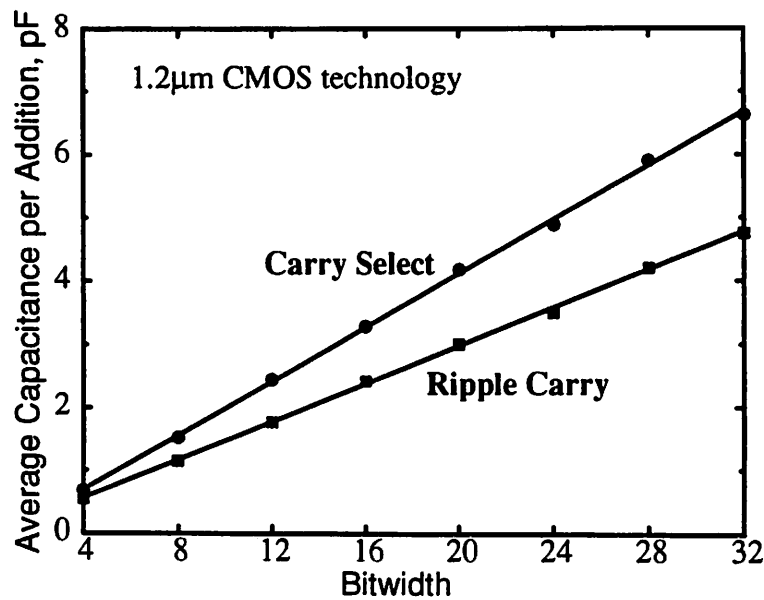


Figure 5-11 : Average capacitance per addition for two different modules.

The model presented above assumes the capacitance contribution due the execution units is relatively independent of allocation; however, as described in Chapter 3, this not always a good assumption since time-multiplexing of hardware can increase the capacitance switched.

Registers

Registers are treated the same way as the execution units. The existing estimation program gives information about the total number of register accesses (read/write) within a given sampling period. This is essentially the “activity” of the registers. For the purposes of calculating the register

energy, it is enough to know this “activity” and the actual number of physical registers is not required. The number of register accesses is multiplied with the average capacitance per register access to yield a register contribution given by:

$$C_{register} = N_{registers} \cdot C_{registers} \quad (\text{EQ 129})$$

While the total number of registers is not important in calculating the register switching capacitance, it will affect floorplanning and chip area and therefore the interconnect capacitance. Gated clocks are used and the clock capacitance is taken into account during the characterization of the registers. Each register has a control slice that locally buffers the incoming clock.

Interconnect

While estimating the power consumed by the execution units and registers is quite simple and accurate, estimating the interconnect component is a very difficult and challenging task. Driven by yield, floorplanning and synthesis considerations for throughput and area optimization, several elaborate prediction models for total chip and interconnect area have been built and successfully used [Kurdahi89, Heller77]. However, high-level synthesis adds additional requirements on the prediction tools next to accuracy; during the optimization process in high level synthesis, it is necessary to estimate the final cost frequently and therefore computationally intensive models are prohibited, regardless of their precision.

Estimating the interconnect component of the final implementation should not only take into account the effects of a wide spectrum of high level synthesis tools, such as assignment, allocation, scheduling and partitioning into macro blocks, but also the effects of many low-level CAD tools, such as placement, floorplanning and global and detailed routing. Of course, an accurate model for such a complex system can be built only when a particular set of design tools is targeted. As mentioned earlier, we targeted the HYPER high-level synthesis tools and the Lager IV silicon assembler [Shung91]. We used the scalable CMOS design rules provided by Mosis and targeted feature sizes of 1.2 μm and 2 μm .

The selection of this particular suit of design tools, enabled us to somewhat simplify the estimation process. We concentrated our attention only on the inter-block (between macro blocks, e.g. different datapaths) routing capacitance. The effect of intra-block (between logic modules inside a datapath) routing capacitance is already taken into account during the calculation of execution units and register contribution, by incorporating an average loading capacitance (determined for the datapath compiler used in the LagerIV silicon assembler).

Building a model which will take into account the effects of the various tools mentioned above is a formidable task. An extensive experimental study, followed by in-depth statistical analysis and verification is the only viable solution which will satisfy the contradictory requirements of modeling a complex system, with high accuracy in a computationally efficient manner.

The model for interconnect capacitance was built using fifty examples which were mapped from their Silage descriptions to layout using the HYPER synthesis system and the LagerIV silicon assembler. The selected examples cover a wide variety of DSP applications including linear and nonlinear filters (including FIR, direct form, cascade, parallel, continuous function, ladder, wave digital, Rao-Kailath IIR, polynomial and homomorphic filters), fast transformations algorithms (FFT and DCT), several video and image processing algorithms and audio examples. Selecting the set of examples for building the model was guided by the goal of including as diverse and as typical examples as possible. While the smallest example had only 12 operations, the largest one had more than 400 operations. One half of examples was pipelined to various extents, and a subset of the rest were transformed using several transformation in different orders. The examples cover a wide variety in the ratio of critical path to available time, amount of parallelism, types of used operations, level of multiplexing, size of the final implementations and other parameters.

After assembling the results for half the examples (for 25 examples), we started building the statistical model. It immediately became apparent that the best correlation is one between the total interconnect capacitance and the implementation area predicted by HYPER. It is widely

recognized that the quality of the prediction model is inversely proportional to number of parameters used during the prediction model building [Breiman84]. The number of parameters is equal to the sum of the number of input variables in the model, and amount of data needed to describe the model. We built the interconnect capacitance model using only one variable as the predictor (estimated area of the chip) and the complexity of the curve which fits data was minimized as much as possible. Although it appeared that both the line and quadratic polynomial, and in particular the third order polynomial fit the data well, none of these models passed the strict statistical test for goodness of fit and resubstitution validation procedure. However, a piecewise linear least square fit showed both excellent accuracy and robustness.

The three segment piece-wise linear fit used to model the interconnect is characterized by 4 points, which are (2.95, 16.5), (10, 77.6), (23.5, 217) and (80, 1257). The model is valid from a predicted area of 2 mm² (which is equal to the size of a chip with one execution unit, with a few registers and interconnects) to a predicted area of 90 mm² (which is equivalent to a chip can that can accommodate more than 30 execution units, with more than hundred registers and multiplexers).

The measurement on all 50 examples, showed that the average error of the piecewise linear model is less than 18% percent, and the maximum error is smaller than 33%. Robustness of the model is illustrated by the fact that during 10 random resubstitution of 25 different values for prediction, the largest average error was below 20%, and the maximum error did not exceed 40%.

During the development of prediction tools, often consistency (for two different randomly selected instances, the one with lower predicted value indeed has lower measured value) is more important than accuracy. Consistency of our single prediction variable (predicted area) was, despite the very simple and robust model, higher than 96%. Figure 5-12 shows both the measured (for all 50 examples) and predicted values of the interconnect capacitance as a function of the estimated chip area.

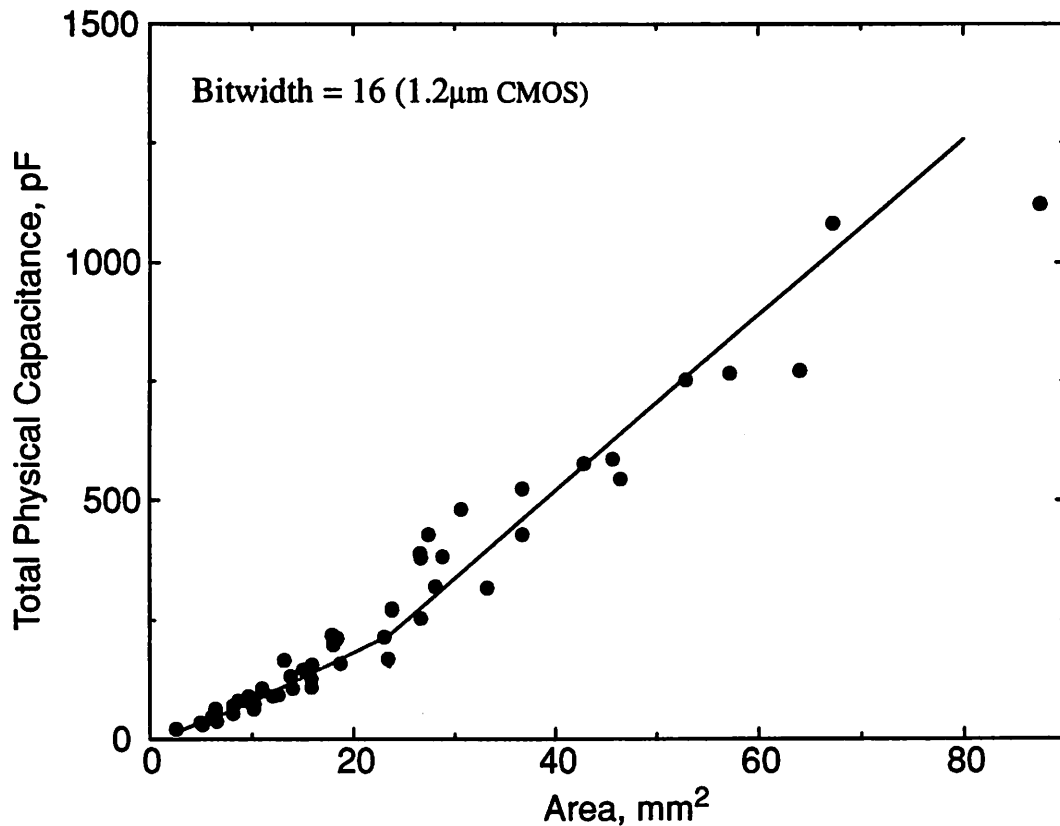


Figure 5-12 : Statistical estimation of interconnect capacitance.

Once the physical interconnect capacitance is accurately estimated, it is easy to establish a good interconnect power consumption model. The interconnect capacitance component is then given by:

$$C_{\text{interconnect}} = \alpha * C_{\text{total}} / N \quad (\text{EQ 130})$$

where α is the average activity (the total number of interconnect accesses multiplied by an average signal transition probability), C_{total} is the total estimated interconnect capacitance of the chip and N is an estimate of the number of physical interconnects (after bus-merging). The HYPER system provides accurate estimates of the number of interconnects and activity.

The interconnect model was built using the automatic place and route features of the LagerIV

placement and layout tool (Flint). In the future, a model should be developed based on hand-optimized floorplanning so the user can get feedback on the “goodness” of their placement.

Control Logic

The high level estimate for power consumed by control, like that for interconnect, is complicated since the control is not defined until after scheduling and hardware mapping. Neither the number of control blocks nor their function / size is known at the estimation stage, making it impossible to estimate any properties of the control theoretically. After scheduling, the control is defined and optimized by the *hardware mapper* and further during the *logic synthesis process* before mapping to layout. Like interconnect, therefore, the control needs to be estimated statistically.

The distributed control model used by HYPER is specially suited for low power. The control model incorporates a central finite state machine that generates the state information which is distributed to the local controllers. Control signals (e.g. LOAD for the register file) for the datapath are generated in the local controllers. Bus capacitance on the control lines is reduced by placing the local controllers close to the datapaths. Other than the lines carrying global state information, there are no global control lines.

Statistical models were built to estimate the power consumed by the global and local controllers. The models were generated using several DSP algorithms. Both initial and transformed versions were used to get a complete description of the sample space. Fully pipelined versions were not considered since the critical path and hence the number of states is one, eliminating the need for a controller. Each of the benchmark examples were mapped to SDL (structural description) using HYPER and then into layout using the LAGER IV silicon compiler. IRSIM was used to extract the switching capacitances required to build the statistical model. This process has been automated for characterizing new libraries or the effects of new tools.

Global Control Model:

The amount of capacitance switched in the global controller was found by simulating each benchmark circuit for a whole sample period after the initial conditions were set up. The capacitance switched per sample period in the global controller was found to be directly related to the number of states, N_{states} , and is given by:

$$C_{FSM} = \alpha_1 N_{states} + \alpha_2 \quad (\text{EQ 131})$$

For a $1.2\mu\text{m}$ technology, α_1 is 4.9fF and α_2 is 22.1fF . Figure 5-13 shows the total capacitance switched by the global controller as a function of the number of states. Note that the number of states not only determines the number of cycles the FSM goes through, but also the number of output bits it drives. Therefore, the total number of transitions and hence the capacitance switched, is strongly dependent on the number of states.

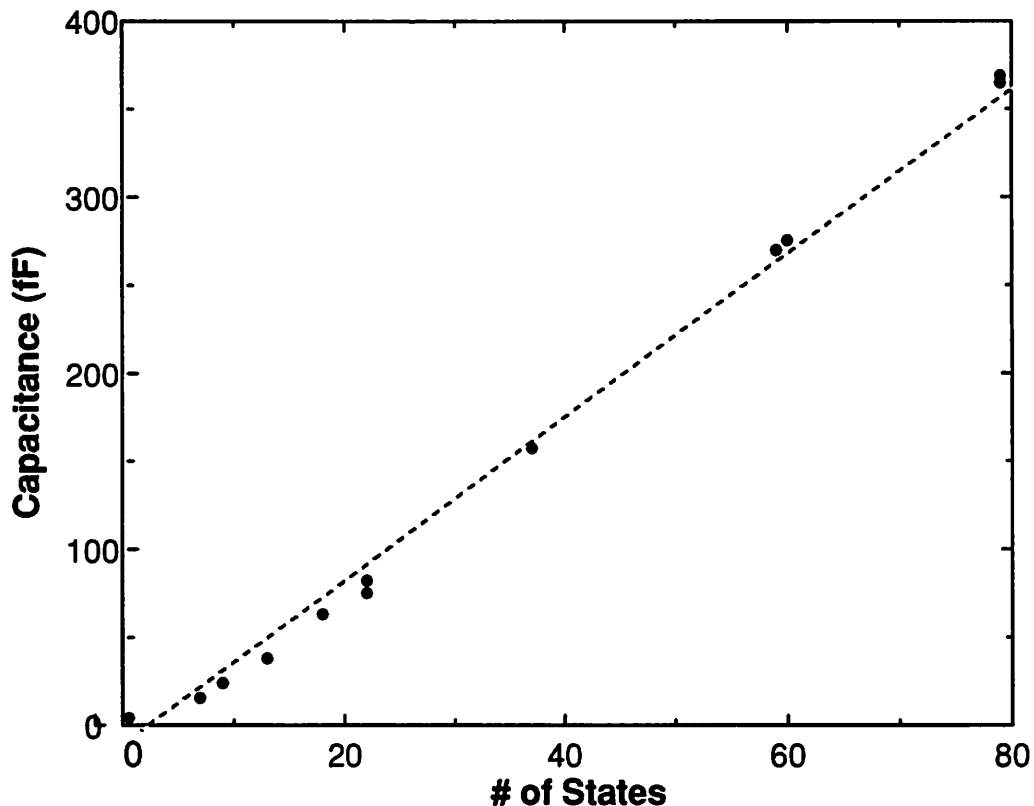


Figure 5-13 : Total capacitance switched vs. number of states for the global controller.

Local Control Model:

Since there are several local controllers in each design, the local controllers account for a larger percentage of the total capacitance than the global controller. A set of fifty examples were used to build the capacitance model for the local controller. Table 5-4 shows the data obtained for four different examples.

Table 5-4 : Results from various local controller units of the selected sample set.

Name (example)	Name (exu-unit)	# of outputs	Total Switched Capacitance (pF)	# of trans itions	length, width (in λ)	Area in μ^2
Wavelet (xlarge) 3 States 2 Inputs	add160	5	0.928	14	195, 94	6598.80
	add1610	5	0.943	14	219, 86	6780.24
	add1611	5	0.721	14	76, 46	1258.56
	shr160	11	1.787	34	227, 166	13565.52
	shr1610	9	1.509	30	211, 174	13217.04
	shr1611	5	0.943	14	211, 86	6532.56
	sub160	5	0.691	14	203, 94	6869.52
	sub1610	7	1.129	22	203, 118	8623.44
	sub1611	7	0.980	18	219, 118	9303.12
	transfer160	5	0.845	18	203, 78	5700.24
	transfer161	5	0.930	14	203, 126	9208.08
	transfer162	5	0.808	18	195, 70	4914.00
IIR 22 States 5 Inputs	adder120	53	82.738	544	451, 1504	244189.44
	adder121	47	76.981	528	427, 1232	189383.04
	shr120	30	42.553	296	403, 1048	152043.84
	shr121	25	55.556	302	363, 976	127543.68
	shr122	24	43.253	312	403, 1096	159007.68
	sub120	59	49.855	364	411, 1232	182286.72
	sub121	45	55.556	380	427, 1088	167247.36
	transfer	19	14.706	64	307, 504	55702.08
Discrete Cosine trans- form; 36 states; 6inputs	adder80GL	51	128.8085	644	539, 1726	334913.04
	shr80GL	35	101.891	428	475, 1566	267786.00
	subtractor80GL	53	145.349	704	547, 1806	355637.52

Table 5-4 : Results from various local controller units of the selected sample set.

Name (example)	Name (exu-unit)	# of outputs	Total Switched Capacitance (pF)	# of trans itions	length, width (in λ)	Area in μ^2
Avenhaus- Ladder Filter 79 States 7inputs	adder140GL	7	27.68	174	251,344	31083.84
	adder1410GL	7	24.916	170	243,256	22394.88
	adder1411GL	7	13.410	174	235,232	19627.20
	adder1412GL	7	27.635	174	251,280	25300.80
	shr140GL	12	97.841	420	395,1128	160401.60
	shr141GL	13	119.757	560	355,1008	128822.40
	shr142GL	12	102.044	426	403,1000	145080.00
	shr143GL	12	143.883	542	395,1120	159264.00
	subtractor140GL	25	97.104	430	87,1000	139320.00
	subtractor141GL	17	78.879	406	371,952	127149.12
	subtractor142GL	17	81.948	426	355,896	114508.80
	subtractor143GL	11	71.729	374	339,744	90797.76

The capacitance switched in the local controllers were determined by applying the state input sequence to each of the local controllers and simulating using IRSIM over a whole sample period. Correlations between the capacitance switched and several parameters like the number of states in the control, the number of outputs of the controller, and the total number of transitions on the output control signals were measured. The capacitance of the control was found to be highly correlated to the number of transitions on the output control signals. Note that, in this case, unlike the global controllers, the number of states gives no information about the number of transitions on the output nodes which depends on the glue-logic to be implemented. The transitions must therefore be separately accounted for. Using statistical tools to fit the data to a polynomial function, it was found that the total capacitance switched, in one sample period, for any local controller is a linear function of the number of transitions, the number of states and the bus factor and is given by:

$$C_{lc} = \beta_0 + \beta_1 N_{trans} + \beta_2 N_{states} + \beta_3 B_f \quad (\text{EQ 132})$$

where N_{trans} is the number of transitions, N_{states} is the number of states, B_f is the bus factor

(explained below), and C_{lc} is the capacitance switched in any local controller in one sample period. B_f represents the activity factor on busses and is defined as the ratio of the number of bus accesses to the number of busses. It is a measure of the average number of times busses are accessed. It represents a measure on the number of multiplexors and each multiplexor requires control signals from the controller. For a 1.2 μm technology, β_0 , β_1 , β_2 and β_3 are 72, 0.15, 8.3 and 0.55 respectively. Figure 5-14 shows the correlation between the actual and predicted capacitance switched per sample period for several different examples.

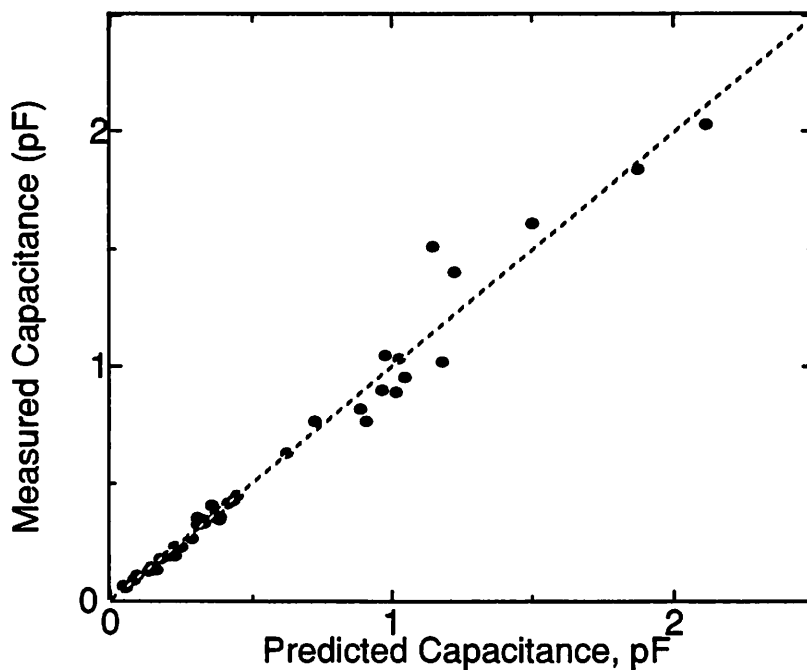


Figure 5-14 : Capacitance switched vs. predicted capacitance for local controllers.

This model, though accurate to within 20% (as indicated by resubstitution validation procedure) is not sufficient since it assumes that the number of transitions is known. The number of transitions depends on assignment, scheduling, optimizations performed by the hardware mapper and the logic optimization tool(misII), the standard cell library used, the amount of glitching, and the statistics of the inputs. It is impossible to determine the combined effects of all these elements apriori. A statistical model relating the number of transitions to high level parameters was therefore derived. The important high level parameters that affected the number of transitions are:

-
- Size/complexity of the graph, i.e. the number of nodes and the number of edges. This is because the number of reads from and write to registers is dependant on the number of edges and nodes in the graph respectively. Control signals need to be generated for every read and write process.
 - The number of execution units times the number of states. This is because each execution unit receives the clock and clock inverse every control cycle. In the hardware model used, the clock inverse is generated in the local controllers and fed to the execution units.

The HYPER high level synthesis system provides a lower bound on the number of execution units within an accuracy of 10% [Rabaey91b]. The lower bound on the busses tracks the actual number of busses closely and the maximum number of busses tracks the total number of bus accesses. The numbers predicted by HYPER were therefore used in building the correlation model. The fit function obtained for the total number of output transitions on local controllers based on the three high level parameters mentioned above is given by:

$$N_{trans} = \gamma_1 + \gamma_2(N_{nodes} + N_{edges}) + \gamma_3(S \times N_{Exu}) \quad (\text{EQ 133})$$

where N_{trans} is the number of transitions on the outputs of the local controllers, S is the number of control cycles per sample period, N_{edges} and N_{nodes} are the number of edges and nodes respectively in the CDFG and N_{exu} is an estimate for the total number of execution units. For a 1.2 μm technology γ_1 , γ_2 and γ_3 are 178.7, 7.2 and 2.0 respectively. Figure 5-15 shows the total number of transitions in one sample period for the different examples vs. the fit function for the transitions based on high level parameters.

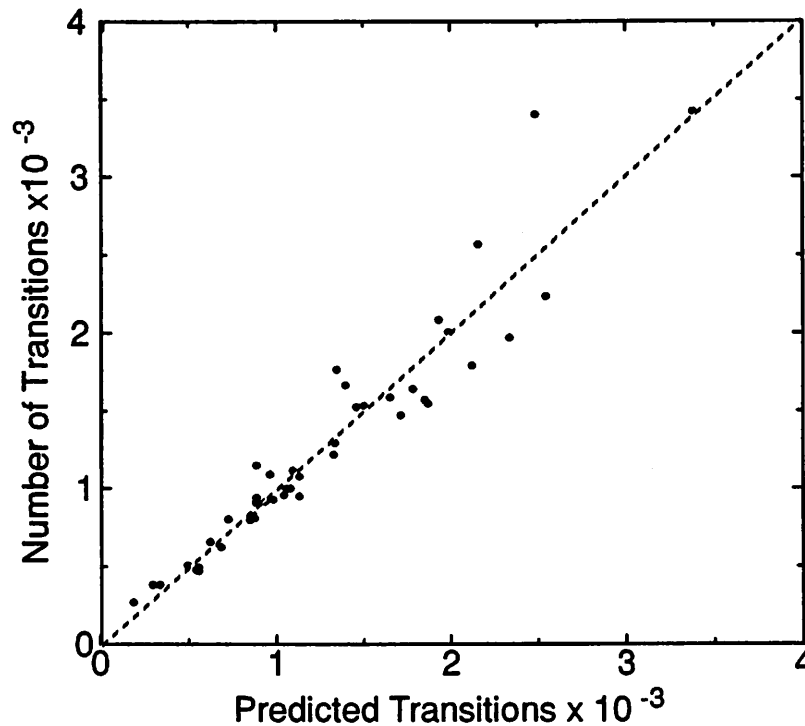


Figure 5-15 : Actual number of transitions vs. predicted transitions for local controllers.

This statistical estimate takes into account the effects of logic synthesis and optimization. Effect of undetermined factors such as glitching are also included. Each datapath element in the cell library has a built in control-slice to locally buffer the incoming control signals. The internal gate and routing capacitance is taken into account when characterizing the leafcells. For example, the multiplexor select signal for a 16-bit datapath is buffered in the control slice. Therefore, the controller only drives a single (typically minimum sized) gate. This information was used to determine the loading capacitance during simulations.

5.3.2 Supply Voltage Estimation

The power supply voltage at which the flowgraph implementation will meet the timing constraints is estimated. The initial flowgraph which meets the timing constraints is typically assumed to be

operating at a supply voltage of 5V with a critical path of T_{initial} (the initial voltage will be lower if $T_{\text{initial}} < T_{\text{sampling}}$). After each move, the critical path is re-estimated, and the new supply voltage at which the transformed flowgraph still meets the time constraint, T_{sampling} , is determined. For example, if the initial solution requires 10 control steps (and let's assume that this is the same as the sampling period) running at a supply voltage of 5V, then a transformed solution that requires only 5 control steps can run at a supply voltage of 2.9V (where the delay increases by a factor of 2, Figure 2-26) while meeting the same constraints as the initial graph.

A model for delay as a function of V_{dd} is derived from the curve shown in Figure 5-16. Let the speedup of a transformed solution be defined as:

$$\text{Speedup} = T_{\text{sampling}} / T_{\text{criticalpath}} \quad (\text{EQ 134})$$

where $T_{\text{criticalpath}}$ is the critical path of the transformed solution and T_{sampling} is the throughput constraint.

Since, the major power reduction during CDFG optimization using transformations is attributed to a reduction of the supply voltage and since the supply voltage has to be constantly evaluated (every time the objective function is called), it is important to model delay- V_{dd} relationship using an accurate and computationally efficient procedure. This relationship (of delay- V_{dd}) was modeled using Neville's algorithm for rational function interpolation and extrapolation [Press88]. Neville's algorithm provides an indirect way for constructing a polynomial of degree $N-1$ so that all of the used points are exactly matched.

Figure 5-16 also shows the accuracy of the interpolated data (plotted for speedup values ranging from 0.86 to 26 with increments of 0.1) when compared to the experimental data. Figure 5-17 shows an overview of the power estimation routine. It takes a control dataflow graph as input and computes the power using the existing estimation routines (critical path, bounds on resources and activity) along with the newly developed capacitance and voltage estimation routines (as discussed in this section).

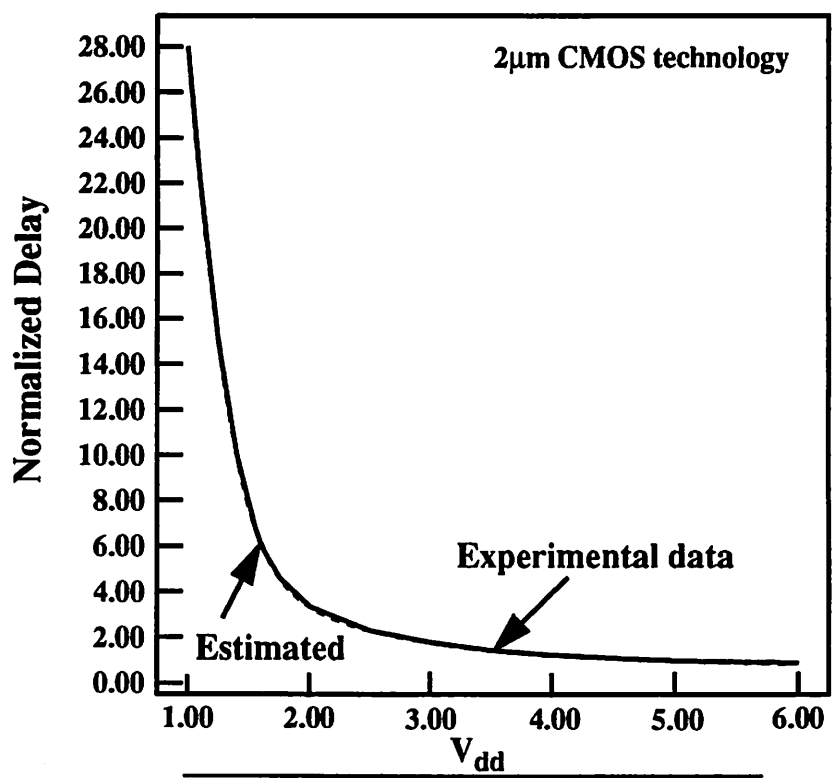


Figure 5-16 : Accuracy of V_{dd} estimation.

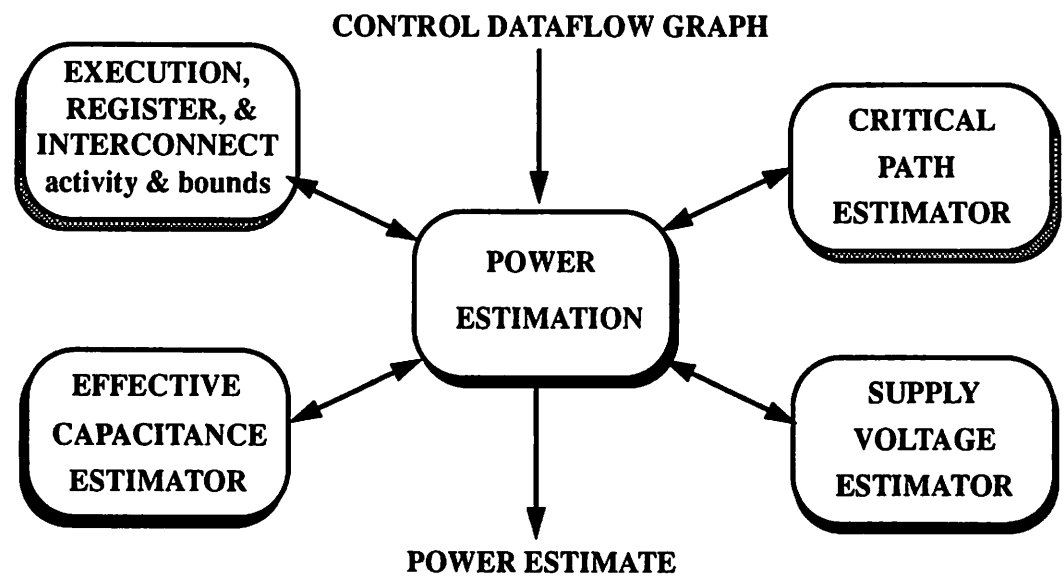


Figure 5-17 : Overview of power estimation.

5.4 Optimization Algorithm

The algorithm development for power minimization using transformations was greatly driven by several considerations and constraints. The basic strategy involved:

- Efficient tailoring of existing library transformations for power optimization.
- Fully utilizing the information about the design solution space obtained through extensive experimentation and theoretical insights (so that the algorithm can quickly come out of deep local minima).
- Efficient use of computer resources (CPU time and memory) so that large real-life examples can be addressed.
- Developing the modular software so that future enhancements or changes can easily be incorporated.

The computational complexity analysis of the power minimization problem showed that even highly simplified versions of the power optimization tasks using transformations are NP-complete. For example, we have shown that retiming for power minimization is a NP-complete problem. Computational complexity of retiming for power optimization is particular interesting and somewhat surprising, since retiming for critical path reduction has several algorithms of polynomial-time complexity [Leiserson91].

The computational complexity of the power minimization problem implies that it is very unlikely, even when the set of applied transformations is restricted, that polynomial time optimal algorithm can be designed. Two widely used alternatives for design of high quality suboptimal optimization algorithms are probabilistic and heuristic algorithms. Both heuristic and probabilistic algorithms have several distinctive advantages over each other. While the most important advantage of heuristic algorithms is a shorter run time, probabilistic algorithms are more robust and have stronger mechanisms for escaping local minima's.

An extremely wide spectrum of probabilistic algorithms are commonly used in various CAD and

optimization problems. Among them, the most popular and successful are simulated annealing, simulated evolution, genetic algorithms, and various neural network algorithms (e.g. Boltzmann machine). Although recently there has been a considerable effort in analyzing these algorithms and the type of problems they are best suited for (for example a deep relationship between simulated annealing and solution space with fractal topology have been verified both experimentally and theoretically [Sorkin91]), algorithm selection for the task at hand is still mainly an experimental and intuitive art.

In order to satisfy all major considerations for the power minimization problem, we decided to use a combination of heuristic and probabilistic algorithms so that we can leverage on the advantages of both approaches. Before we get into the algorithmic details, we will briefly outline the set of transformations used.

The transformation mechanism is based on two types of moves, global and local. While global moves optimize the whole DCFG simultaneously, local moves involve applying a transformation only on one or very few nodes in the DCFG. The most important advantage of global moves is, of course, a higher optimization effect; the advantages of local moves is their simplicity and small computational cost. We used the following global transformations (i) retiming and pipelining for critical path reduction (ii) associativity (iii) constant elimination and (iv) loop unrolling. In the library of local moves we have implemented three algebraic transformations: associativity (generalized to include properties of inverse elements as introduced in [Potkonjak91]), and commutativity as well as local retiming. All global moves minimize the critical path using polynomial optimal algorithms. Their use is motivated by the need for shorter run-times. For example, the global pipelining move for a 7th order IIR filter with 33 nodes took 1.8 CPU seconds on a SPARC2 workstation. Extensive experimentation showed that although the best solutions in semi-exhaustive searches is rarely one with the minimum critical path, it is very often topologically very close (a small number of moves is needed to reach it starting from a solution with the shortest critical path).

The Leiserson-Saxe algorithm for critical path minimization using retiming is used. The algorithm is modified in such a way that it automatically introduces the optimal number of pipeline stages needed to minimize the cost function for power. Pipelining is accomplished by allowing the simultaneous addition of the equal number of delay elements on all inputs or all outputs but not both. The Leiserson-Saxe algorithm is strictly based on minimizing the critical path and inherently does not optimize the capacitance being switched for a given level of pipelining. The modified algorithm determines the level of pipelining where the power is minimized and this is used as a “good” starting point for optimization using local moves (as will be discussed below). The global move for associativity involves the minimization of the critical path using the dynamic programming algorithm. Loop unrolling does not involve any optimization, instead it enables transformations which reveal large amount of concurrency for other transformations. For each local move, we defined the inverse local move which undoes the effect of the initial move.

As previously mentioned, the algorithm for power minimization using transformations has both heuristic and probabilistic components. While the heuristic part uses global transformations, the probabilistic component uses local moves. The heuristic part applies global transformations one at the time in order to provide good starting points for the application of the probabilistic algorithm. The probabilistic algorithm conducts a search in a broad vicinity of the solution provided by the heuristic part with the goal of minimizing the effective capacitance (trading-off control, interconnect and activity). The underlying search mechanism of the probabilistic part is simulated annealing with the goal that local minima’s can be escaped by using uphill moves and the effects of various transformations can be efficiently combined so as to minimize the capacitance component of power.

Initially constant propagation is applied to reduce both the number of operations and the critical path, followed by the modified Leiserson-Saxe algorithm and the dynamic programming algorithm for the minimization of the critical path using associativity. This step is followed by simulated annealing which tries to improve the initial solution by applying local moves

probabilistically. For simulated annealing we used the most popular set of parameters [Algoritmica91]. For example, we used a geometric cooling schedule with a stopping criteria which terminates the probabilistic search when no improvement was observed on three consecutive temperatures. After each move, a list of all possible moves is generated for each local transformation. The transformation and particular move to be applied is then selected randomly. In addition to the above algorithm, loop-unrolling can be used as preprocessing step.

HYPER-LP also allows optimization using a user specified sub-set of transformations (for example, optimizing with only pipelining). Figure 5-18 shows an overview of the HYPER-LP system.

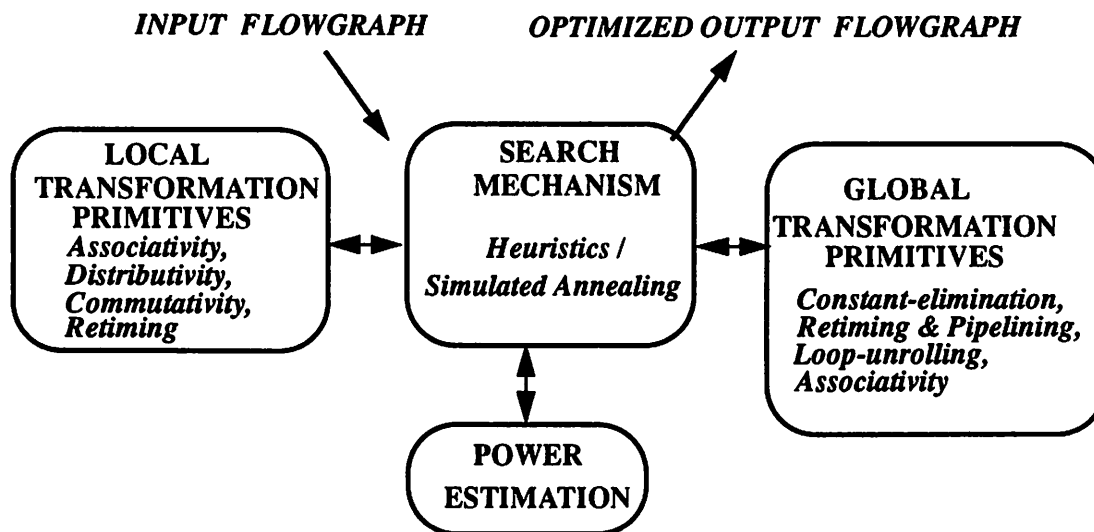


Figure 5-18 : Overview of the HYPER-LP system.

5.5 Examples and Results

The techniques described in the previous sections will now be applied to “real-world” examples to demonstrate that a significant improvement in power can be achieved. Three examples of distinctly different computational structures are presented in this section and are optimized for power using the HYPER-LP system. Multiplications with constants were converted to shift-add

operations. The improvement from this transformation was not taken into account since this is a standard transformation performed for area or throughput optimization.

5.5.1 Example #1: Wavelet Filter

The first example is a wavelet filter, used in speech and video applications. The implementation contains high-pass and low-pass filters that are realized as 14th order FIR modules using a wordlength of 16-bits (determined by high-level simulation). This example is representative of a wide class of important signal processing applications such as FFT, DCT (Discrete Cosine Transform), matrix multiplication, cyclic convolution and correlation that have no feedback loops in the signal flowgraph. For this class of applications, retiming and pipelining provides an efficient and straight forward way to reduce critical paths (and thus voltage) while preserving throughput and the number of operations.

The initial reference design is a fully time-multiplexed area-optimized realization of the filter, with minimal amount of resources (1 adder, 1 subtractor, 1 shifter, and 4 global busses). The number of global busses were minimized using global bus-merging (i.e busses that are never accessed simultaneously can be collapsed) with the goal of minimizing the area. This solution has a critical path of 22 clock cycles and can be clocked at a maximum frequency of 1.5Mhz (since the critical path for this 16-bit datapath is around 30ns). Using HYPER-LP, it was found that Retiming by itself was sufficient to reduce the critical path of this design to 3 control cycles. Since the throughput requirement is 1.5Mhz, the clock period can be made approximately 7 times as long ($= 22 / 3$) while meeting the timing constraint and therefore the supply voltage can be reduced to 1.5V.

Since the amount of resources increases significantly (since the allocation is done with available time = critical path = 3 cycles), we expect the chip area and hence the interconnect component of power to increase. This is seen from Figure 5-19, which shows a plot of the average capacitance per bit obtained from layout extraction as a function of the bus number for the initial 22 cycle

implementation (which contained 4 busses) and for the final 3 cycle implementation (which contained 25 busses). The figure also shows the capacitance obtained from the estimation of interconnect using the model presented in section 5.3.1.

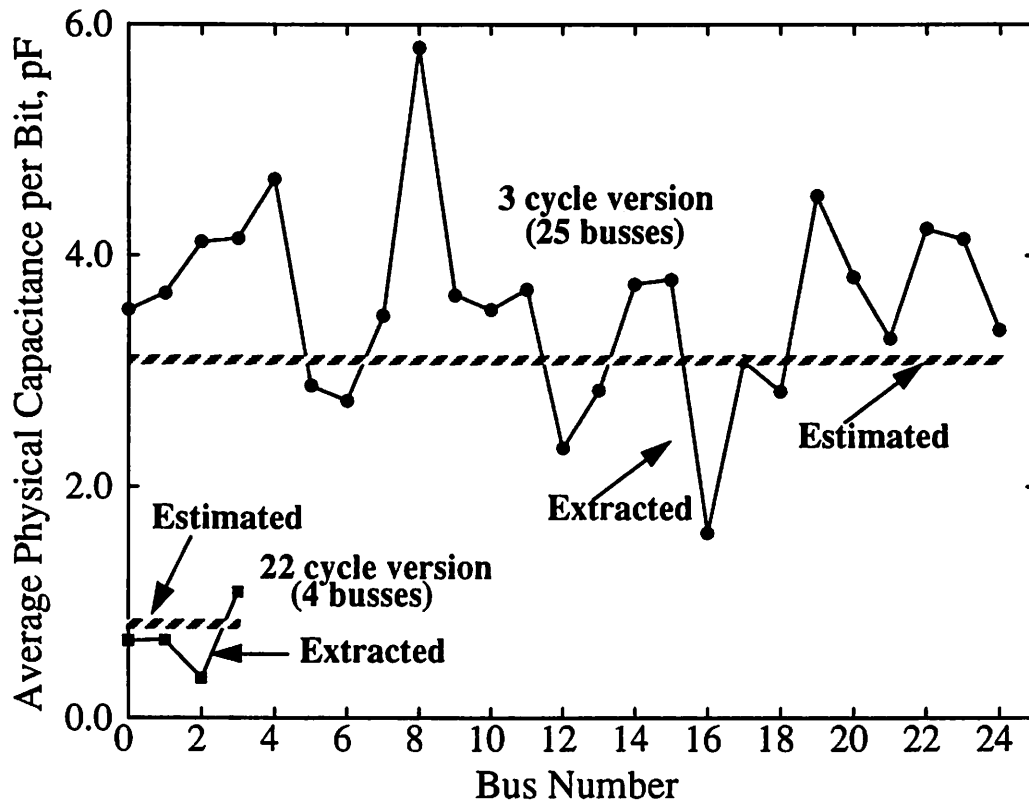


Figure 5-19 : Bus capacitance for the wavelet filter (estimated and extracted).

The global bus capacitance switched increased by a factor of 3 from 400pF to 1200pF. However, the total capacitance switched was actually *smaller* than the minimal area version! There are several factors contributing to this decrease in capacitance:

- The amount of multiplexor, tri-state and control switches (which contributed to 45% of the total power of the minimal area design) were reduced significantly.
- The initial design uses bus-merging to significantly reduce the interconnect area. However, this does not come for free as the loading for the tri-state buffers driving the bus increases. Also all of the multiplexors whose inputs are connected to the shared bus will switch every time the bus

value transitions, resulting in extra power. The problem is not as severe in the final design since there is little room for bus-sharing since there are fewer control cycles.

- The average transition activity (which is not modeled by the estimator presented in Section 6) and hence the capacitance per access for the logic modules (adder, subtractor, and shifter) was significantly lower in the final version. The reason is that in the initial version, since the modules were fully time-multiplexed, the inputs to the modules were coming from different sources (and hence uncorrelated) and had a higher probability of transitioning. In the final version (in which many units were allocated), there are longer periods of inactivity during which the inputs to the modules do not change and hence the units do not switch, resulting in lower activity.

The total power hence reduced by a factor of 18 (these results are obtained from the IRSIM switch level simulator using a linear RC model using real input data). Table 5-5 shows the statistics of the initial and final designs. Note that area has been traded for lower power.

Table 5-5 : Results for the Wavelet Chip (1.2 μ m CMOS technology)

Version	f_{sampling}	# of Control Cycles / Clock Period	Supply Voltage	Total Average Capacitance	Power, mW	Area, mm ²
Initial	1.5Mhz	22 / 30ns	5V	2870pF	107mW	8.5 mm ²
Optimized for Power	1.5Mhz	3 / 220ns	1.5V	1735pF	5.8mW	62.9mm ²

5.5.2 Example #2: IIR Filter

The second example is a 16-bit 7th order IIR filter with a 4th order equalizer designed using the filter design program Filsyn [Comsat82]. This example is representative of the largest class of examples where feedback is an inherent but not particularly limiting part of the computation structure.

Unlike the previous example, when the transformation set of the HYPER-LP system was limited only to retiming, no reduction in the power consumption was observed. However, when the

transformation set was enhanced to include both retiming and pipelining, the tool was very effective and achieved a reduction in the number of control steps used from 65 to 6 cycles. The implementation area increased as the number of control cycles decreased. Figure 5-20a shows a plot of the number of execution units vs. the number of control cycles. The number of registers increased from an initial number of 48 (for 65 control cycles) to 102 (for 6 control cycles). This is a typical example where the area is traded for lower power. It is immediately evident that the price paid for reducing critical path is the increased registers and routing overhead. Figure 5-20b shows a plot of the normalized clock cycle period (for a fixed throughput) as a function of number of control cycles. We see that reducing the number of control cycles through transformations allows for an increase in clock cycle period or equivalently a reduction in supply voltage. However, at very low voltages, the overhead due to routing increases at a very fast rate and the power starts to increase with further reduction in supply voltage. The power reduced approximately by a factor of 8 as a result of dropping the voltage from 5V to approximately 1.5V. Figure 5-21 shows a plot of Power vs. V_{dd} for a fixed throughput. Note that the low-power solution is somewhere between the minimal area solution and the maximum throughput solution.

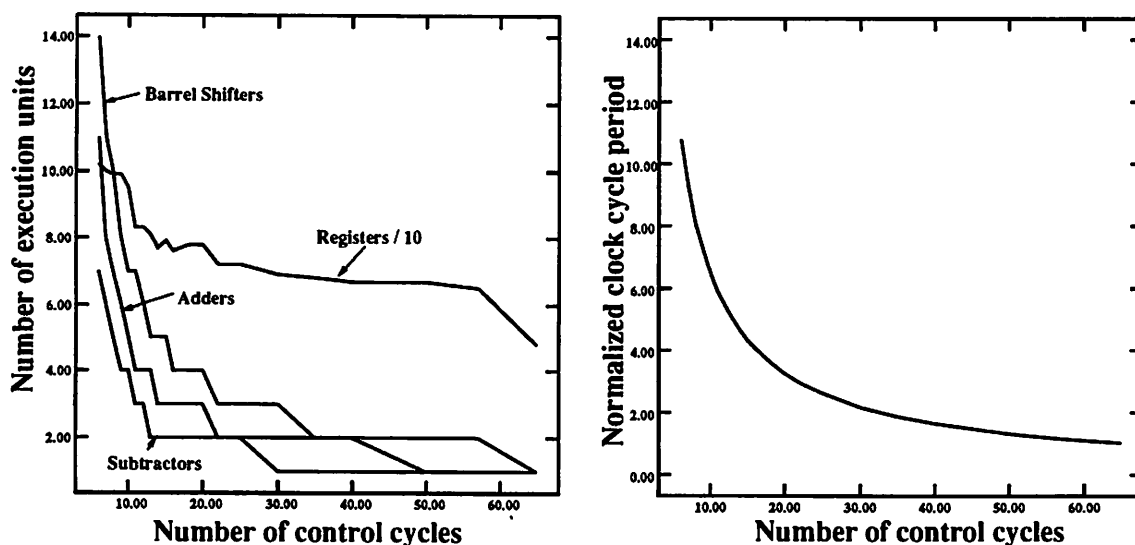


Figure 5-20 : Plot of # of units (a) and clock cycle period (b) vs. # of control cycles for the IIR example

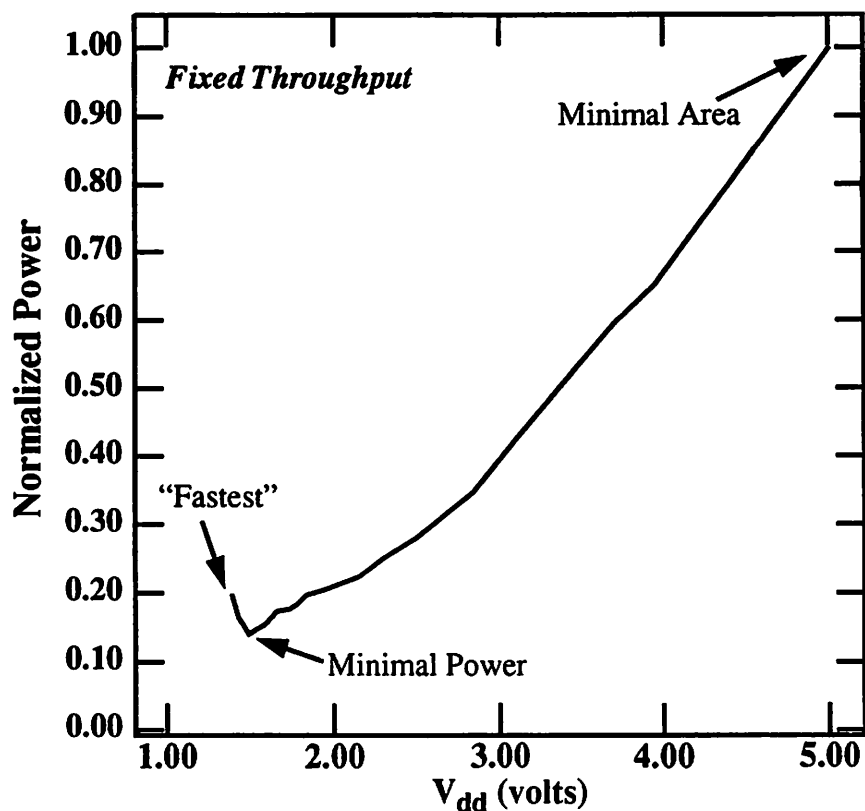


Figure 5-21 : Power vs. V_{dd} for the IIR example.

5.5.3 Example #3: Volterra Filter

The third example is a second order Volterra filter. This is a particularly challenging example since pipelining cannot be applied due to a recursive bottleneck imposed by long feedback loops (optimizing with pipelining as the only transformation results in power reduction only by a factor of 1.5). For this case, loop unrolling, retiming and associativity transformations proved to be important in alleviating the recursive bottleneck, allowing for a reduction in critical path and voltage.

Figure 5-22 shows the final layouts of optimizing the Volterra filter for power as well as for area while keeping the throughput fixed. This final implementation has approximately nine times better

power characteristic and the same area as the initial implementation.

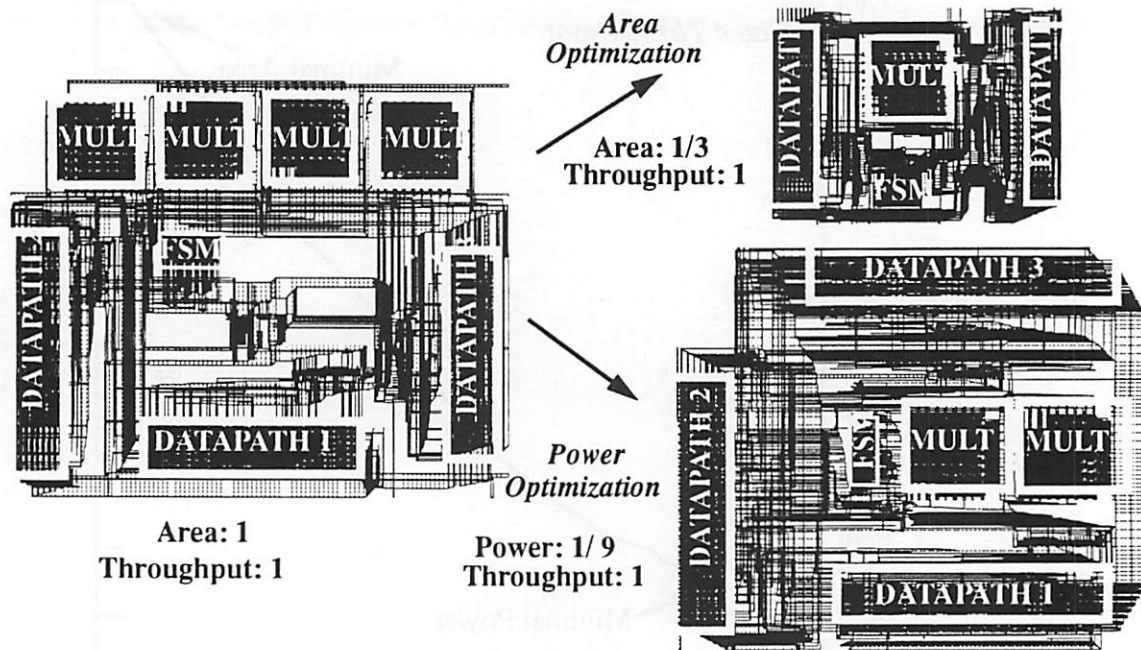


Figure 5-22 : Optimizing the volterra filter.

Table 5-6 shows a summary of power improvement after applying transformations relative to initial solutions that met the required throughput constraint at 5V for the representative examples described in this section. The results indicate that a large reduction in power consumption is possible (at the expense of area) compared to present-day methodologies. Also interesting was the fact that the optimal final supply voltage for all the examples was much lower than existing standards and was around 1.5V.

Table 5-6 : Summary of results.

Example	Power Reduction	Area Increase
Wavelet (FIR) Filter	18	7.4
IIR Filter	8	6.4
Volterra Filter	9	1

5.6 Summary

The focus of this chapter is on automatically finding computational structures that result in the lowest power consumption for DSP applications that have a specific throughput constraint given a high-level algorithmic specification. The synthesis approach consisted of applying transformation primitives in a well defined manner in conjunction with efficient high-level estimation of power consumption. The basic approach is to scan the design space utilizing various algorithmic flowgraph transformations, high-level power estimation, and efficient heuristic/probabilistic search mechanisms. The estimation of power consumption involved estimating the total capacitance switched and the operating voltage. The estimation of the total capacitance switched includes four components: execution units, registers, interconnect and control. Both analytical and statistical approaches were used in the capacitance modeling. The synthesis system was used to optimize power for several design examples and the results indicate that more than an order of magnitude reduction in power is possible over current-day design methodologies (that tend to minimize area for a given throughput) while maintaining the system throughput. This chapter has addressed some key problems in the automated design of low-power systems, and provides a good starting point for addressing other research problems like detailed power estimation, module selection, partitioning, and scheduling for power optimization.

CHAPTER 6

A Low-power Chipset for a Portable Multimedia Terminal

The near future will bring the fusion of three rapidly emerging technologies: personal communications, portable computing, and high bandwidth communications. Over the past several years, the number of personal communications services and technologies has grown explosively. For example, voiceband communications systems such as mobile analog cellular telephony, radio pagers, and cordless telephones have become commonplace, despite their limited nature and sometimes poor quality of transmission. In portable computing, “notebook” computers more powerful than the desktop systems of a few years ago are commonplace, while even more capable RISC-based portable workstations are quickly emerging. Much of the recent efforts have gone into the design and building of the “information superhighway,” the fiber-optic network with a bandwidth far exceeding 10 gigabits/second. However, in spite of their individual popularity, there has been little integration of these three diverse services. Our goal is to exploit these new technologies to provide ubiquitous access to data, computing, and communications through a specialized, wireless multimedia terminal, dubbed the “InfoPad” [Brodersen93].

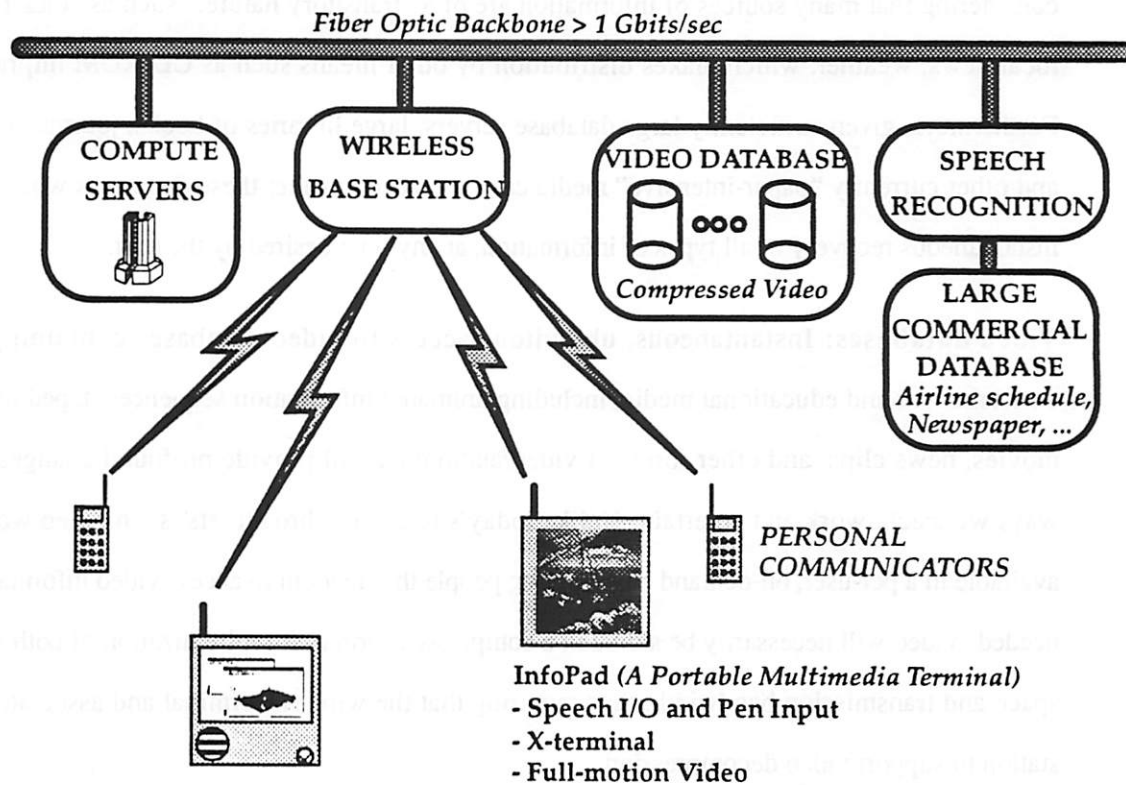


Figure 6-1 : Overview of a future personal communications system.

In Figure 6-1, a schematic view of the InfoPad system is shown. In addition to providing portable voice communication to the user, InfoPad will provide portable access to the wide variety of information and computation services that will be offered on the future wide band network. By coupling to this high-bandwidth network through a low-latency interface, InfoPad gives the user equally good service no matter where the person and the service provider is located in a building. Some key features that will evolve for this “wireless computing” environment are as follows:

Commercial databases: Servers will provide access to large text and graphics commercial databases containing various information, such as international and domestic news, financial information, traffic data, transportation schedules, voice mail, telephone numbers, news, bulletin boards, and educational information. The need for instantaneous connectivity is apparent when

considering that many sources of information are of a "transitory nature," such as stock pricing, local news, weather, which makes distribution by other means such as CD-ROM impractical. Furthermore, given sufficiently large database servers, large libraries of books, journal archives, and other currently "paper-intensive" media can be placed on-line; these databases would allow instantaneous recovery of all types of information, at any time desired by the user.

Video databases: Instantaneous, ubiquitous access to video databases containing both entertainment and educational media, including animated information sequences, taped lectures, movies, news clips, and other forms of video/audio data will provide profound changes in the ways we teach, work and entertain. Unlike today's television broadcasts, such video would be available in a per-user, on-demand basis, giving people the freedom to access video information as needed. Video will necessarily be stored in a compressed format, for minimization of both storage space and transmission bandwidth, thus requiring that the wireless terminal and associated base station to support video decompression.

Advanced User Interface Technology: Simplified entry mechanisms such as voice-recognition and handwriting-recognition offer novel access to the above functions. The design of an effective user interface to access such a vast information storehouse is a critical issue. By using speech recognition and pen-based input, supported by a large, speaker-independent recognition servers placed on the network, such interfacing and information access can be tremendously simplified. Placing the recognition units on the network conserves power in the portable, and enables much larger and much more complex recognition algorithms to be employed. Such recognizer servers can also make use of context-sensitive analysis, which can increase recognition accuracy by determining which words are most likely to be used in a given application.

Compute Servers: Support for a distributed computing environment based on network attached compute servers, such as the MIT X-Window system. Such client-server computing environments have made it clear that computation need not be done on the local machine (the display server) that

a user is operating; instead, the computation is done by programs executing on many remote machines (clients), that simply issue graphics commands to the server to display their results. Many inexpensive "X-terminals" already exist, with no computation capability within them save that of behaving as an intelligent display server. Unlike TTY terminals that can only communicate with a single host machine, such X-terminals possess all of the necessary networking capability to communicate with as many remote servers as needed. It is upon this model that the Infopad multimedia terminal will be based; remote computation servers will be used to run applications such as spreadsheets, word processors, and so forth, with the results being transmitted to the terminal. Likewise, through the InfoPad, the user will have interactive access to all the computers in the building to perform intensive tasks requiring simulations and computer-aided design.

Figure 6-2 summarizes the functionality that must be provided by the portable terminal to support the access of various multimedia information servers. This includes the interface to a high speed

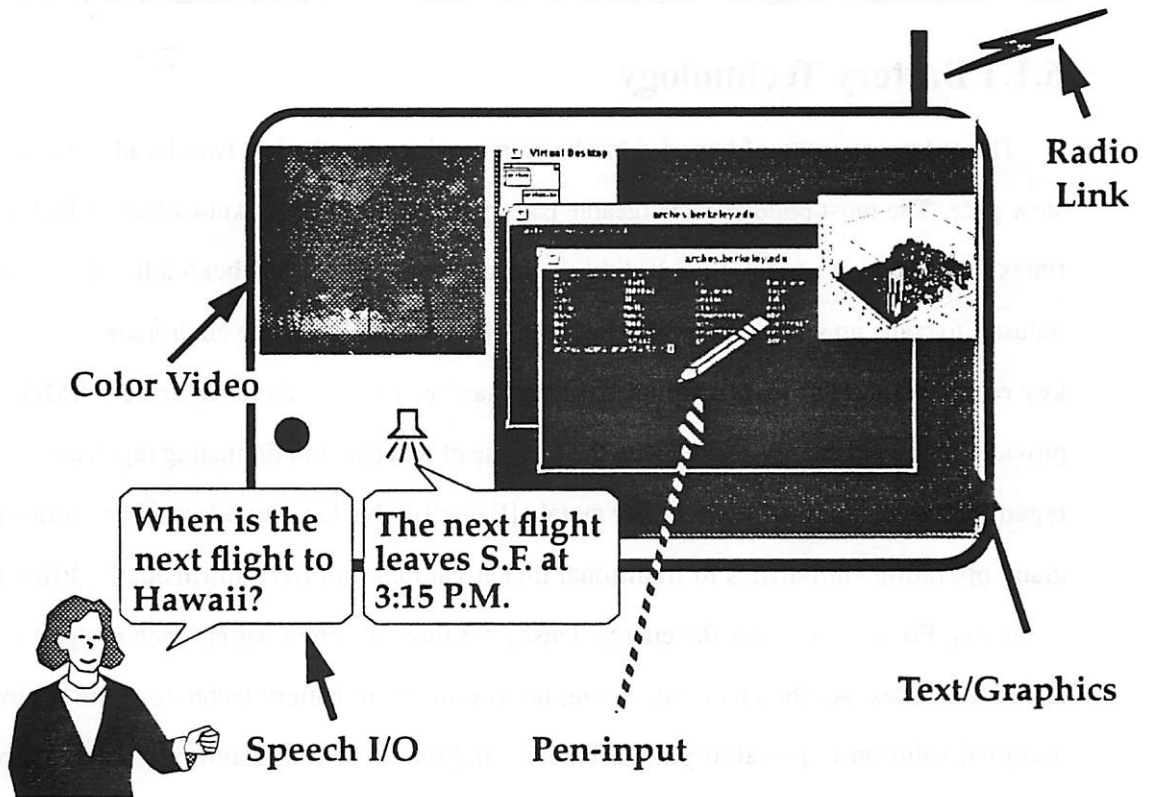


Figure 6-2 : Desired features of a Portable Multimedia Terminal.

wireless link, text/graphics output, simplified user interfaces like pen input and speech I/O, and finally support for one-way full-motion video. A keyboard input can be included as an option to pen and speech input.

The key design consideration for such a portable multimedia terminal is the minimization of power consumption. The portable multimedia form factor must be slim and light weight, while being durable and tolerant of the users environment. The focus of this chapter and the next chapter is on the low-power approaches used in the design of a chipset that performs the various I/O functions required by the InfoPad terminal. Minimizing power consumption of the terminal electronics requires a design methodology that supersedes the technology and circuit levels and addresses architecture, algorithm and system partitioning as well.

6.1 Technology Trends for Low Power

6.1.1 Battery Technology

The energy capacity of batteries has been improving over the last two decades, but at a very slow pace. The most popular rechargeable battery in use has been nickel-cadium, which has two times more capacity today than it did two decades ago. There has been a lot of efforts from industry to come up with rechargeable high capacity batteries which are environmentally safe. The key competition for nickel-cadium batteries has been nickel-metal hydride (NiMH) which provides improved energy density with the promise of reducing or eliminating regulatory concerns regarding the disposal of toxic heavy metals [Eager91]. Nickel-metal hydride batteries possess many operating similarities to traditional nickel-cadium, but have intrinsically different cell chemistry. Figure 6-3 shows the energy density for three different battery technologies over the last two decades. As shown on this figure, improvements in battery technology only provides a marginal solution to portability requirements, and therefore we cannot rely solely on battery improvements.

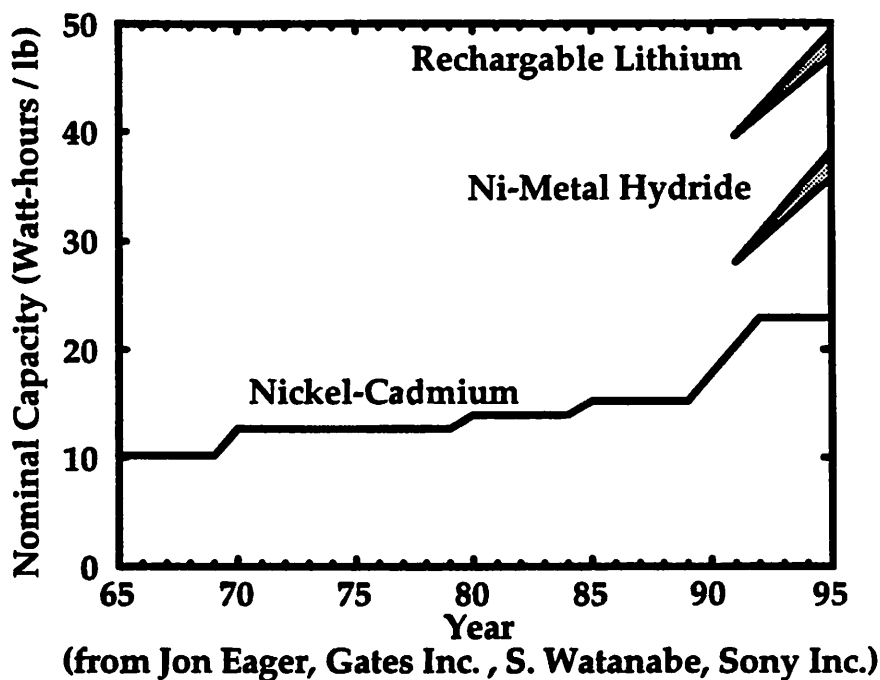


Figure 6-3 : Trends in battery technology for the last 20 years.

6.1.2 Display Technology

Improvements in display technology will be a key element to the implementation of future low-power portable systems. Currently, two display technologies are used for laptop computers: passive displays and active-matrix displays. Passive displays consume significantly lower power (2-3W) while having a large response time (150mS). For applications that require a fast response time, like full-motion video, active matrix displays are required. For color active matrix displays, majority of the power goes into the backlight - required due to the poor transmission characteristics of the color filters - which requires more than 10W. Fortunately, improvements are being made in color display technology that uses reflective technology. SHARP recently introduced a prototype passive color active matrix display. They have used a technology that forms high-reflectivity contacts using an organic dielectric film deposited on top of the TFT pixels. Typically reflective LCDs place the reflector on the outside of the LCD panel glass, but in their

new device, the pixel electrodes double as reflectors, making it possible to view the screen even from oblique angles while delivering a bright, parallax-free display without generating double images. Since they don't require a backlight, the power consumption is only 50mW. Figure 6-4 shows a 5" (320 x 240 pixels) prototype display which currently has 4 colors, with 512 color versions in development.

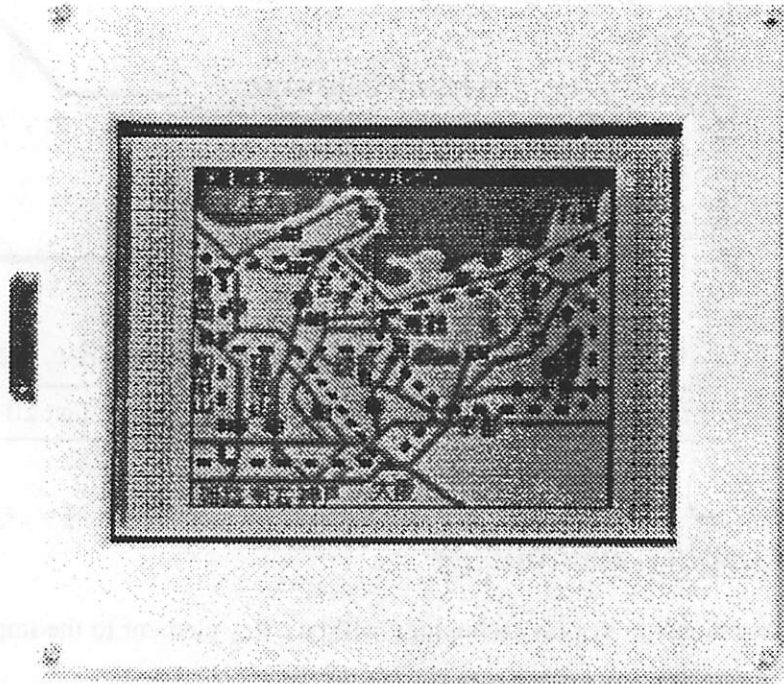


Figure 6-4 : Low-power color TFT display (courtesy of SHARP).

6.1.3 Packaging Technology

Implementing systems using present day technology - single chip packaging using printed circuit boards for interconnection - results in a third or more of the total power being consumed at the chip input/output (I/O), since the capacitances at the chip boundaries are typically much larger than the capacitances internal to the chip. Typical values range from a few 10's of femtofarads at the chip internal nodes, to 10's of picofarads at the chip interface attributed to pad capacitance (approximately 5-10 pF/pin) and PCB traces (3-4 pF/in)[Benson90].

Advances in packaging technology will tremendously improve system performance by lowering power consumption, reducing system delays, and increasing packing density. The emerging multichip module (MCM) technology is one important example in that it integrates many die onto a single high-density interconnect structure, hence reducing the size of inter-chip capacitances to the same order-of-magnitude as on-chip capacitances [Benson90]. This reduces the requirements on the CMOS output drivers, and thus can reduce total power by a factor of 2 to 3. Since this style of packaging also allows chips to be placed closer together, system delays reduce and packing density increases. Thus, with MCM's, the majority of the power is consumed within the functional core of the chip itself, as opposed to the interface.

6.2 System Partitioning for Low-power

Future systems will allow user to be connected in a wireless fashion to fixed multimedia servers. This provides a major degree of freedom for power minimization at the system level which involves the partitioning of computation between the portable terminal and the computer servers on the backbone network. One approach to implement the various general purpose applications (X-server, word processing, etc.) for the portable terminal is to use a general purpose processor. This approach can result in power consumption levels of 10's of Watts; for example, the DEC Alpha chip consumes 30W running at 100MHz. The problem of implementing low power, general purpose processing is primarily being addressed by power down strategies of subsystems, with a limited amount of voltage scaling. This is effective to some extent, but because of the difficulty in determining when an application is really idle, there also needs to be modifications to the software and operating system to fully utilize processor inactivity. Unfortunately, a primary figure-of-merit for computers, that use one of the industry standard microprocessors, has become the clock rate, which results in architectures which are antagonistic to low power operation. On the other hand, as discussed in Chapter 8, parallel architectures operating at reduced clock rates, can have the *same* effective performance in terms of instructions/second at substantially reduced power levels.

Another strategy to reducing the power required for general purpose computation is to effectively *remove* it from the portable. This is accomplished by providing communications capability to computational resources, either at a fixed base station at the other end of a wireless link or back on the network at available compute servers. This is simply the use of the well known client-server architecture, in which the portable unit degenerates to an I/O server (e.g. a wireless X-terminal).

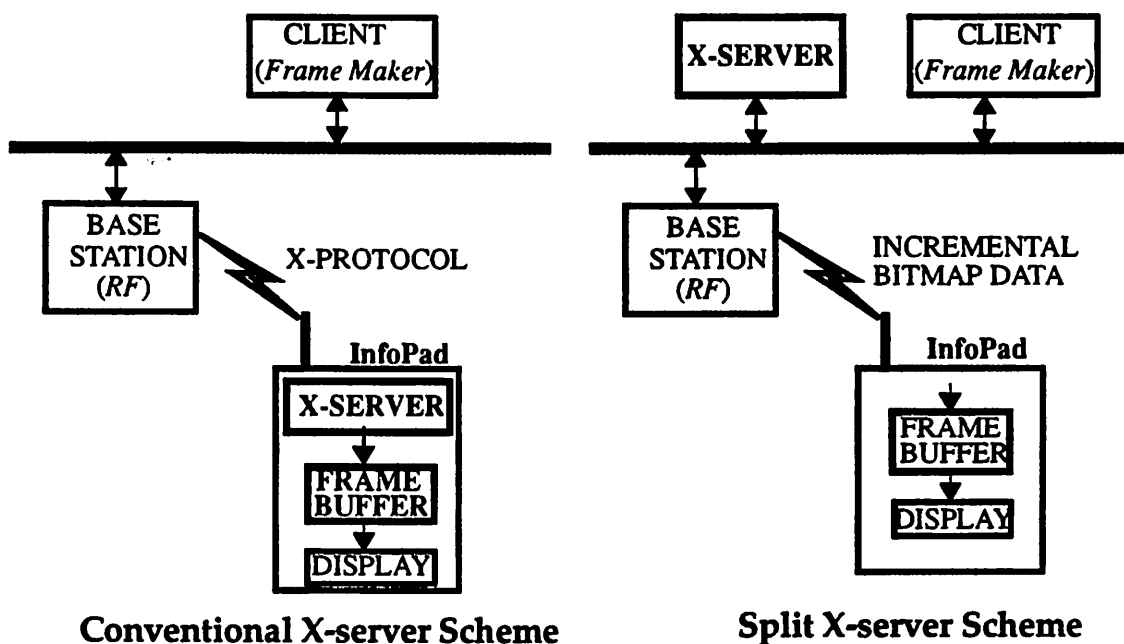


Figure 6-5 : Partitioning of computation: X-server example.

Figure 6-5, for example, illustrates this trade-off for partitioning the X-server computation. On the left side is a conventional implementation in which the X-server runs on a general purpose processor and it communicates with client applications using X-protocol. This approach as mentioned above is very power inefficient. Another approach, as shown on the right, is to use an X-server split between the terminal and the backbone; the basic strategy is to transmit raw pen data over the uplink, perform the processing (X-server as well as client applications) on the base station or on a remote compute server sitting on the wired backbone and transmit incremental bit-maps (the updates to the text/graphics display) from the X-server over the down-link. Hence the processing for supporting text/graphics in terminal, degenerates to frame-buffer interface and

display interface.

The approach of transmitting bitmaps for text/graphics versus X-protocol will indeed tend to slightly increase the overall bandwidth for the downlink traffic. However, the saving in power and robustness to noisy environments (as will be described later) more than compensates for the slight increase in bandwidth. Also, since some client applications like Frame-maker use a bit-map approach anyway, the overall bandwidth of the bit-mapped approach compares favorably with the X-protocol approach. Figure 6-6 shows the average bandwidth (averaged over a 1 sec period) as a function of time for a frame maker session using X-protocol (which represents the traffic for the bit-mapped data plus the overhead for X-protocol). Through high-level simulations, the average bandwidth required for such applications using the split X-server incremental bitmap approach was found to be typically less than 100Kbs.

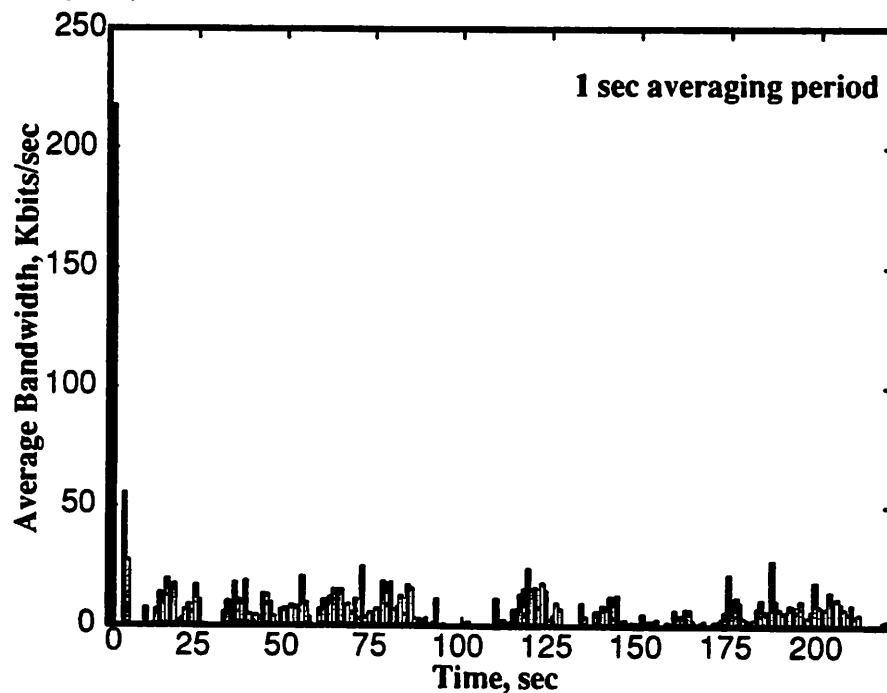


Figure 6-6 : Bandwidth requirement for a Frame-maker session.

Table 6-1 shows a breakdown of bandwidth for various data types. The link is very asymmetrical as seen from the data rates of the uplink and the downlink; no high-rate signals are intended for

transmission from the mobile to the base station.

Table 6-1 : Bandwidth requirement for various data types.

Data Type	Downlink	Uplink
Compressed Color Video	690Kbs	-
Speech	64Kbs	64Kbs
Pen	-	4Kbs
Text/Graphics	Variable < 100Kbs	-

To realize a 1Mbs bandwidth per user, cellular networking techniques are used to achieve spatial reuse [Sheng92]. Unlike conventional sized cells, which have a radius on the order of kilometers, we are using very small picocells (with a radius on the order of 5m). Using picocells implies that we will need about 100Mhz of RF spectrum to support a high density of users (~8-10 users in a 5m radius) while using conventional sized cells would require more than 1 Ghz of RF spectrum.

The use of picocells is another example of significant power reduction that can be achieved by optimizing at the system level since the transmit power is scaled down as the cells move closer together to reduce interference, power consumed in the portable's transmitter to drive the antenna is correspondingly reduced. Whereas existing cellular systems utilize 1 watt transmit power for voiceband RF links in 5 mile cells, a picocellular system with 5 meter cells will require less than one milliwatt to maintain the link.

Similarly to the X-server computation, certain I/O processing tasks such as speech and handwriting recognition are moved to the backbone. PCM sampled speech is transmitted back and forth over the wireless link. Since video is accessed over a wireless network with limited available spectrum (< 1Mbits/sec allocated per user), video is transmitted in a compressed format and therefore decompression has to be performed on the terminal. For I/O processing tasks that have to be performed on the terminal such as video decompression, a system level low-power approach is used to minimize power consumption.

6.2.1 Impact of the Low-power Partitioning Approach on the Interactive Latency

For a low-power partitioning approach in which the general purpose applications are run remotely on compute servers, an important issue is the latency for interactive applications. The goal is to minimize the loopback latency which is the time between pen-to-pad contact and the appearance of digital “ink” on the display. The goal was to keep the maximum latency to 30ms [Burghardt94]. This number was chosen to ensure that the user does not experience excessive delay in the system’s response.

Since pen input is the application with the strictest requirement for latency, it serves as a good timing boundary for the system. The pen data travels from the PAD to the Gateway -> PAD Server -> Pen Server -> Application -> XServer (which generates the incremental bit-maps) -> PAD Server -> Gateway -> PAD. Figure 6-7 shows the histogram of measured roundtrip time for pen data (for the terminal presented later). In this case, most delays are under the 30ms delay target. A similar study of video data and pen data combined has shown that there is not noticeable degradation in the pen’s response time [Burghardt94].

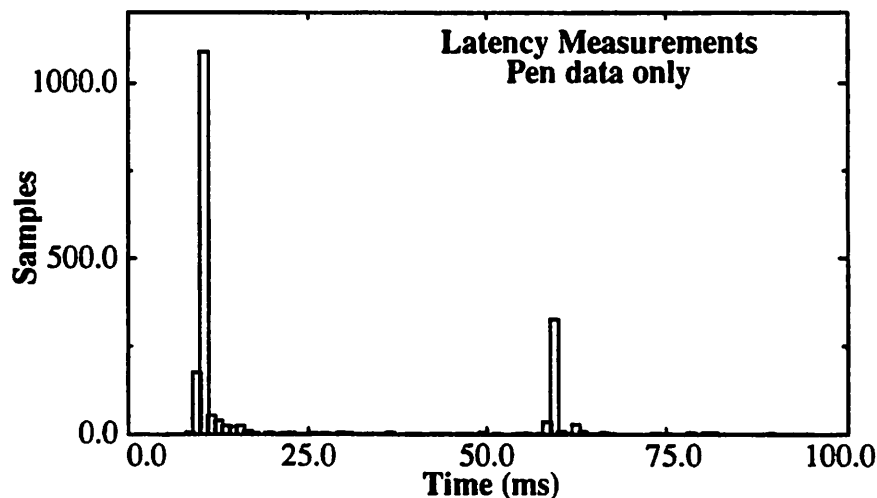


Figure 6-7 : Latency measurement for pen data [Burghardt94].

6.2.2 Impact of the Low-power Partitioning Approach on the Tolerance to High BER

Wireless communications naturally implies a noisy transmission medium. To compare, wired ethernet achieves BER far lower than 10^{-12} , due to the extremely clean nature of the channel. For this system, with all users simultaneously communicating with full power, we have estimated a worst case BER of 10^{-3} [Sheng92]. One approach to deal with high BER is to introduce a lot of redundancy in the data being transmitted, which will unfortunately reduce the bandwidth efficiency in the wireless link. Another approach is to choose representations for the data being transmitted on the link to be robust against noise and add redundancy only to critical portions of the data stream like header information.

For example, consider the robustness of the low-power text/graphics partitioning approach which transmits incremental bit-maps over the RF link (video decompression algorithm selection for a high BER environment is presented in the next chapter). The protocol used is to transmit a base address, length, and bit-mapped data. So errors show up in two forms; errors in the bit-mapped data which appear as dots on the screen, as seen from Figure 6-8b, and errors in the control information (address and length) which appear as lines on the screen. The approach used is to protect the control information and leave the bit-mapped data unprotected. We see that even for an

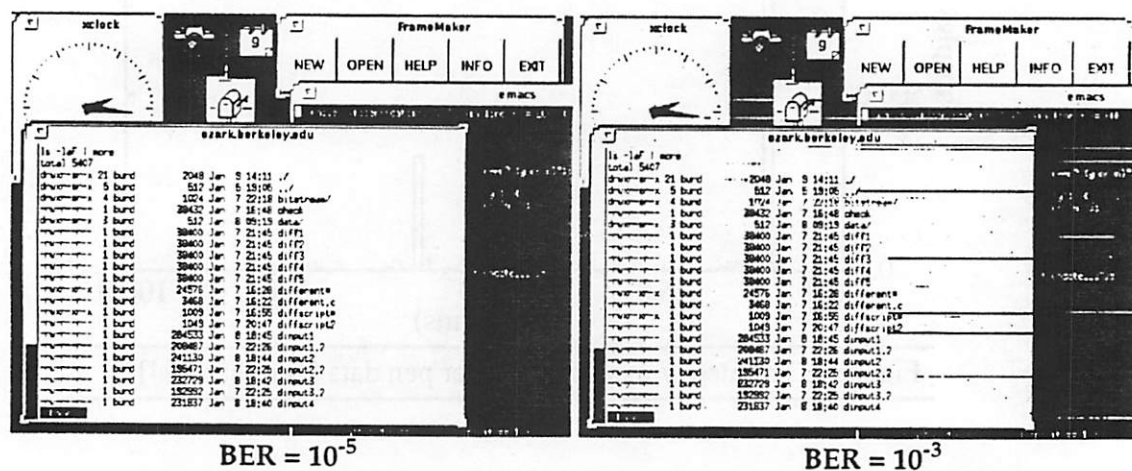


Figure 6-8 : Approach to high BER; transmit more “data” and minimal “control”.

absolute worst case BER of 10^{-3} , we still get “acceptable” performance. Bandwidth can be traded for better quality by providing extra error protection for the control data.

An X-protocol approach would perform poorly for high bit error rates since it has a lot of control information. For example, a draw square command can be interpreted as a draw circle command corrupting large portions of the screen. Therefore, a scheme that transmits X-protocol over the wireless link will have to have a lot of protection and would therefore require retransmission. Therefore the low-power partitioning strategy in which the X-protocol information is converted to bitmaps by compute servers on the backbone network and transmitted over the wireless link is actually better for high BER than the approach of transmitting X-protocol over the wireless link.

6.3 Architecture of the Portable Multimedia Terminal

The portable multimedia terminal described here provides untethered access to fixed multimedia information servers. The terminal is designed to transmit audio and pen input from the user to the network on a wireless uplink and to receive audio, text/graphics and compressed video from the backbone on the downlink. Since the general purpose applications are performed by compute servers on the backbone network, the terminal electronics only has to provide the interface to I/O devices, as shown in Figure 6-9 [Chandrakasan94].

Six chips (a protocol chip, a bank of six 64 Kbit SRAMs for text/graphics frame-buffering, and four custom chips for the video decompression) provide the interface between a high speed digital radio modem and a commercial speech codec, pen input circuitry, and LCD panels for text/graphics and full-motion video display. The chips provide protocol conversion, synchronization, error correction, packetization, buffering, video decompression, and digital to analog conversion. Through extensive optimization for power reduction, the total power consumption is less than 5mW.

The protocol chip communicates between the various I/O devices in the system. On the uplink, 4

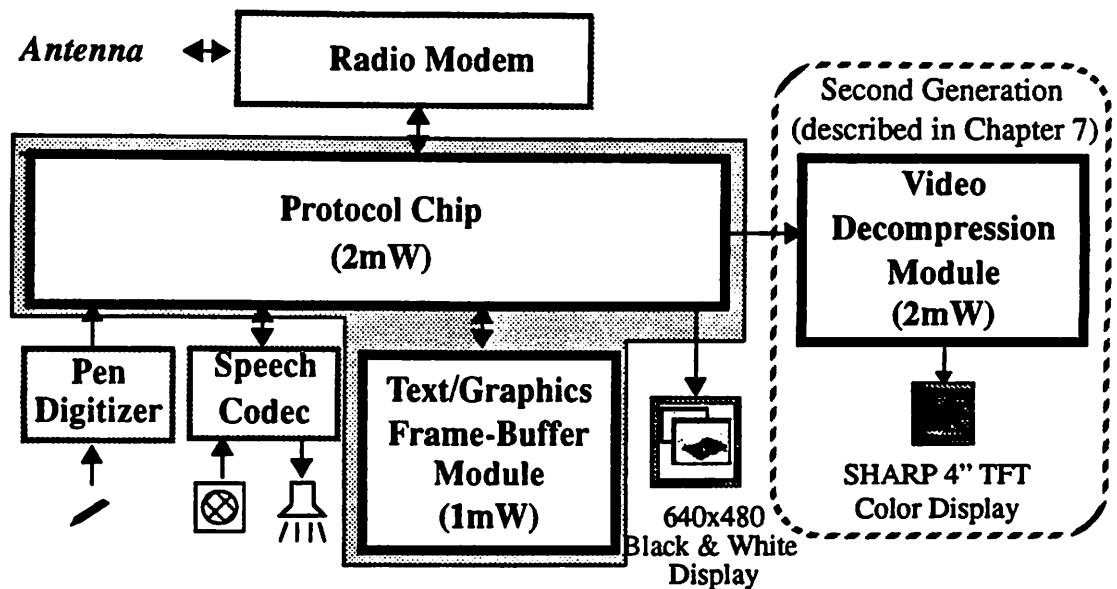


Figure 6-9 : Block Diagram of the InfoPad Terminal.

Kbps digitized pen data and 64 Kbps, μ -law encoded speech data are buffered using FIFO's, arbitrated and multiplexed, packetized, and transmitted in a serial format to the radio modem. On the down link, serial data from the radio at a 1Mbs rate is depacketized and demultiplexed, the header information (containing critical information such as data type and length) is error corrected and transferred through FIFO's to one of the three output processing modules: speech, text/graphics, and video decompression.

The speech module inside the protocol chip transmits serial data from the FIFO to a codec at a rate of 64Kbps. The text/graphics module also inside the protocol chip handles the protocol used to transmit the bitmaps for text/graphics. The text/graphics module corrects error sensitive information using a Hamming (7,4,1) code, and generates the control, timing and buffering for conversion of the 32 bit wide data from the text/graphics frame-buffer (implemented using six low-power 64Kbit SRAM chips) to the 4-bits at 3MHz required by a 640x480, passive LCD display. The final output module is the video decompression, which is realized using four chips. This module includes all of the circuitry required to take the compressed video stream from the radio and convert it to the decompressed analog RGB video format required by the 4 inch color

LCD display. The decompression algorithm is vector quantization, which involves memory look-up operations from a codebook of 256 4x4 pixel patterns.

The first generation terminal, described in this chapter, only supports pen input, speech I/O and text/graphics output. The video decompression module has been implemented and is described in the next chapter. Future terminal designs will integrate the chips described in this chapter and the video decompression chips described in the next chapter.

6.4 Protocol Chip

The protocol chip, shown in Figure 6-10, communicates between the various I/O devices in the system. It provides the interface between a commercial radio modem and the pen digitizer, the speech codec, and the 640x480 text/graphics LCD display. It also controls the custom frame-buffer of the text/graphics display and provides the necessary interface between the frame-buffer and the LCD display. The radio receiver and transmitter modules are isolated from the I/O processing modules using 32-bit FIFOs. The text/graphics FIFO has one extra bit to delimit packets.

The architecture driven voltage scaling strategy presented in Chapter 3 is used to scale the supply voltage to the 1-1.5V while meeting the computational throughput requirements. Since all of the circuits operate at a supply voltage which is less than the sum of the NMOS and PMOS threshold voltages ($V_{tn} = 0.7$, $V_{tp} = -0.9$), the devices can never conduct simultaneously for any possible input voltage values, *eliminating* short-circuit power. Therefore, the primary focus was on minimizing dynamic power consumption. In addition to minimizing the supply voltage, this chip exploited signal statistics to reduce the total number of transitions. The physical capacitance is also reduced by utilizing a low-power cell library that features minimum sized transistors and optimized layout.

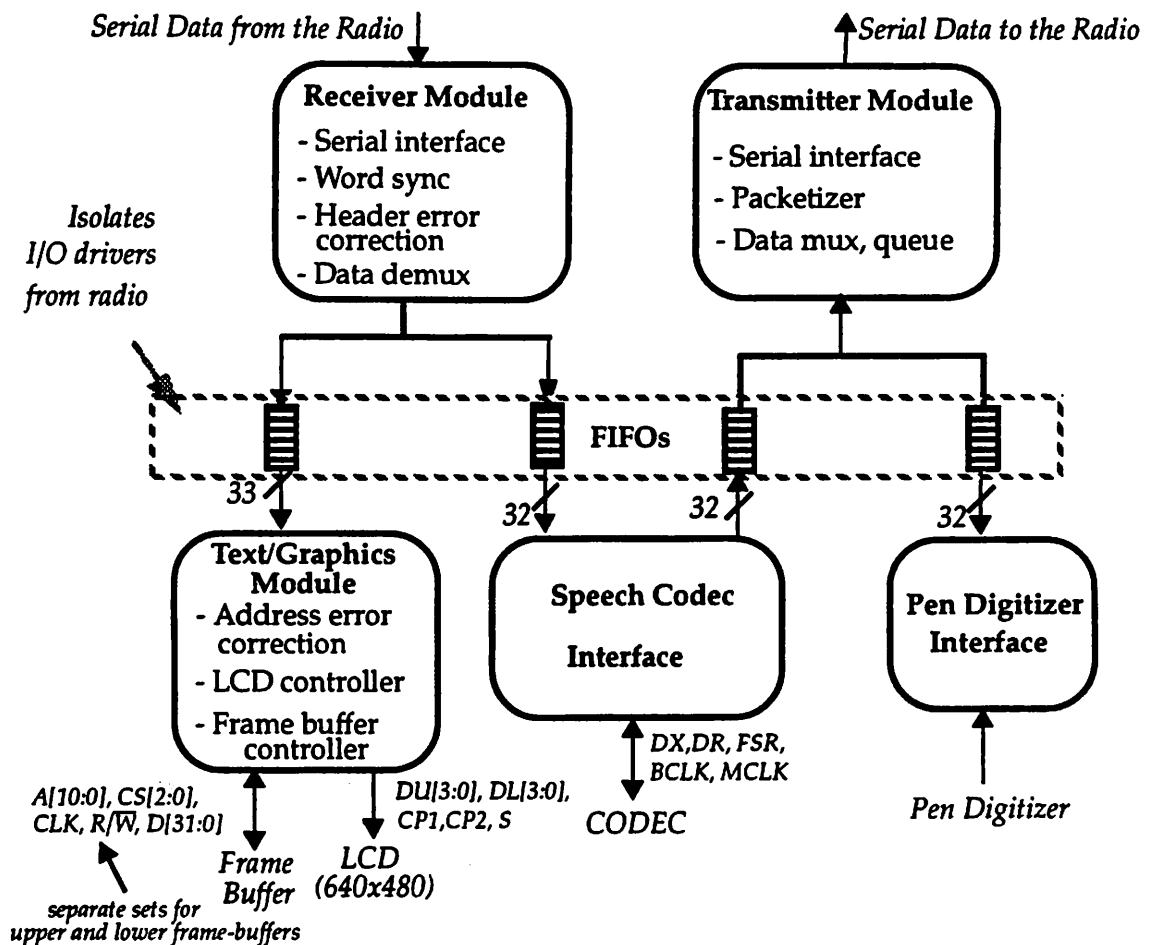


Figure 6-10 : Block diagram of the protocol chip.

6.4.1 Radio Receiver Module [Burd93]

On the down-link (from the wired network to the terminal) three types of data packets are sent: text/graphics data for the black and white display, speech data for the codec and compressed video data. The protocol chip handles only the first two data types and the video data is processed by the video controller described in the next chapter. Future versions of the protocol chip will integrate the protocol processing of the video data.

The receiver module takes as input a serial data stream containing the text/graphics data and the

speech data from the radio modem module. Figure 6-11 shows the data packet protocol coming into the protocol chip. Each protocol chip packet has a header which contains a synch pattern used for word alignment, a packet length indicating the number of 32-bit data words in the packet, a control field which is currently not used and data type packet indicating the type.

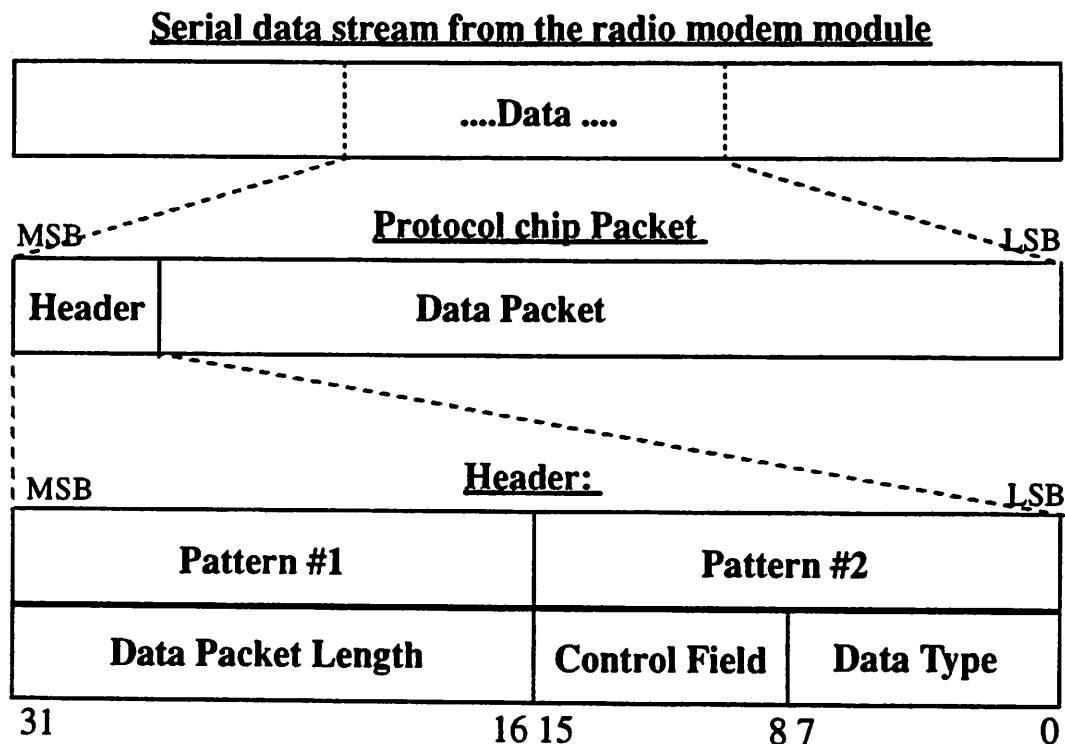


Figure 6-11 : Packet format coming into the protocol chip.

If either of pattern#1 or pattern #2 matches, then the data following the synch pattern is aligned on word boundaries. The patterns are hard-wired on the chip when auto-booting is enabled. The hard-wired patterns are set to: pattern #1 = 0x8765 and pattern #2 = 0x4321. Through two external I/O pins, new patterns can be loaded. Since the chip looks for these patterns, the base-station must guarantee that these patterns do not appear in the data (one approach is to flip a bit in the data if the original data contains the synch pattern). If due to bit error rates on the wireless channel, the data gets interpreted as a synch, then synchronization will be lost till the next data packet is sent - the receiver module will re-synchronize on the header of the next packet.

The length field indicates the number of 32-bit words in the data packet that follows. Some modules like the text/graphics require information about packet boundaries (as described in the next sub-section, the text/graphics packets has some header information encoded in the packet) while other modules like speech do not require information about the packet boundaries. The receiver module performs the function of delimiting the packets - that is, it keeps track of the length and sets an End of Packet flag (EOP) in an extra bit of the text/graphics FIFO.

The data type field is used to encode the packet type. There are two packet types defined for the downlink receiver module in the protocol chip: speech (x00) and text/graphics (0xff) (video packet types are defined for the controller chip in the next chapter). Error correction is used to protect the type field and up to 2 bits can be corrected. The error corrected data type field is decoded and is used to directed the following data in the packet to either the text/graphics module or the speech module.

The receiver module and the I/O modules communicate using 32-bit FIFO's. The receiver module monitors the "fullness" of the FIFOs to determine if data is to be pushed. The receiver has various guard modes to protect from loss of synchronization in the radio module.

Figure 6-12 shows the dataflow in the receiver module. Serial data SERIAL_IN is shifted in using a scan register with the SCAN input connected to VALID_DATA. After each bit-shift, the 32-bit parallel word at the output of the register is compared with a fixed pattern (which can be changed using another scan register through PAT_IN and PAT_LOAD). If a match is detected with either the top 16-bit word or the bottom 16-bit word, the signal PAT_MATCH is asserted. This will indicate the beginning of new packet. The next 32-bit word is then loaded into another register from which the length and data types are decoded. The length is decremented until enough words in the packet are pushed into the appropriate FIFO and a control bit of the FIFO is set to indicate the end of the packet (EOP). As will be described in the next section, only the text module uses this EOP flag and the speech module does not require information regarding packet boundaries.

The receiver module also has guard modes using control circuitry (not shown in Figure 6-12) to detect errors such as a zero length packet.

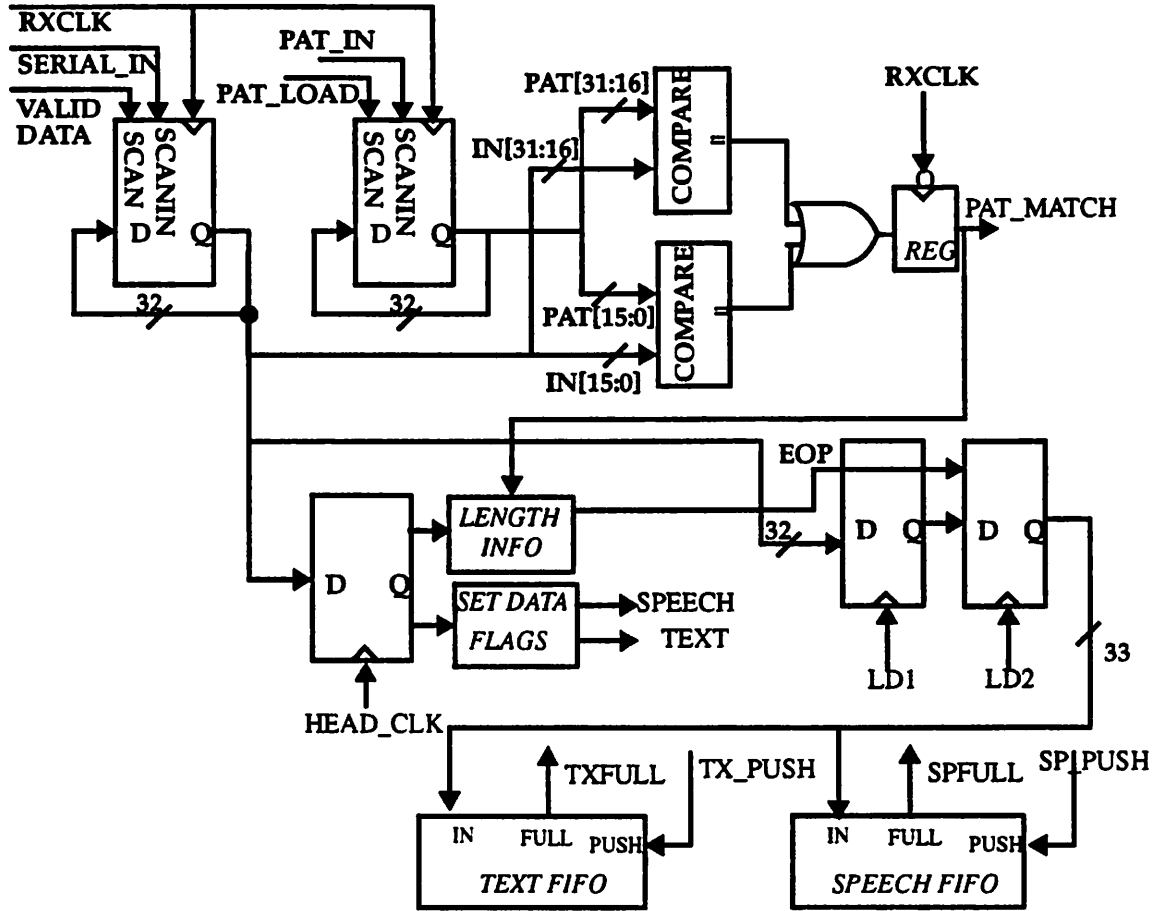
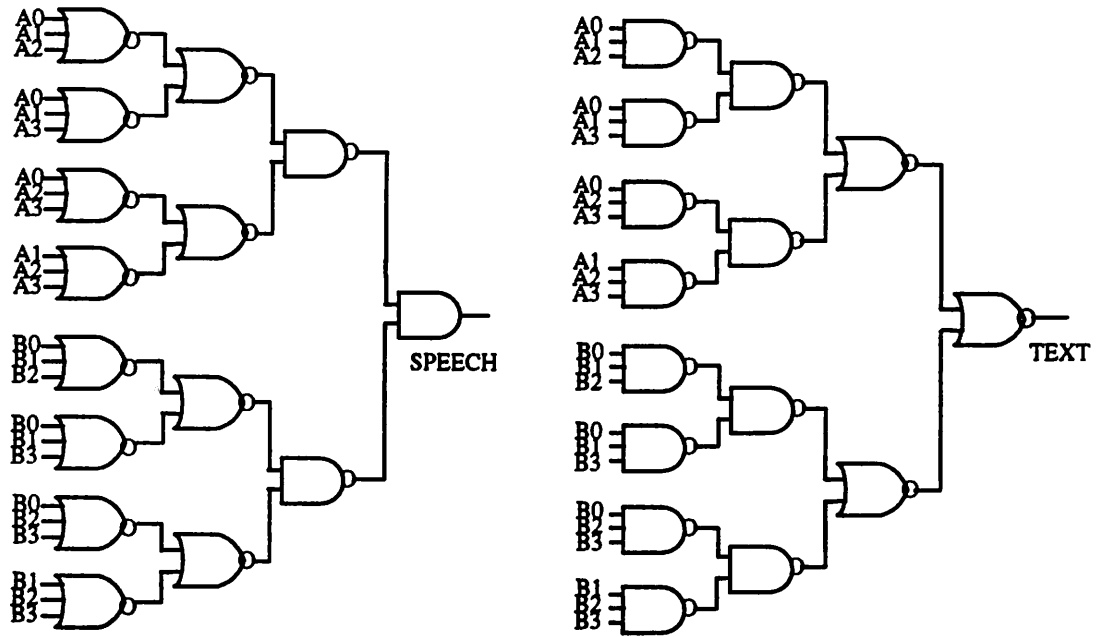


Figure 6-12 : Datapath for depacketization and demultiplexing.

Figure 6-13 shows the decoding logic of the two data types (the SET DATA TYPES block), speech (x00) and text/graphics (0xff). Error correction is used to protect the type field and up to 2 bits can be corrected.



DATATYPE FIELD (8 lower bits of header latch): <A0,A1,A2,A3,B0,B1,B2,B3>
LSB MSB

Figure 6-13 : Decoding of data types.

6.4.2 Text/Graphics Module

The text/graphics module performs the interface between the radio receiver and the 640x480 black and white LCD display. Figure 6-14 shows a block diagram of the text/graphics module in the protocol chip.

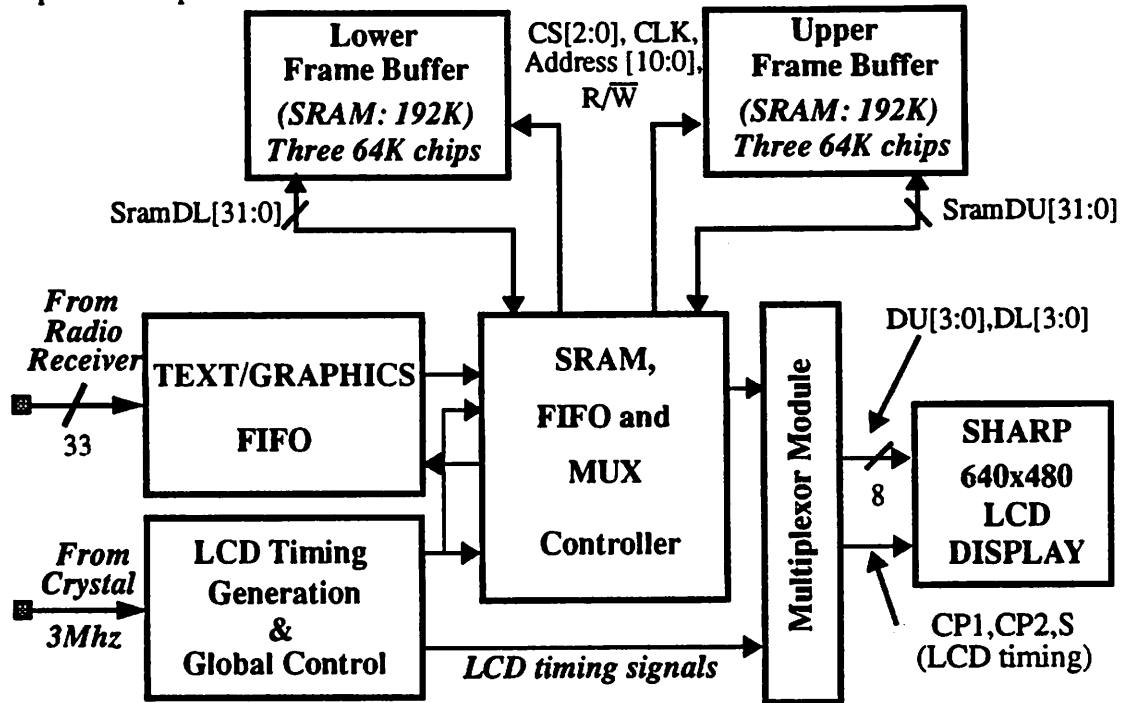


Figure 6-14 : Block diagram of the text/graphics module.

LCD Timing generation

It takes a reference clock from an external crystal oscillator and generates the timing signals required for the LCD display. The minimum refresh rate for the LCD display is 8ms while the maximum refresh rate is 16.9ms. The display is a split panel display in which the 640x480 screen is broken into two 640x240 screens. Given this, the input clock can have any frequency value between 2.5MHz and 4MHz. A frequency of 3MHz was used in the system. Figure 6-15 shows the three synchronization timing signals required for the LCD display: a pixel clock (CP2), a line clock (CP1) and a frame clock (S). CP2 is a clock signal to synchronize the LCD pixel data DU

and DL, which are latched on the falling edge of the clock CP2 (data is latched out from the protocol chip on the rising edge and therefore there is 1/2 cycle available to set up the data). Since four 1-bit pixels are sent to the display in parallel (for each half of the display), there are $640/4$ clock pulses on CP2 per line.

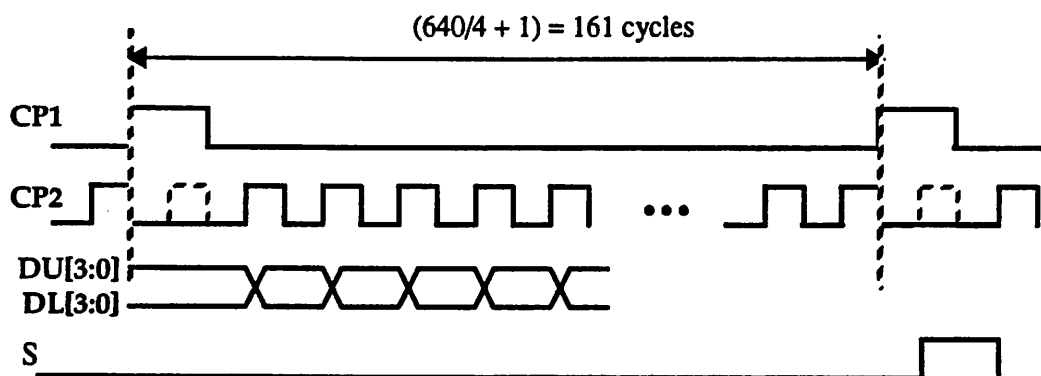


Figure 6-15 : Horizontal timing signals for the SHARP LCD Display.

Figure 6-16 shows the schematic of the circuit to generate the timing signals CP1 and CP2 from a master clock. Note that one pulse of the clock is suppressed ever 161 pulses of the master 3MHz clock to generate the LCD pixel clock, CP2. A loadable counter whose input is fixed to the value 95 is used to count 161 cycles (i.e the output of the counter sequences from 95 to 255). CP1 is generated by latching the carry out of the counter (Cout) which is high for one clock period for every 161 clock cycles. This loadable counter implementation reduces the switched capacitance compared to an implementation which uses a counter that sequences from 0 to 160 and a comparator which checks the output of the counter with a constant and generates the reset signal for the counter. CP2 is generated by gating the master clock and the line synchronization signal, CP1. The signal CP1 is delayed through a buffer before gating the clock to eliminate any glitching on CP2. Delayed versions of the timing signals (CP1D2 and CP2D2) are also generated to compensate for the 2-cycle pipeline delay in the system. The signals CP1 and CP2 are used for timing inside the text/graphics module while the delayed signals are used for synchronization

inside the LCD display.

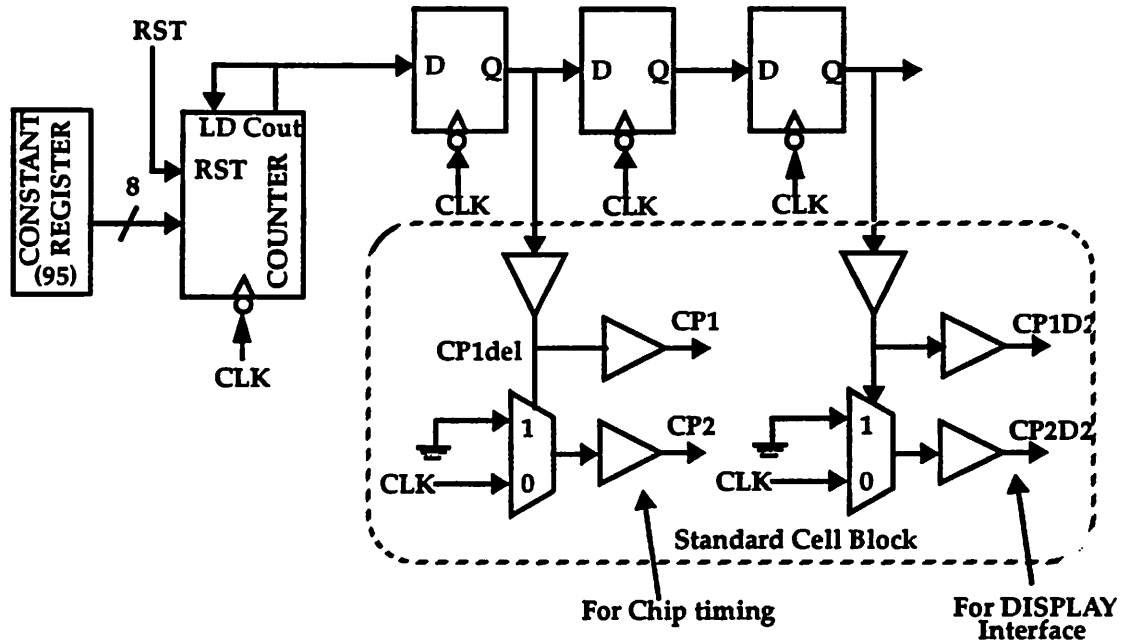


Figure 6-16 : Line synchronization clock generation.

Figure 6-17 shows the timing of the vertical synchronization signal, S, for the LCD display. A pulse is created every 240 lines (rather than 480 since a split panel display is used). Note that the vertical synchronization pulse is skewed by one line with respect to the line clock, CP1. This is because a whole line is buffered in the LCD and all columns are driven in parallel.

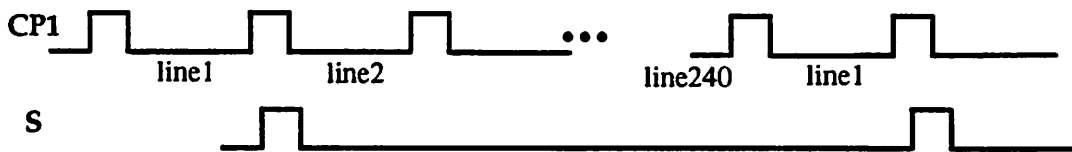


Figure 6-17 : Vertical timing signals for the SHARP LCD Display.

Figure 6-18 shows the schematic of the circuit to generate the field timing signals S based on the master clock and CP1del of Figure 6-16. Similar to the generation of the horizontal clock, a loadable counter is used to sequence from 16 through 255 (i.e count 240 lines) and generate the field clock signal, S. Note that the clock to the counter is CP1 (whose frequency is 161 times lower

than the master clock) and therefore the power consumption is very low compared to a scheme which uses the master clock as the clock and a count enable that is asserted once every 161 cycles. This shows an example of how the clock frequency can be tuned for each module. The signal S is used for synchronization inside the text/graphics module. The signal SD2L1 is sent to the LCD display, which is the signal corresponding to S in Figure 6-17, but delayed by 2 pixels for compensating the pipeline delay.

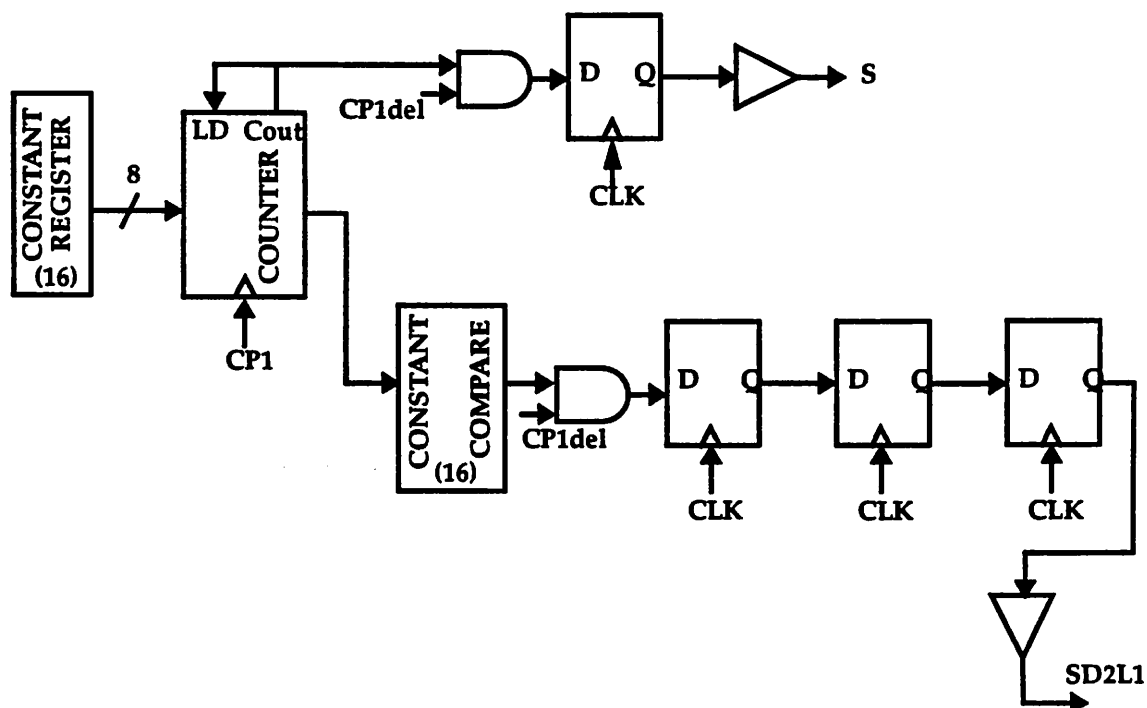


Figure 6-18 : Field synchronization clock generation.

Choice of Frame-buffer I/O Bitwidth and Supply Voltage

Reducing the power supply voltage has the greatest impact on the power consumption. Unfortunately, this simple solution to low power design comes at the cost of increased gate delays and therefore lower functional throughput. Figure 6-19 shows the normalized access time for a 64Kbit SRAM as a function of supply voltage V_{dd} .

As presented in Chapter 3, an architecture driven voltage scaling strategy using parallelism and

pipelining to compensate for the increased gate delays at reduced supply voltages can maintain functional throughput. Parallelism and pipelining were used extensively in this chipset for both arithmetic computation and memory access, allowing the supply voltages to scale down to 1.1V.

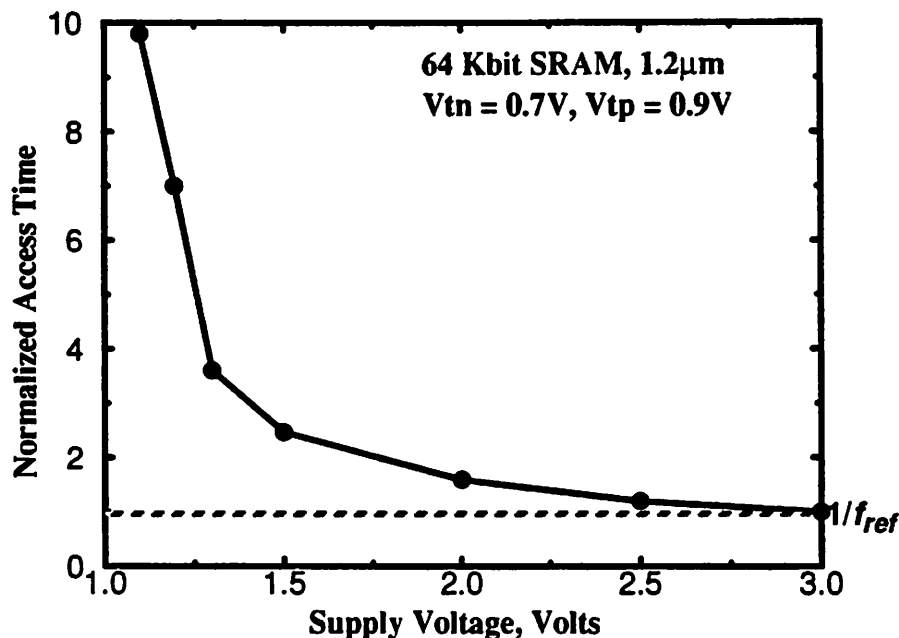


Figure 6-19 : Normalized access time vs. V_{dd} for a 64Kbit SRAM.

Parallelizing arithmetic computation was presented in previous chapters; here an application of parallelism to memory access is presented, which involves reading black and white pixel data from the text/graphics frame-buffer memory to the 640x480 LCD display. The LCD display is a split-panel display in which the top half (640x240) and bottom half are addressed in parallel. Each half requires 4-bits (or 4-pixel values) at a rate of 3MHz.

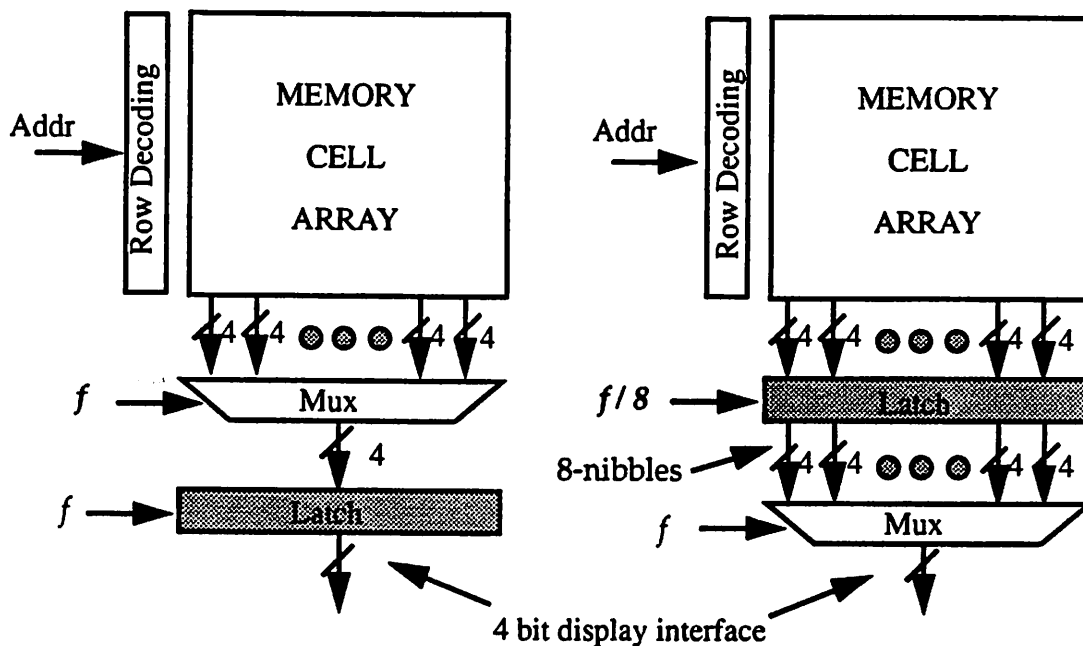


Figure 6-20 : Parallel memory access enables low-voltage operation.

Figure 6-20 shows two alternate schemes for reading the required 4-bits of data from memory at the throughput rate, f . On the left side is the serial access scheme in which the 4-bits of data are read out in a serial format and the memory is clocked at the throughput rate f . For this single-ported SRAM frame-buffer implementation (in which the read/write data and address busses are shared) utilizing a serial access scheme can operate at a supply voltage of 3V. Another approach (shown on the right side) is to exploit the sequential data access pattern for the frame-buffer and access several 4-bit nibbles in parallel, allowing the memory to be clocked at a lower rate. For example, reading eight nibbles in parallel from the memory allows the memory to be clocked at $f/8$ without loss in throughput. The latched data (8 nibbles) in the protocol chip is then multiplexed out at the throughput rate. This implies that the time available for the memory read operation for the parallel implementation is 8 times longer than the serial scheme and therefore the supply voltage can be dropped for a fixed throughput. The parallel version can run at a supply voltage of 1.1V (which corresponds to the voltage at which the gate delays increase by a factor of 8 relative

to the serial access scheme running at 3V from Figure 6-19) while meeting throughput requirements. It is interesting that architectural techniques can be used to drive the supply voltage to such low levels even with a process that has $V_{tn} = 0.7V$ and $V_{tp} = -0.9V$. If the process could be modified to reduce the threshold voltage, the power supply voltage and therefore the power consumption can be further reduced.

Text/Graphics Pixel Dataflow

The text/graphics bitmapped data that is transmitted over the radio is sent to the text/graphics FIFO by the receiver module. The FIFO is then read out and the pixel data is stored in the frame-buffer. Figure 6-21 shows the dataflow for the pixel data. The SRAM frame-buffer is single-ported and therefore there is a single time-multiplexed I/O data bus where half the cycles are allocated for reading and half for writing. Each half of the display has its own frame-buffer (an upper frame-buffer and a lower frame-buffer) each consisting of three 64Kbit SRAM chips. Therefore the 32-bit FIFO output bus is routed to two sets of I/O pads, one for the upper frame-buffer and one for the lower frame-buffer. The lower order 28-bits of the FIFO output bus is also routed to the SRAM address generator since the text/graphics packet also contains information about the starting location (called the start or base address) in memory for the pixel data in each packet.

The read from the frame-buffer is periodic while the write into the SRAM is dependent on updates to the screen. The FIFO controller generates the timing signals for the FIFO. If the FIFO is not EMPTY, the data is POPed from the FIFO (by asserting the `FifoPop` signal) and an `SramWritePending` flag is set indicating that pixel data is available to be written into the frame-buffer. During the write cycles of the SRAM, based on the `SramWritePending` flag, data is conditionally written into the SRAM and the `SramWritePending` flag is reset. Since the write into the frame-buffer is much faster than the input rate into the FIFO from the radio, the FIFO depth can be small. A depth of 16 was chosen.

Bit 32 of the FIFO represents the End of Packet flag which is used to delimit the text/graphics

packets. The EMPTY and EOP signals are sent to the text/graphics controller from which the POP signals and other signals such as OEN (output enable) are generated.

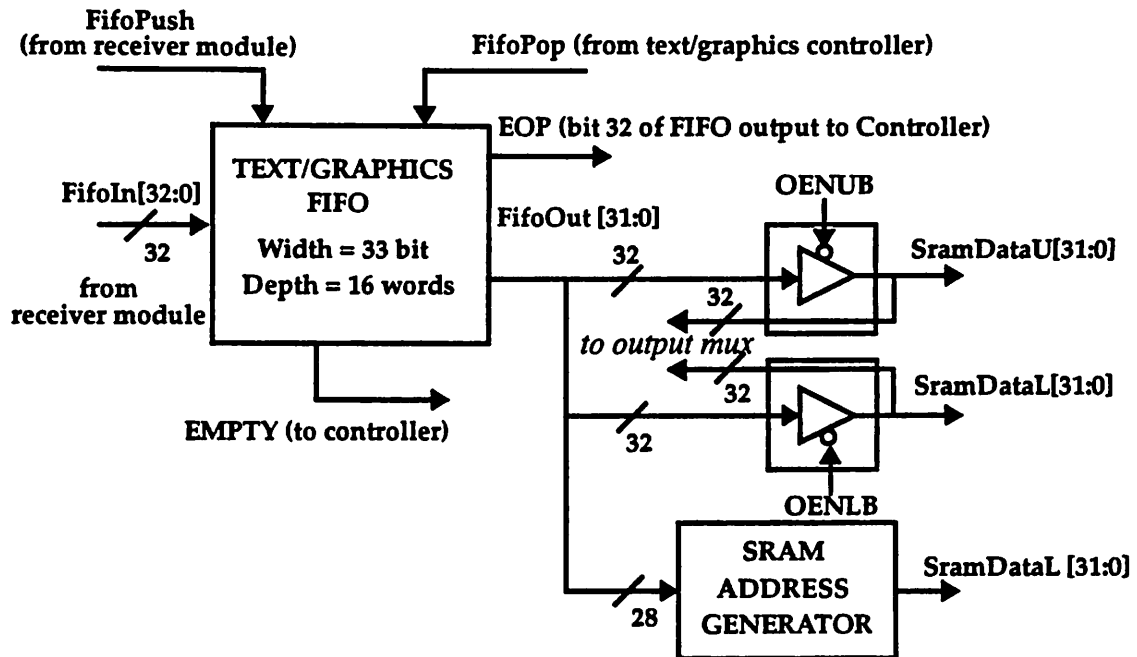


Figure 6-21 : Data and address busses for the text/graphics module.

Figure 6-22 shows the flow of data from the memory to the display (i.e the read operation). As described in the previous section, the frame-buffer uses 32-bit parallel data words to operate at reduced supply voltages. The display requires 4-bits or 4-pixels every cycle and therefore the memory read operation takes place once every 8 display clock cycles. The 32-bits read from the frame-buffer is latched in the protocol chip at the reduced clock rate (OutLatchCLK has a frequency of $3\text{MHz}/8 = 375\text{KHz}$) and then multiplexed out at 3MHz to the LCD display. An identical datapath is used for the upper frame-buffer. The select signals are generated from the controller and is just the output of the counter.

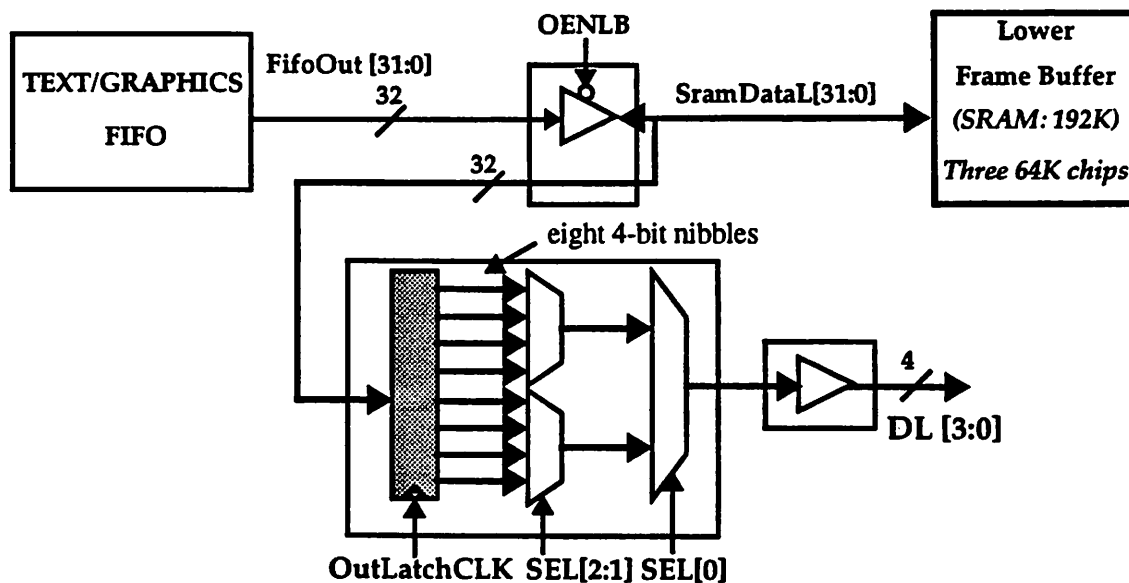


Figure 6-22 : Conversion from 32-bit frame-buffer data to 4-bit LCD data.

The place and route of the data and control busses is optimized such that signals that have high switching activity (such as clocks) are assigned short wires and signals that have low switching activity are allowed to have long wire; i.e the cost function $\sum \alpha C_f$ is minimized. Figure 6-23 shows an example that involves the routing the display data and clocks, the 32-bit upper frame-buffer data bus, and the address bus and chip selects for the upper frame-buffer from the text/graphics module of the chip core to the pads for the protocol chip, shown in Figure 6-34.

The split-panel display requires 8-bits (4 for the top half and 4 for the bottom half) at a rate of 3Mhz, while each SRAM module uses 32-bits clocked at $3\text{Mhz}/8 = 375\text{KHz}$ (using the parallel access scheme described earlier). An activity factor (for both 0->1 and 1->0 transitions) of 1/2 is assumed for the data. The address bits are also clocked at 375KHz but they have a much lower switching activity since the accesses are mostly sequential, coming from the output of a counter - the address bus is time-multiplexed between the read and write addresses for the SRAM, but since the write into the text/graphics frame-buffer is relatively infrequent, the data being placed on the address bus typically just the read address and is therefore very temporally correlated (the number

of transition for a counter output per cycle is $1 + 1/2 + 1/4 + \dots \approx 2$ since the lsb switches every cycle, the next lsb switches every other cycle, etc.).

As seen from the plot of physical capacitance as a function of distance from core to pads, the high activity display data and display clock which have high switching activity are assigned the shortest wire lengths, while the SRAM address, which has an activity factor 16 times lower, is allowed to have the longest wires. This results in a 60% reduction of power compared to a scheme that assigns the address wires to be shortest and the display data wires to be the longest.

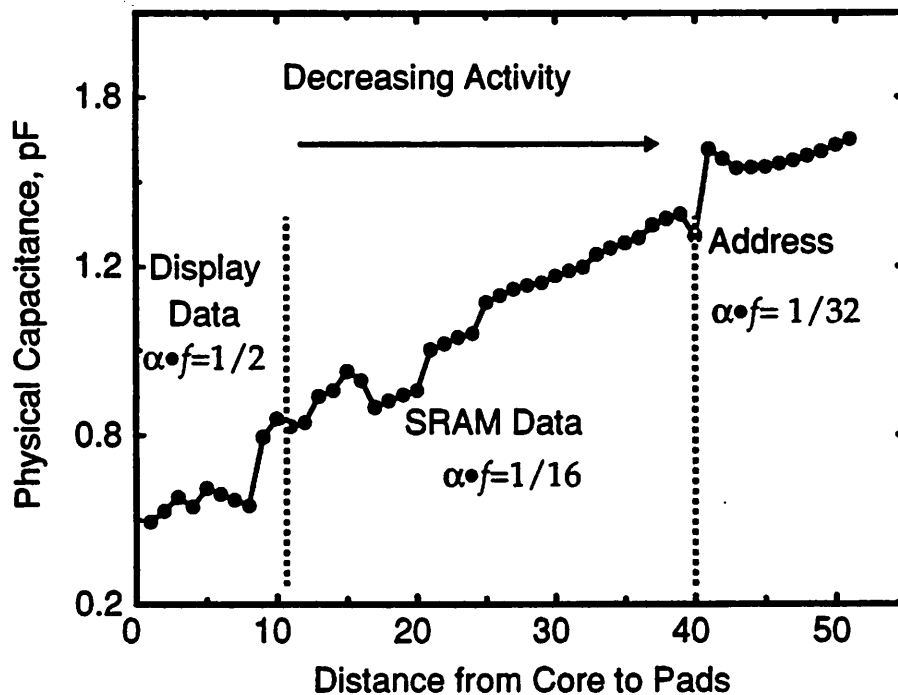
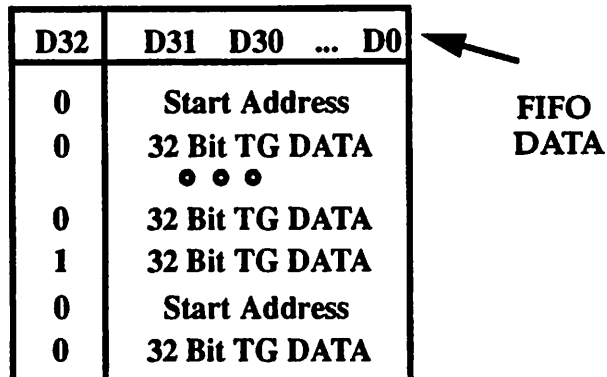


Figure 6-23 : Optimizing placement for low-power: routing large data/control busses.

SRAM Address Generation and Error Correction

Figure 6-24 shows the protocol of the text/graphics packet. The first entry is a start address which indicates the location where the pixel data that follows is written into. The start address also contains a bit which indicates the BANK number which determines if the data should be written into the upper frame-buffer module or the lower frame-buffer module. The pixel data is written

into consecutive locations starting from the start address in the frame-buffer.



- End of Packet Flag (EOP):

Bit 32: 1 = Last Data Entry in Packet

- Start Address (in unencoded format):

Bit 13: Bank #, 0 = Lower & 1 = Upper

Bit [12:0]: Address

Figure 6-24 : Application specific protocol: text/graphics information.

While address information in the text/graphics packet is very sensitive to channel errors (since errors can cause the data to be written in the wrong area of the screen), bit-mapped data is not very sensitive since errors in data appear as dots on the screen. Therefore, to enable a bandwidth efficient wireless protocol, only the address information is error corrected and the bit-mapped data is left unprotected. The start address can be error corrected using a Hamming (7,4,1) code. On the encoder the 14 bits of the unencoded base address is padded to 16 bits and encoded using the Hamming code to get a 28bit word (i.e. each 4-bit nibble is coded to 7bits). On the protocol chip, the start address is decoded using a Hamming decoder module if the control signal ERRORC is set through an external pin. If ERRORC is zero, then the Hamming decoder is bypassed and the base address is assumed to be unencoded. This allows us to experiment with the effect of error correcting the text/graphics data. The effective bandwidth is the same using either approach since a 32-bit word alignment scheme is used.

The capacitance switched in the error correction module is minimized by using gated clocks.

Figure 6-25 shows a block diagram of a power efficient implementation of the error correction function. A register is introduced at the output of the FIFO and a gated clock (coming from the controller) is used to enable the error correction module to process only the address information; the ECC is shut down during the rest of the time. In this manner, the inputs to the ECC are not switching when the data portion of the protocol is accessed from the FIFO. Since typically, the address is only a small portion of the text/graphics packet, significant power savings is possible. Gated clocks were used in many circuits to tune the frequency and hence clock load for each module. Gated clocks are not usually used because of the timing skew introduced; this is not a problem in our approach since we use a low clock rate methodology.

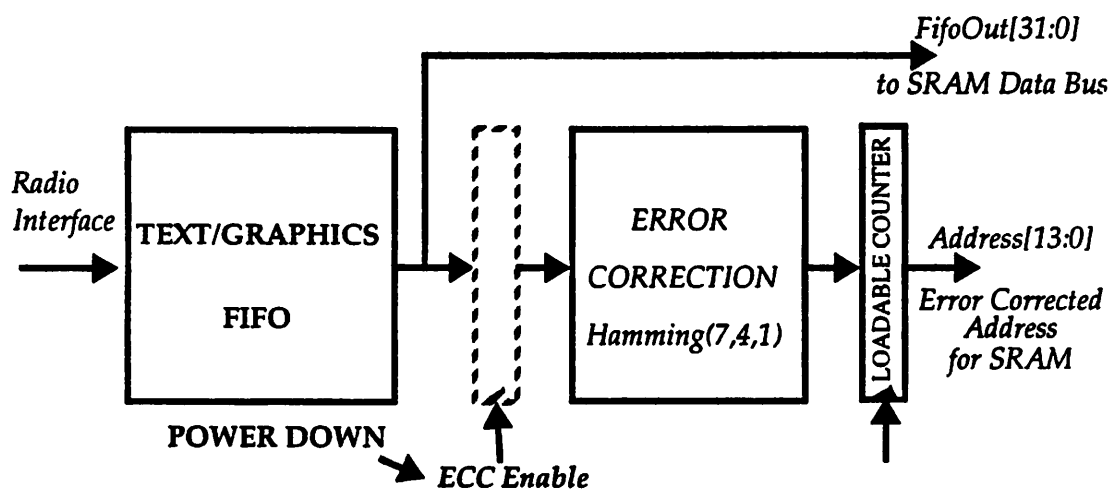


Figure 6-25 : Gated clocks are used to shut down modules when not used.

Figure 6-26 shows the address generation datapath for the SRAM frame-buffer. Two counters are used: a read-address counter and a write-address counter. The read address counter is reset based on the field sync signal (i.e the counter output is forced to 0 at the beginning of the frame). Then every 8 cycles, the counter is incremented. The write address counter loads an error corrected start address and increments each time a new pixel word in the text/graphics packet has to be written to the frame-buffer. When a new text/graphics packet is read, the new base address is loaded. The address lines for the upper and lower frame-buffers are multiplexed between the output of the read and write counters. The lower order bits are sent directly to the address bus. The upper two bits are

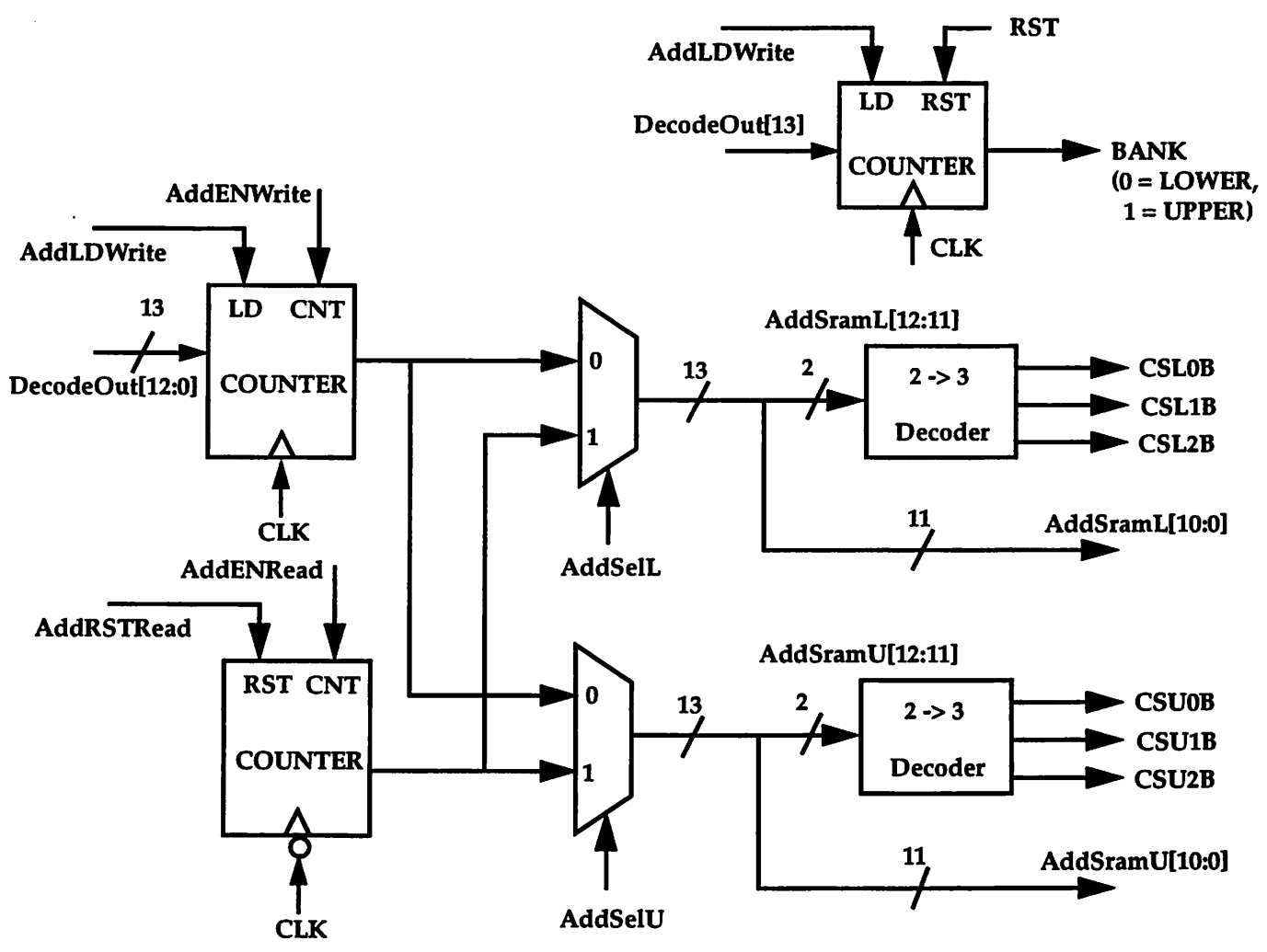


Figure 6-26 : Address generation for the SRAM.

decoded and active low chip select signals are generated for the 3 memory chips in each bank. The BANK information is sent to the controller which generates the control signals for writing.

Text/Graphics Controller

The text/graphics controller generates the control signals required for interfacing with the text/graphics FIFO, the R/\overline{W} signals and the clock signals for the frame-buffer SRAMs, the OEN of the tri-state I/O pads that interface the protocol chip with the frame-buffer chips, the signals for loading and controlling the address generator units, and the clock signals for the output latch.

The controller also keeps track of the packet information through a `FifoState` register (`FifoState = 0` implies that the next data entry read from the FIFO is the start address and `FifoState = 1` implies that the next entry is a pixel data word). As soon as a data entry is read from the FIFO, the `FifoState` register is set to 1, unless a zero length packet was detected in the radio receiver module - in which case, the `FifoState` remains at 0. When in the 1 state, all data coming across the FIFO is pixel data, until the `FifoState` is reset to 0 by an EOP flag.

The SRAM is read once every eight cycles and therefore three bits of state information is required. A 3-bit counter is used for generating the state information which is synchronized on CP1. The timing of the `CurrentState` relative to CP1 and CP2 is shown in Figure 6-27. The read from the SRAM is synchronous while the write is asynchronous. That is, the read clock pulse always occurs every eight cycles, when `CurrentState = 0`. The read uses two clock cycles and the memory data is latched in the protocol chip two clock cycles later. This output latch uses a clock (`OutLatchCLK`) which is delayed by 2 cycles relative to the `SramCLK`. The write to the SRAM occurs conditionally. When a data word is POPed from the FIFO, and `SramWritePending` flag is set indicating that data is ready at the output of the FIFO to be written into the SRAM. If this flag is set, then based on the `BANK` information, a clock pulse will occur either on `SramCLKU` or `SramCLKL`. The output enable of either the upper or lower data I/O pads will also be asserted for the write period. As soon as the data is written into the frame-buffer, the `SramWritePending` flag is reset, allowing another word to be read from the FIFO if it contains new data.

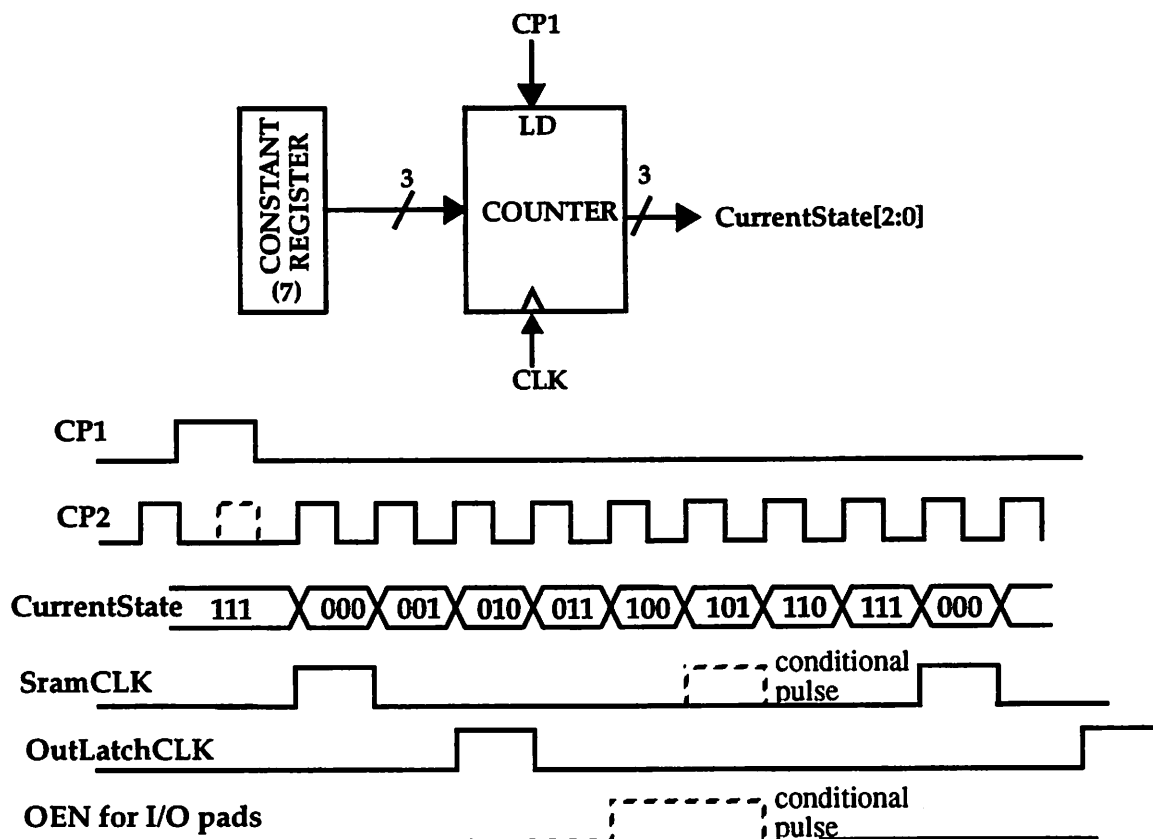


Figure 6-27 : State information and SRAM clock timing.

6.4.3 Speech I/O Module

InfoPad supports bi-directional audio I/O using 8-bit μ -law samples at 8kHz. A commercial speech codec from Motorola (MC14554) is used to interface with the speaker and the microphone. The speech I/O module on the protocol chip provides the interface between the radio modem module and the commercial codec. Figure 6-28 shows the interface signals between the speech module on the protocol chip and commercial speech codec. The speech I/O module takes a master clock from an external crystal oscillator, $f_{clk} = 2.048\text{MHz}$, and generates the lower frequency clocks and timing signals required by the speech codec. DX and DR are the serial data signals to and from the codec. An external power down pin is used to power down the uplink speech module

which conserves power and bandwidth. BCLK is a 50% duty cycle 128KHz clock obtained from dividing the 2.048MHz clock. FSR is a 8KHz clock with a duty cycle of 1/16.

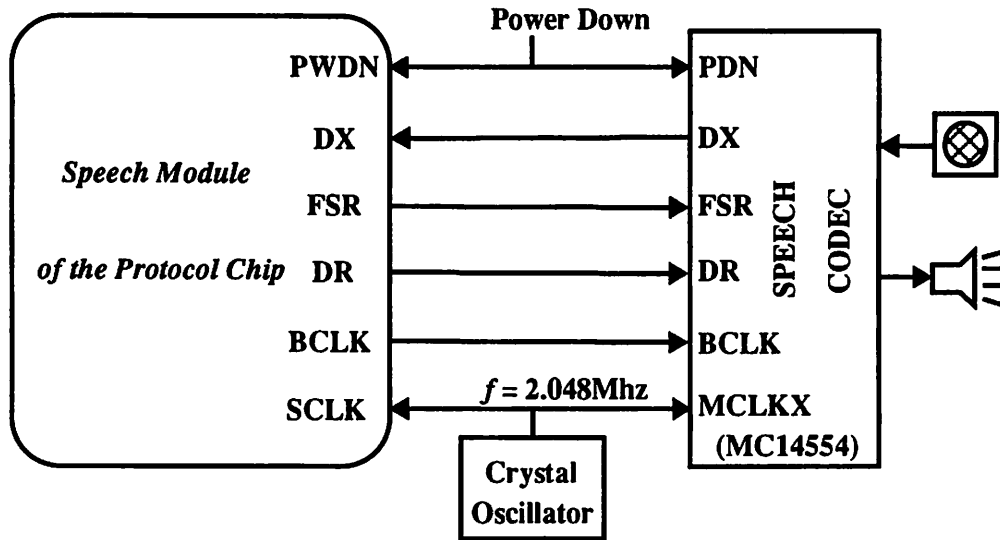


Figure 6-28 : Interface signals between the speech module and the commercial codec.

Figure 6-29 shows the timing relationship between the speech clock and the speech data signals.

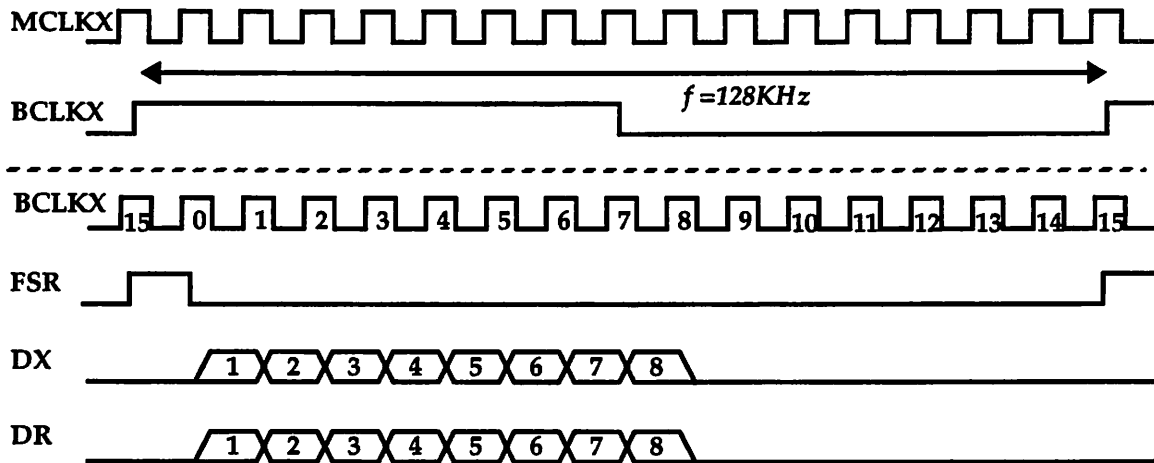


Figure 6-29 : Timing relationship between the clock and the data signals.

On the downlink, speech data from the radio modem is stored in 32-bit speech output FIFO. If data on the FIFO is empty (for example if the base station for some reason could not meet the throughput constraint) then a 0xff is sent to the speech codec till new data arrives over the link.

The speech controller reads this parallel data and transmits it in a serial format required by the speech codec at a throughput rate of 64Kbs. Since DR is sent out in the first 8 cycles, LD_REG is high for 8-cycles and the data is scanned out. During the rest of the cycles, DSCAN is low and the data in the 32 register is held using the feedback from the output of the register to the multiplexor input.

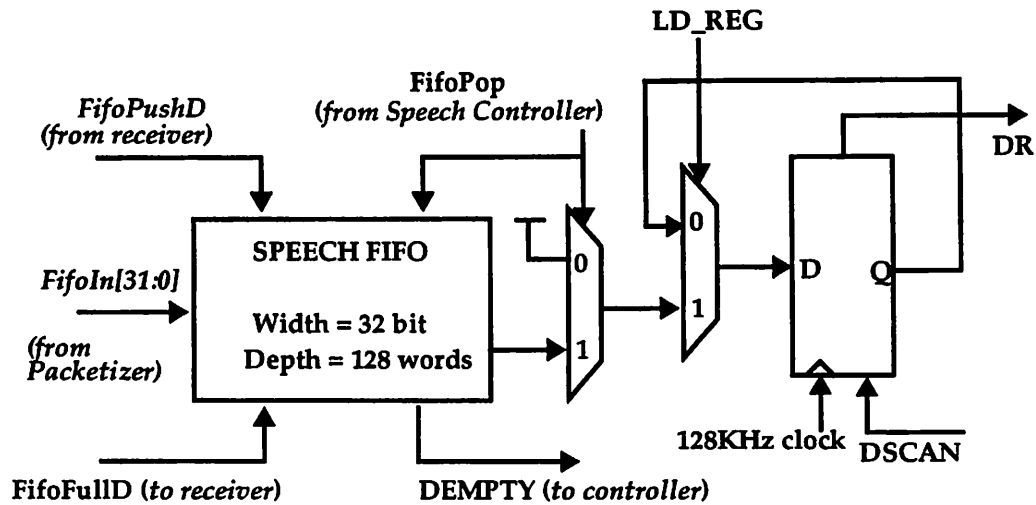


Figure 6-30 : Speech downlink datapath.

On the uplink, serial data from the codec at a rate of 64Kbs is converted to a parallel format and pushed onto the uplink speech FIFO. The controller monitors the “fullness” of the FIFO and pushes data only if the FIFO is not full. If the FIFO is full, the new incoming samples are discarded until the FIFO is no longer full. The packet size on the uplink is programmable and is set in the radio transmitter module.

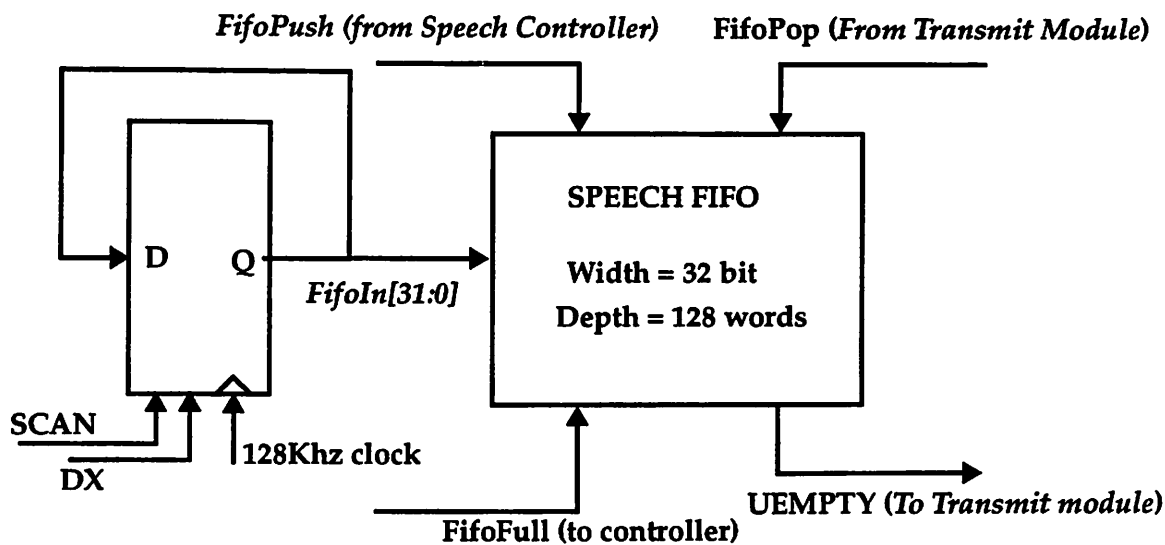


Figure 6-31 : Speech uplink datapath.

6.4.4 Pen Input Module [Narayanaswamy93]

The pen data from the pen digitizer uses an interface similar to the speech data. The pen module takes pen data (co-ordinate and control information) from the output of a commercial pen digitizer, converts it to parallel format and sends it to the radio transmitter module through a 32-bit FIFO.

6.4.5 Radio Transmitter Module

The transmitter module takes the 32-bit data from the pen and speech, packetizes it and converts it to the serial format required by the radio modem. The packet sizes for the speech and pen data can be set to an arbitrary value through external I/O pins. The controller monitors the state of the speech input FIFO and the pen input FIFO and initiates a packet transmission to the uplink radio transmitter when enough data is buffered in either FIFOs (which is determined by the length). The FIFOs only generate **FULL** and **EMPTY** status signals and therefore it was necessary to use an **UP/DOWN** counter to keep track of the “fullness” in the FIFO. The requests for pen data transmission and speech data transmission are arbitred and queued. The serial packet transmitted

on the uplink is similar to the one shown in Figure 6-11. The transmitter module does a 32-bit comparison between the synch pattern and the data for every single bit shift to see if the data contains the synch pattern. If the data does contain a synch pattern, then a bit is flipped.

6.4.6 Low-Power Circuits Common to all Modules

Reduced Swing FIFO

Reducing the power supply voltage is clearly a very effective way to reduce the energy per operation since it has a quadratic impact on the power consumption. At a given supply voltage, the output of CMOS logic gates make rail to rail transitions; an approach to reducing the power consumption further is to reduce the swing on the output node. For example, using an NMOS device to pull up the output will limit the swing to $V_{dd}-V_t$, rather than rising all the way to the supply voltage. The power consumed for a $0 \rightarrow V_{dd}-V_t$ transition will be $C_L V_{dd} (V_{dd}-V_t)$ (which was derived in Chapter 2), and therefore the power consumption reduction over a rail to rail scheme will be $\propto V_{dd} / (V_{dd}-V_t)$. This scheme of using an NMOS device to reduce the swing has two important negative consequences: first, the noise margin for output high (NM_H) is reduced by the amount V_T , which can reduce the margin to 0V if the supply voltage is set near the sum of the thresholds. Second, since the output does not rise to the upper rail, a static gate connected to the output can consume static power for a high output voltage (since the PMOS of the next stage will be “on”), increasing the effective energy per transition.

Therefore, to utilize the voltage swing reduction, special gates are needed to restore the noise margin to the signal, and eliminate short-circuit currents. These gates require additional devices that will contribute extra parasitic capacitances. Figure 6-32 shows a simplified schematic of such a gate, used in the FIFO memory cells of the low-power cell library used in this chipset [Burd94]. This circuit uses a precharged scheme that clips the voltage of the bit-line (which has several transistors similar to M5 connected to it) to $V_{dd}-V_t$, where $V_t > V_{t0}$ due to the body effect. The devices M1 and M4 precharge the internal node (the input of the inverter) to V_{dd} , and the bit-line

to $V_{dd} - V_T$. During evaluation ($\phi = "1"$), if V_{in} is high, the bit-line will begin to drop, as shown in the SPICE output next to the schematic. Because the capacitance ratio of the bit-line to the internal node is very large, once the bit-line has dropped roughly 200mV to sufficiently turn on M3, the internal node quickly drops to the potential of the bit-line, providing signal amplification. Thus, this circuit conserves energy greatly by reducing the voltage swing on the high-capacitance bit-line, and reduces delay by providing signal amplification.

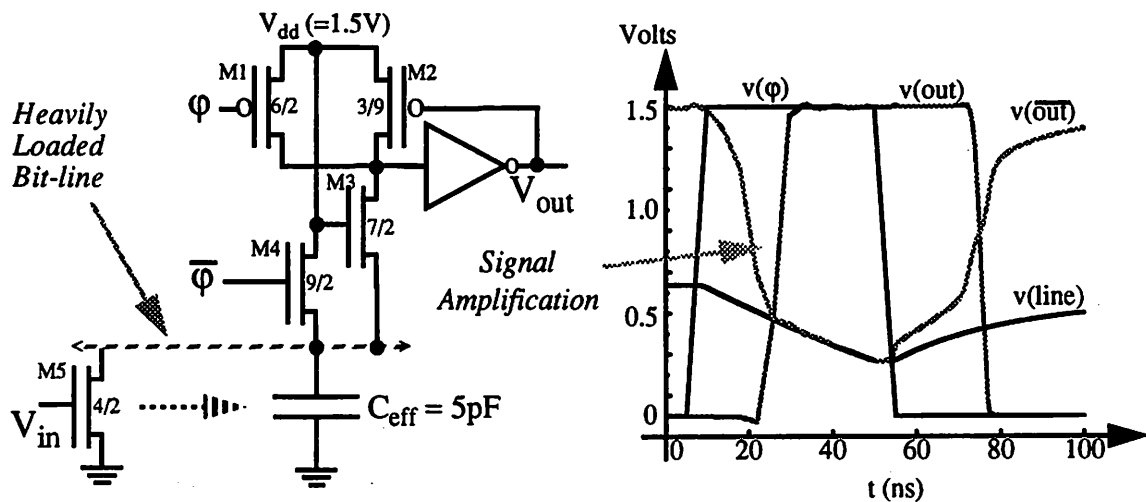


Figure 6-32 : Signal swing reduction for memory circuits: FIFO example.

Level-conversion Circuitry to Interface with I/O Devices

The protocol chip operates at a supply voltage of 1.1V and communicates with I/O devices running at higher voltages - for example the text/graphics module in the chip core has to communicate with the 640x480 LCD display running at 5V or the speech I/O module of the core has to communicate with the speech codec running at 5V. Level-conversion I/O pads - described in Section 3.2.2 (schematic shown in Figure 3-32)- are used to convert the low voltage-swing signals from the core of the low-power chips (running at 1.1V) to the high-voltage signal swings required by I/O devices (e.g. 5V required by displays) or vice-versa. For communication between the protocol chip and the frame-buffer chips, normal low-voltage low-power pads were used. With signal swings of only 1.1V, the I/O power was significantly reduced as well.

Low-power Cell Library

The entire chipset was designed using an extensive cell-library that was optimized for low-power operation [Burd94]. The library contains parameterized datapath cells (adders, registers, shifters, counters, register files, buffers, etc.), multiple strength and multiple input standard cells for random logic (NAND, NOR, registers, buffers, etc.), low-power memories (FIFO, SRAM), low-voltage pads (clock pads, low-voltage I/O pads, level-converting pads, etc.) and low-voltage high efficiency switching regulators [Stratakos94]. Some of the key design approaches for the cell library are outlined below:

- **Device sizing** Minimum sized devices were used in most datapath and standard cells. As analyzed in Chapter 3, minimum sized devices should be used to minimize the physical capacitance for circuitry inside the datapath blocks where self-load, rather than interconnect capacitance dominates. When driving large interconnect capacitances, for example when communicating between datapaths, large sized devices are used.
 - **Reduced swing circuitry:** as described earlier, reduced swing circuitry is used limit the voltage swing in memory circuits (FIFO and SRAM) and reduce the power consumption in a linear fashion.
 - **Self-timed Circuits:** The memory circuits use self-timing to eliminate spurious transitions on high capacitance data busses. This guarantees the minimum number of transitions per memory access.
 - **Efficient Layout:** as described in Chapter 4, for tilable datapath cells, all buses are routed in metal 2, and metal 2 is not used for routing within a cell. Unfortunately, this forces some of the modules, such as the adder, to use poly for intracell routing, which has two to three times the capacitance per unit area. However, this scheme reduces the amount of cell stretching inside each datapath (for routing channels) which reduces the overall datapath area. This in turn results in smaller global bus lengths and therefore lower global interconnect capacitance. That is, the inter-block bus routing becomes much more efficient, and the capacitance increase from using poly is small compared to the overall decrease attributable due to smaller global buses.
-

-
- **Integrated control slice:** the bit-sliced datapaths have integrated control slices which perform various functions. This reduces or sometimes eliminates the amount of standard-cell control that is used. This reduces power since devices in the control slice can be optimized for the load that has to be driven. For standard cell control, the load is the load for the cells in the bit-slice plus the extra interconnect capacitance.

Clocking strategy

Highly pipelined systems have a significant portion of the total power consumed by the clock. Therefore, it is important to minimize the number of global clock nets (one approach is to use gated clocks), as well as all the gate capacitance switching at the clock rate. To accomplish this, the TSPC (True Single Phase Clocking) methodology was used in the design of all register elements, with the basic register shown in Figure 6-33. The two main advantages of this style are that only one global clock net is needed and there is only a total of four minimum sized transistors attached to the clock net per register. This provides almost a 50% reduction in clock power over non-overlapping two-phase latches. The library also contains a dual-edge triggered register which allows the clock frequency to be halved for the same throughput as a normal register. This once again reduces the power consumed in the clock routing.

There are four key design considerations when using the TSPC register: First, the rise-time of the clock signal is critical to proper operation and therefore a control slice attached to each datapath register provides local clock buffering. Second, it is a dynamic register, such that the clock has a minimum operating frequency. Measurements have shown the minimum to be on the order of 500 Hz, which satisfies the constraints of most applications. A modified version of the register exists with two extra transistors, that provide static feedback while the clock is low. Thus, the register can be used to gated-clock configurations, without signal leakage. Third, there is partial internal glitching due to a momentary race condition. This was not found to be a problem since the glitch is small and the output does not switch. Finally, the internal node INT will have transitions if the input is a constant LOW since INT will be discharged during evaluation and precharged when the

clock is low. This is typically not significant for data which is switching around frequently. If the TSPC is used as a register for control signals that are LOW most of the time, then the complement of the signal should be sent through the register and a static inverter should be used at the output to achieve correct polarity.

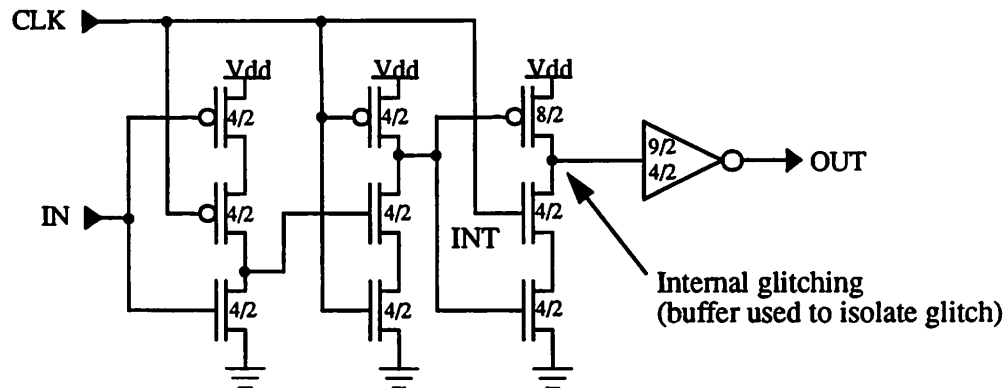


Figure 6-33 : True single phase clock pipeline register.

6.4.7 Chip Layout and Statistics

The protocol chip was designed using the LagerIV silicon compiler. Figure 6-34 shows the layout of the protocol chip. Table 6-2 shows the statistics of the protocol chip. The chip operates all the way down to a supply voltage of 1.1V and consumes only 1.9mW at 1.5V.

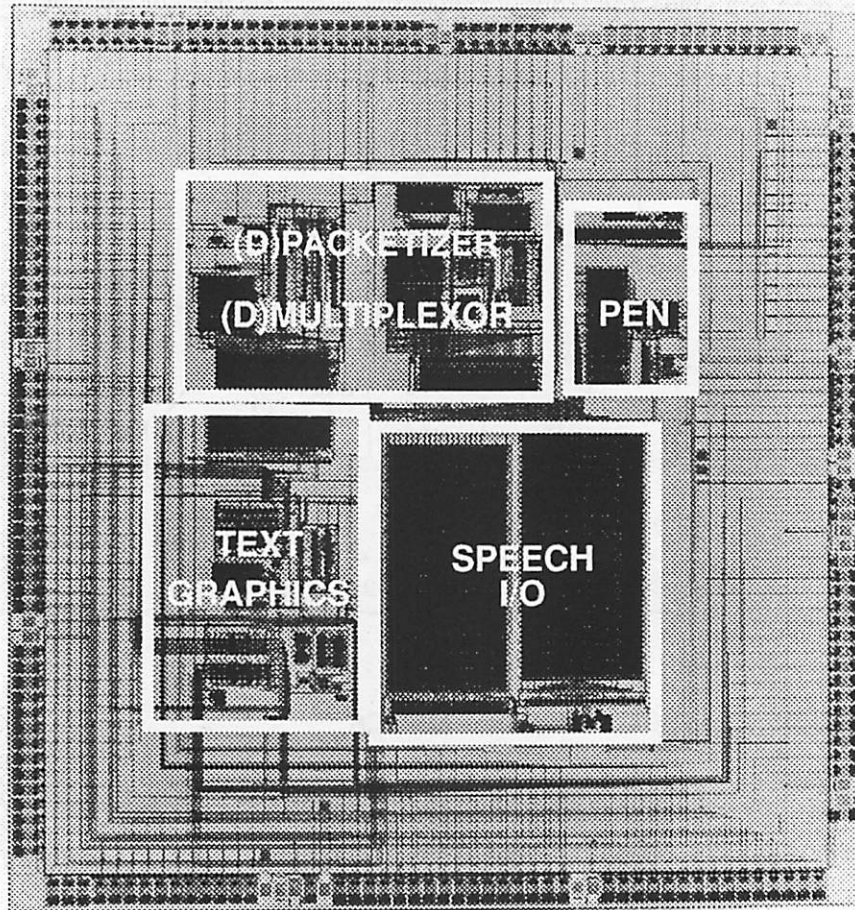


Figure 6-34 : Die photo of the protocol chip.

Table 6-2 : Statistics for the protocol chip.

Parameter	Value
Technology	1.2 μ m CMOS
Size	9.4mm x 9.1mm
Number of Transistors	136000
Minimum Operating Voltage	1.1V
Power Consumption at 1.5V	1.9mW

Table 6-2 : Statistics for the protocol chip.

Parameter	Value
Throughput	1Mbs serial for radio 3MHz for 4-bit text/graphics 4Kbs for pen input 64Kbs for speech I/O

Table 6-3 shows a summary of the various power reduction techniques that were applied to the protocol chip and the amount of power reduction that is achieved using each technique.

Table 6-3 : Summary of power reduction techniques applied to the protocol chip.

Approach	Power Reduction	Comments
Supply Voltage Scaling (Parallelism and Pipelining)	21	1.1V operation vs. 5V operation
Optimized Cell Library	x3-4	Transistor sizing, Clocking, Reduced swing FIFO, Self-timed Glitch free FIFO, Layout optimization, Low-power Pads
Gated Clocks	x2-3	Gated clocks reduce both the clock load and power in logic
Activity Driven Place and Route	x1.5-2	This reduction is just for the global busses. Standard placement tools were used for the logic generation

6.5 Text/Graphics Frame-buffer Module [Burstein93]

The frame-buffer for the split panel text/graphics display contains six 64Kbit SRAM chips (Figure 6-35), which were synthesized from the low power cell library's tileable SRAM module. This module meets several constraints to make it a useful component of a wide variety of low power

systems.

First, the SRAM module builds memories over a wide variety of sizes. At one extreme, it can create entire memory chips that are only limited by maximum die size, such as the frame-buffer memories used for the text/graphics display. At the other extreme, the same module synthesizes smaller (hundreds or thousands of bit), area and power-efficient memories that are placed on the same chip as datapath and control systems (as in the video decompression chips described in the next chapter). The designer can specify both the number of words and the bit width of the words. The designer specifies the number of words, which may be any size over two, and the number of bits per word, which may be from two to sixty four. The designer can also control the aspect ratio by specifying the number of blocks at the architecture level, as detailed below.

Next, the SRAM must match the timing and electrical constraints of the rest of the library. Since

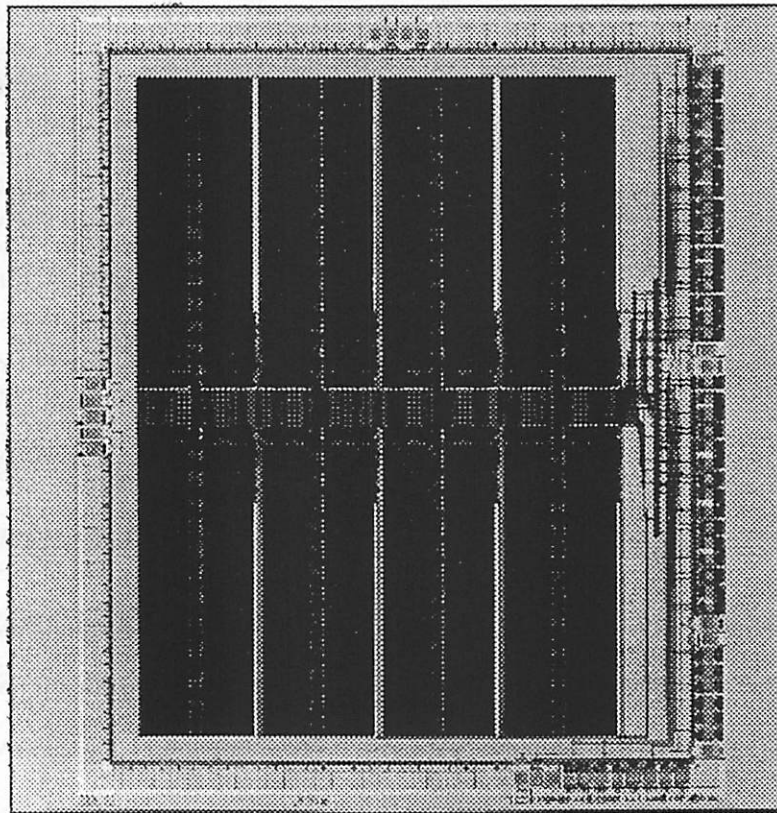


Figure 6-35 : Die photo of the text/graphics frame-buffer.

our low power strategy stresses running highly parallel and pipelined systems operating at low clock rates and lower voltages, the SRAM only needed a cycle time of 100ns - 200ns at 1.5 volts to be useful in our low power systems; there is no advantage to designing faster components that consumed more power, since our systems have fixed data throughputs. Another major principle of low power operation is to use a low supply voltage: 1.5 volts is the target supply for the low power library. Although it is optimized to meet its speed requirement at 1.5 volts, the SRAM functions over a wide range of supply voltages. The text/graphics frame buffer chips have been used with supplies as low as 1.1 volts by exploiting parallelism. The SRAM also can operate with supplies up to 5V, with increases in both speed and power consumption.

The final requirement is to make the timing and control of the SRAM as simple and general purpose as possible, so that system designers will not have to be concerned with clocks internal to the SRAM. It triggers off of the rising edge of a single clock; all other timing signals are generated internal to the SRAM, using self-timing circuitry that scales with the size of the SRAM. Also, the SRAM can use either a single, bi-directional bus or can use separate input and output buses.

6.5.1 Memory Architecture

The architecture of the SRAM is designed both to minimize power consumption and to enhance scalability. At the highest level of the architecture, the SRAM is organized into a parameterizable number of independent, self-contained blocks, each of which reads and writes the full bit width of the entire memory. For example, the 64 Kbit (2k word by 32 bit) frame buffer chips contain eight 8 Kbit blocks, each organized as 256 words by 32 bits. Since one of the fundamental power saving techniques is to minimize the effective capacitance switching per clock cycle, the SRAM only activates circuits in one of these blocks per clock cycle (Figure 6-36). The power savings from this architecture are twofold. First, there is less overhead capacitance, since only one set of control signals and decoders switch at one time. Second, and more fundamentally, by having wide data widths into each block, the memory has minimal column decoding, and hence less column

capacitance to switch per bit. From an ideal power consumption perspective, it would be best to have no column decoding at all; as a practical concession to pitch matching, this design uses 2 to 1 column decoding.

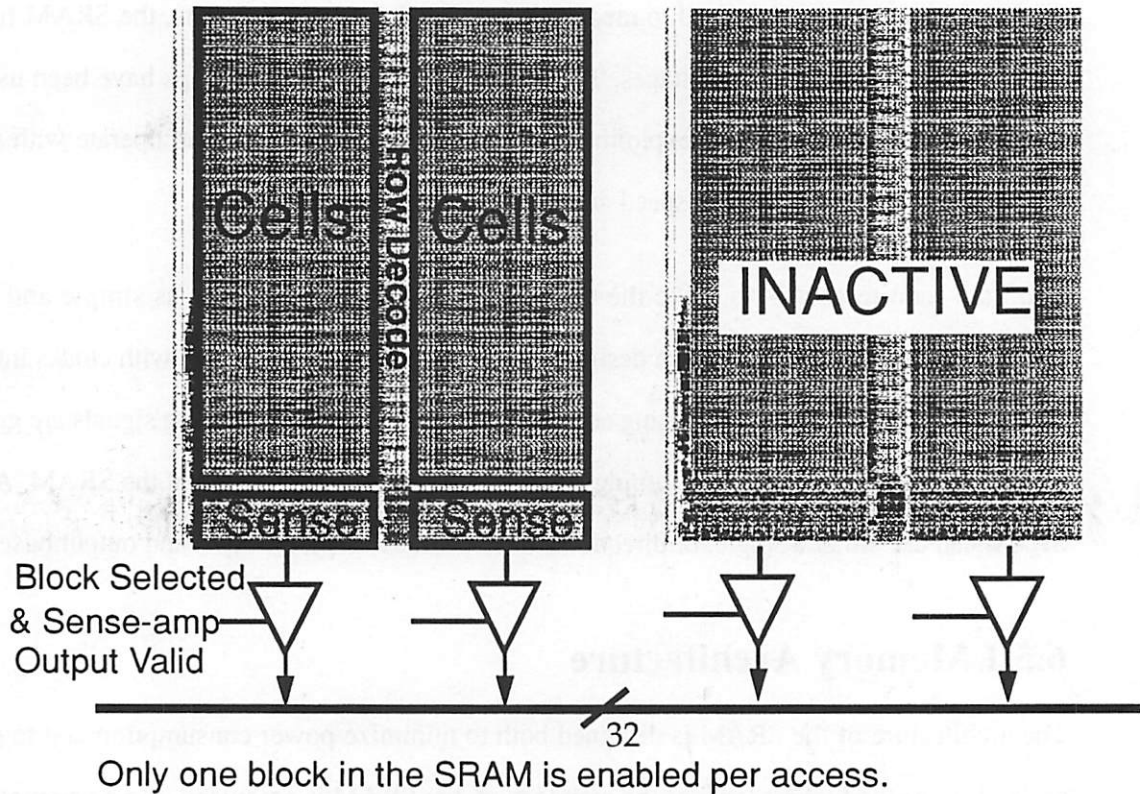


Figure 6-36 : SRAM block organization.

The trade-off for this savings in power is a loss in speed. Since only one block is activated per clock cycle, each block has a decoder (composed of static logic) that must activate its block before that block can begin any operations. Thus, the delay through these block level decoders is added to the total SRAM delay. However, this loss of speed causes no penalty as long the SRAM meets the overall system throughput needs.

Using self-contained blocks also makes the architecture flexible and expandable. Since the number

of words per block and the total number of blocks are parameterized, the designer can control the aspect ratio by trading off between the number and size of the blocks. Adding blocks to the memory increases its number of words, with a minimal effects on power consumption and circuit design. Since circuits in only one block switch at a time (except for the block level decoders, which consume minimal power) the only increase in power consumption is from the increased wiring capacitance between blocks. The only circuits that need to change are the block level decoders. All of the other control, timing, decoding, and sensing circuitry in the blocks are insensitive to the number of blocks in the memory.

6.5.2 Circuit Level Optimization

At the circuit level, the SRAM's most important power saving technique is to reduce the voltage swing on the bitlines. As shown in (Figure 6-37), the bitlines are precharged through NMOS devices, so the maximum bitline voltage is the supply voltage minus the NMOS threshold voltage (with body effect). Compared to full swing bitlines, this reduces power consumption by only 20% for $V_{dd}=5V$, but as much as 50% for $V_{dd}=1.5V$ and 75% for $V_{dd}=1.2V$. (It was not practical to limit the minimum voltage on the bitlines by timing the wordline signals because the timing would have to work for many sized memories, over many voltages, and for different fabrication technologies.) This precharge strategy also allows the column select transistors to double as cascode amplifiers, creating voltage gain between the bitlines and the input to senseamp.

Another important power saving technique is to eliminate glitching on the data bus during read operations. The output driver of the senseamp shown in (Figure 6-37) is tristated at the beginning of each clock cycle. Even after the output enable signal is true, the output remains tristated until the senseamp's cross coupled latch has resolved the data value, at which time either the NMOS or the PMOS output device drives the data bus.

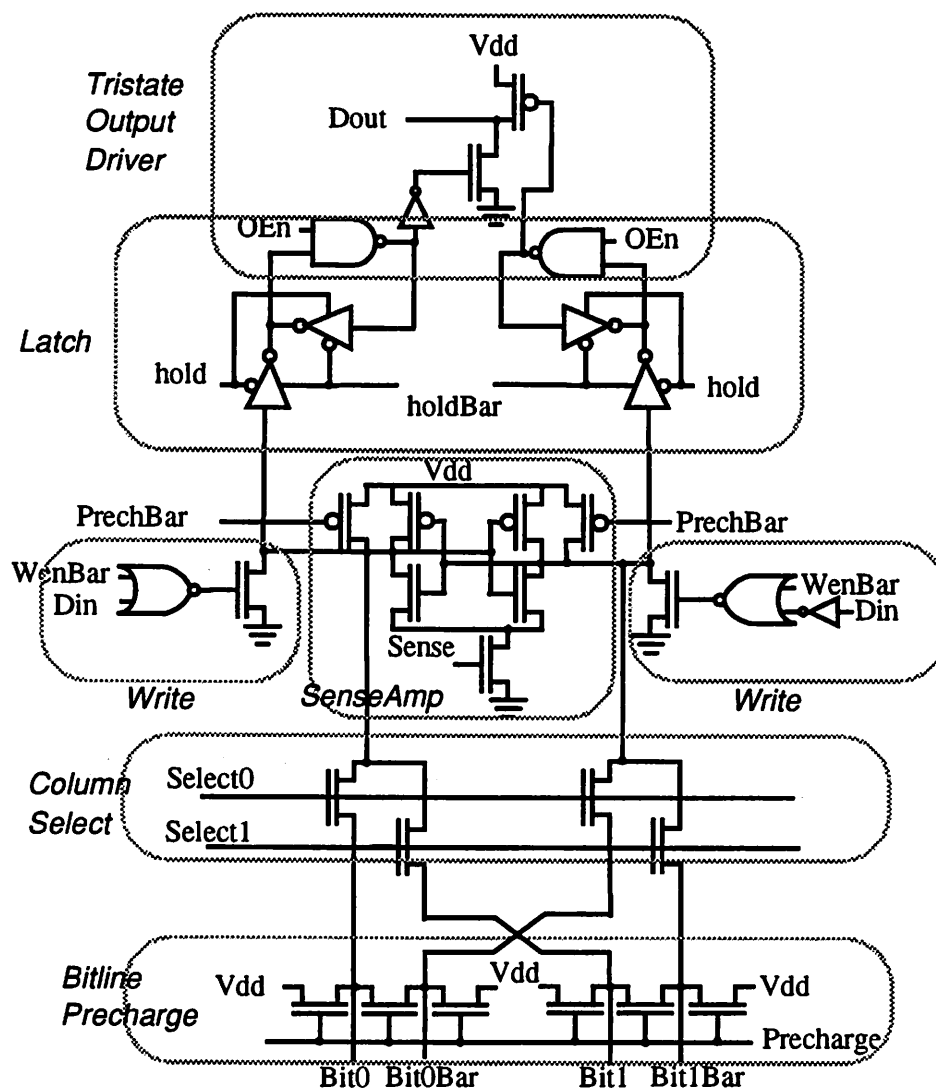


Figure 6-37 : Precharge, sense-amp and “Glitch” free tri-state buffer.

One of the major handicaps of using a high threshold process ($V_{tp}=-0.9V$, $V_{tn}=0.7V$) at low supply voltages is that the low gate overdrive voltage for the PMOS ($V_{gs} - V_t$) exacerbates their lower carrier mobility, creating a large imbalance between the current drive of the PMOS compared to the NMOS devices. A partial remedy is to make sure that large capacitances are never driven by more than one PMOS device in series. The senseamp output driver meets this requirement, in addition to eliminating glitches. Another partial remedy is to use a large $W_p:W_n$ ratio --

approximately 6 -- in static inverters that are in the critical path. Ideally, it would be better to have NMOS and PMOS threshold equal in value, and as low as possible. Ultimately, the minimum threshold voltage would be set by the maximum acceptable power consumption from leakage current. However, since most of the leakage occurs in the memory cells, it would be advantageous to have a dual threshold process, using the high V_t transistors in the memory cells and the low V_t transistors elsewhere.

6.5.3 Chip Statistics

Table 6-4 shows a summary of the frame-buffer chip used to store text/graphics pixel data.

Table 6-4 : Statistics for the text/graphics frame-buffer chip.

Parameter	Value
Technology	1.2 μ m CMOS
Size	7.8mm x 6.5mm
Number of Transistors	428,800
Memory Size	64Kbits
Word Width	32bits
Minimum Operating Voltage	1.1V
Power Consumption at 1.5V (including loading of two other chips)	500 μ W
Throughput	375kHz read (write and read cycles are interleaved)

Table 6-5 shows a summary of the various power reduction techniques that were applied to the frame-buffer chip and the amount of power reduction that is achieved using each technique.

Table 6-5 : Summary of power reduction techniques applied to the frame-buffer chip.

Design Approach	Power Reduction	Comments
Supply Voltage Scaling	21	1.1V operation vs. 5V operation
Self-timed Outputs Drivers	x2	Assuming that the words are temporally uncorrelated and the output capacitance dominates the global bus capacitance - reasonable since it drives off chip interconnect.
Bit Swing Reduction	x3.7	This reduction is only for the memory array. The global bus has rail to rail swing. Comparison is for 1.1V swing vs. 300mV swing
Block Decoding	x8	This reduction is just for the memory array. This is the power reduction due to enabling only one block vs. enabling all blocks in the SRAM - as done in high speed applications

6.6 System Implementation

A PCB board containing the protocol chip, 6 SRAM chips, a speech codec, and pen interface logic has been fabricated and tested. Figure 6-38 shows a block diagram of the first generation InfoPad terminal called the IPGraphics. Various power supply voltages needed for the design including -17V (adjustable using a trim-pot) for display drive, 12V for DC to AC inverter for the backlight, 1.1 V for the custom chips, and -5V for the speech codec have been realized using commercial chips from Maxim.

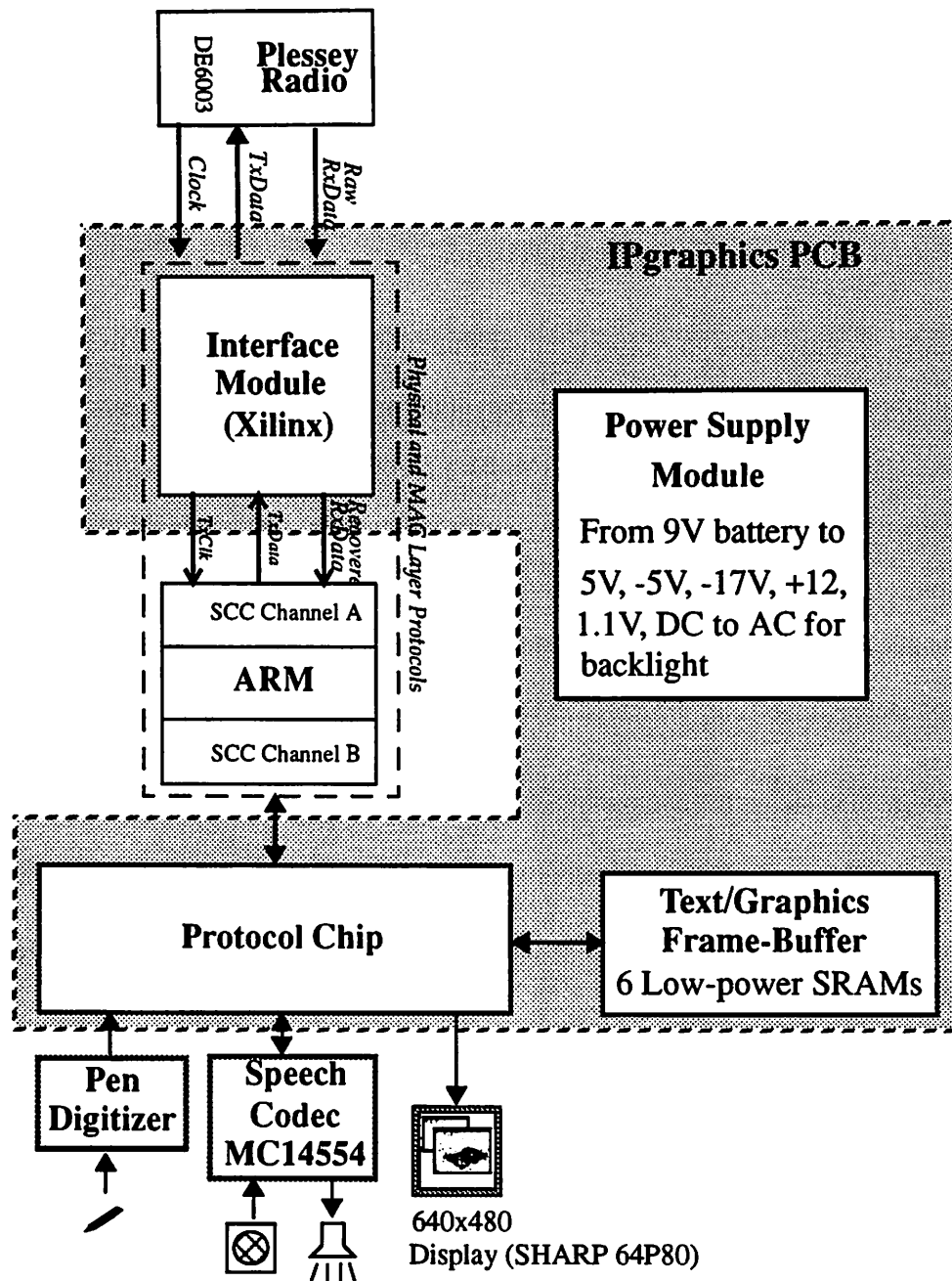


Figure 6-38 : Block Diagram of IPGraphics terminal.

This board is integrated with a commercial radio modem, a half-duplex, 1 Mbps RF modem: the DE6003, from GEC Plessey, to realize a complete I/O terminal with a 1 Mbit/sec wireless channel. The DE6003 provides a demodulated signal to the baseband receiver, implemented in a Xilinx 4000 FPGA, without any explicit timing information or attached protocol. This “raw” received

data is passed to the clock recovery module (the “Interface Module”), which extracts timing information from transitions in the data stream. Once it has attained bit-level synchronization -- achieved by sending a training sequence of alternating 1’s and 0’s at the beginning of a packet -- the interface board attempts to reconstruct the received bits from the raw bitstream, and passes the recovered clock and data to the ARM board.

Early on in the design of the wireless link, the decision was made to use a general-purpose microprocessor or microcontroller as part of the protocol support hardware, providing flexibility in the design of the protocol. The ARM610, a 32-bit RISC device, was the microprocessor chosen because it offered a relatively low-power solution for our needs. Media access control, flow control, and tracking error statistics for link management are the most important duties performed by the ARM, while physical layer functionality, such as clock and data recovery, line coding, and channel sensing, is implemented in the Xilinx FPGA.

Figure 6-39 shows the photograph of the first generation InfoPad terminal.

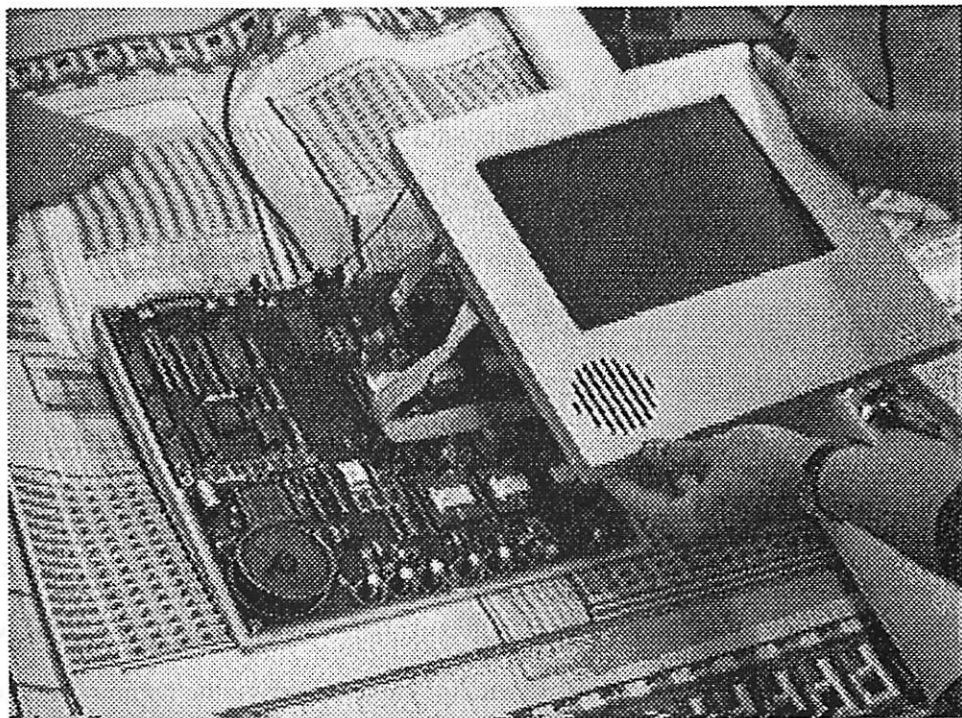


Figure 6-39 : Overview of IPGraphics (Pen, Speech I/O and Text/graphics) Terminal.

6.7 Summary

Recently, the concept of “personal communications” has come to the forefront of communications research, in which individual users will have portable, private access to fixed computing facilities. The ultimate goal is to provide a personal communications system, which will move information of all kinds to and from people in all locations, through an advanced wireless network supporting a wide range of services. As a step beyond today’s portable computers, a high-speed wireless link allows the advent of small, lightweight, multimedia graphics terminals, whose primary function would be to connect the user instantaneously and transparently into powerful fixed processing units and data storage. It would be capable of providing simplified user interface through pen input and speech I/O, data transfer and retrieval, computing services, and high-quality, full-motion video. The design of a low-power chipset has been described for a portable multimedia terminal

which supports pen input, speech I/O, and text/graphics output.

A system approach to power reduction was used which involves optimizing the physical design, the circuitry, the logic design, the architectures and system partitioning. At the system level, the power consumption is reduced by optimizing the partitioning of computation between the terminal and the backbone network. By exploiting wireless communications capability, all general-purpose user-programmable computation is *removed* from the terminal and performed by compute servers on the backbone network. This significantly reduces the power requirements of the terminal. At the architecture level, the biggest power savings came from using the architecture driven voltage scaling strategy which involves parallelism and pipelining and allowed supply voltages as low as 1.1V. At the logic level, gated clocks were used extensively to minimize the effective clock load and the switched capacitance in logic circuits. Various circuit level optimizations were performed to minimize the voltage swings and switched capacitance. In the memory circuits (SRAM and FIFO), reduced swing logic (using NMOS precharge) was used to limit the swing on the bit lines to 300mV at a supply voltage of 1.1V. The memory modules also used self-timed circuits to eliminate glitching on the data busses, hence minimizing the effective capacitance switched. The various logic modules were implemented using a low-power cell-library that featured minimum sized devices, optimized layout, and single-phase clocking. At the layout level, an activity driven place and route was used to route high activity nets using short wires while allowing low activity nets to have long wires. The various low-power techniques results in the power consumption of the protocol chip and text/graphics frame-buffer to be less than 3mW at 1.5V.

CHAPTER 7

A Low-power Video Decompression Chipset

This chapter describes the implementation of a low-power chipset that takes a serial compressed video stream from the radio and converts it to the analog format required by a 4" color LCD display. The implementation of this module will illustrate several low-power techniques that can be applied at all level of the system design ranging from the circuits and logic used to the architectures and algorithms.

7.1 Algorithm Selection for Video Decompression

The choice of algorithm is the most highly leveraged decision in meeting the power constraints. The ability for an algorithm to be parallelized is critical and the basic complexity of the computation must be highly optimized. Minimizing the number of operations to perform a given function is critical to minimizing the overall switching activity and therefore the power consumption. The task of selecting the algorithm for the portable terminal depends not only on the traditional criteria of achievable compression ratio and the quality of reconstructed images, but also on computational complexity (and hence power), and robustness to high bit error rates.

7.1.1 Computational Complexity Trade-off

Most compression standards (for example, JPEG and MPEG) are based upon the Discrete Cosine Transform (DCT). The basic idea in intra-frame schemes such as JPEG is to apply a two-dimensional DCT on a blocked image (typically 8x8) followed by quantization to remove correlations within a given frame. In the transform domain, most of the image energy is packed into only a few of the coefficients, and compression is achieved by transmitting only a carefully chosen subset of the coefficients. One main characteristic of the DCT is the symmetric nature of the computation; i.e., the coder and decoder have equal computational complexity.

Although the computational complexity of the DCT can be optimized by restructuring algorithms, it still requires several arithmetic and memory operations per pixel. Table 7-1 shows a comparison of a few algorithms that can be used to implement the DCT [Clarke85][Feig90][Rao90]. Minimizing the operation count is important in minimizing the switching events and hence the power consumption.

DCT Algorithm	Multiplies (8x8)	Additions (8x8)
Brute Force	4096	4096
Row-Col. DCT	1024	1024
Chen's Algorithm	256	416
Lee's Algorithm	192	464
Feig's Algorithm	54	462

Table 7-1 : Computational complexity of the Discrete Cosine Transform.

An alternative compression scheme is vector quantization (VQ) coding (as described in Chapter 4), which is asymmetrical in nature and has been unpopular due to its complex coder requirements. Figure 7-1 once again shows the basic idea behind intra-frame vector quantization compression. On the encoder, a group of pixels is blocked into a vector and compared (using some metric such

as Mean Square Error) against a set of predetermined reproduction vectors (a set of possible pixel patterns) and the index of the best match is output from the encoder. The decoder has a copy of all possible reproduction vectors (codebook) and the index of the best codeword is used to reconstruct the image using a simple lookup table operation.

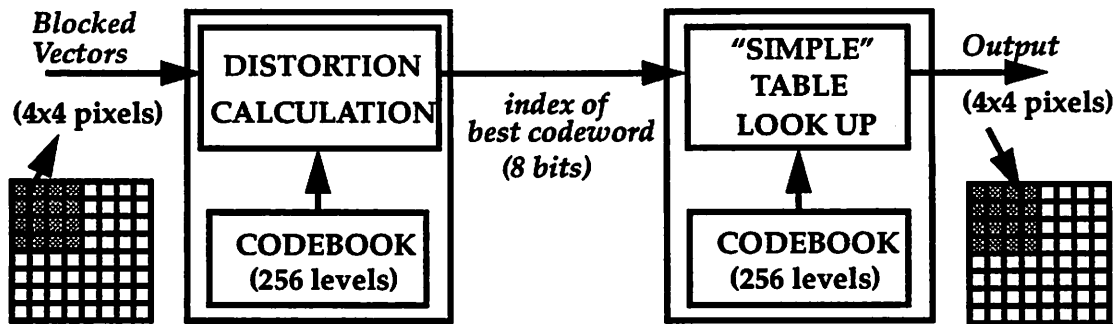


Figure 7-1 : Video compression/decompression using vector quantization.

Vector quantization has been unpopular due to its complex coder requirements. However, since VQ algorithms require a very simple memory-lookup decoder they are well suited for single-encoder, multiple-decoder systems. If one-way video communication is desired, the VQ solution provides a means of implementing real-time decompression using very little computation and power; while a DCT based decompression system based may require many operations per pixel, a VQ based decoder will only require a single memory lookup operation [Gersho90]. Table 7-2 shows a comparison a typical DCT and VQ algorithm. In addition to minimizing the number of operations in the decoder, the power consumption can be further reduced by optimizing the memory structure and organization for reduced voltage operation and minimal switched capacitance.

Method	Computation Requirements	
	coder	decoder
Transform (DCT) (Row - Col. fast algo) [Rao90]	4 multiply 8 additions 6 mem. access	4 multiply 8 additions 6 mem. access
Vector Quantization (Tree search - differential CB) [Fang90]	8 multiply 8 additions 8 mem. access	1 mem. access

Table 7-2 : Asymmetrical algorithm required for low-power implementation.

7.1.2 Robustness to Channel Errors

In addition to the simplicity of the decoder, intra-frame VQ has two other advantages with respect to error resilience. First, VQ localizes errors in space; i.e, errors in the VQ codeword data appear as small corrupted 4x4 blocks on the screen and don't corrupt large portions of the screen. Simulations, as shown in Figure 7-2, indicate that even with Bit Error Rates as high as 10^{-3} , the image looks reasonable and therefore no error correction was applied on the codebook data, significantly enhancing the bandwidth efficiency of the wireless protocol. The codebook used for the simulation did not involve any optimization. The entries in the codebook can be rearranged so that bit errors cause little subjective degradation [Zeger90]; the codebook is arranged so that similar-looking vectors have similar addresses. Run-length coding, as used in JPEG, is not suitable for wireless operation since errors can cause loss of synchronization and can corrupt large portions of the image. Second, intra-frame compression localizes errors in time: errors don't accumulate from frame to frame, unlike differential schemes such as MPEG. In differential schemes, errors stay on the screen until a full new image is sent (in an intra frame mode). This chipset does support inter-frame mode, in which only portions of the screen that are changed are updated. However, this mode can be used only when the bit error rates are low.

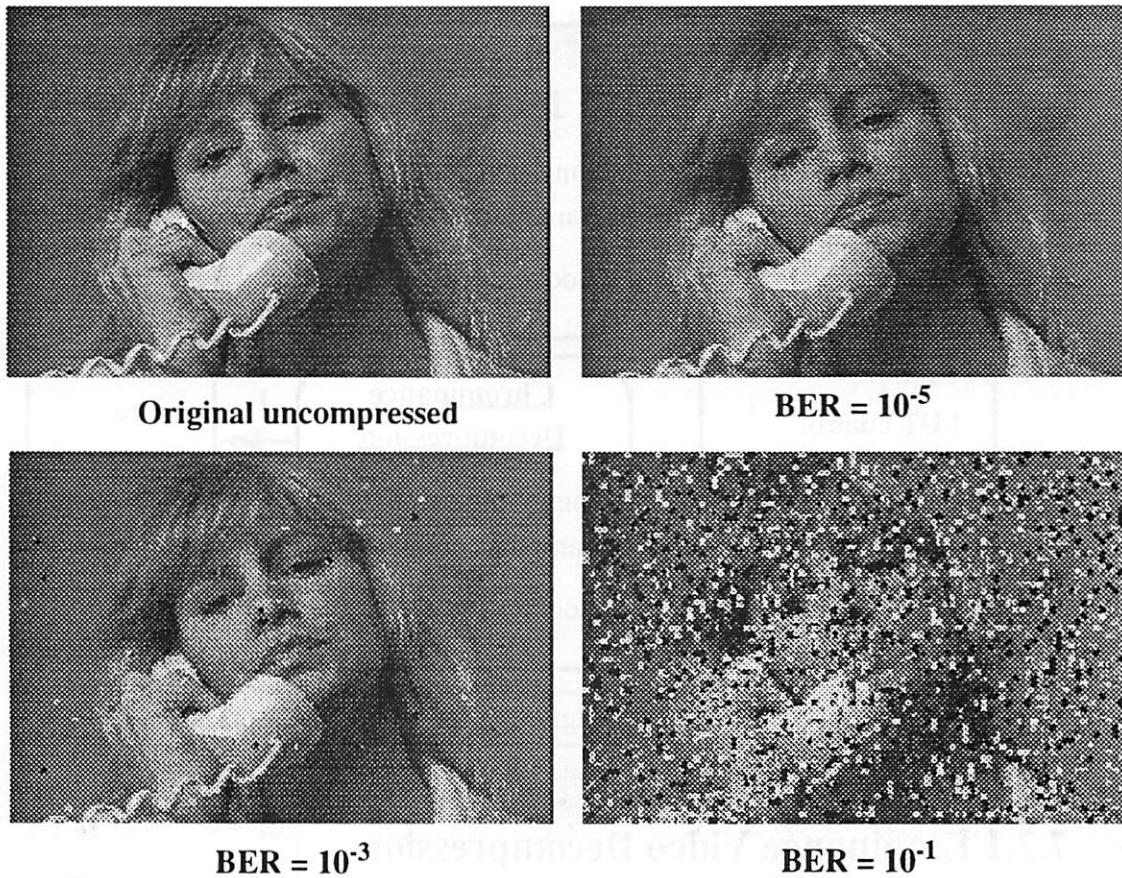


Figure 7-2 : Effect of channel errors on vector quantized images (16:1).

7.2 Video Decompression Module Implementation

The video decompression module includes all of the circuitry required to take a compressed video stream from the radio and convert it to the analog data format required by a 4" (128x240 pixels) color active matrix display. This section presents the design of a set of 4 chips (as shown on the block diagram in Figure 7-3) to perform video decompression for this low-resolution display; one chip performs all of the control functions for the video decompression and interface to the radio, two chips perform the frame-buffering and the video decompression based on the vector quantization table look-up algorithm, and one chip performs the color space conversion and

analog interface to the display.

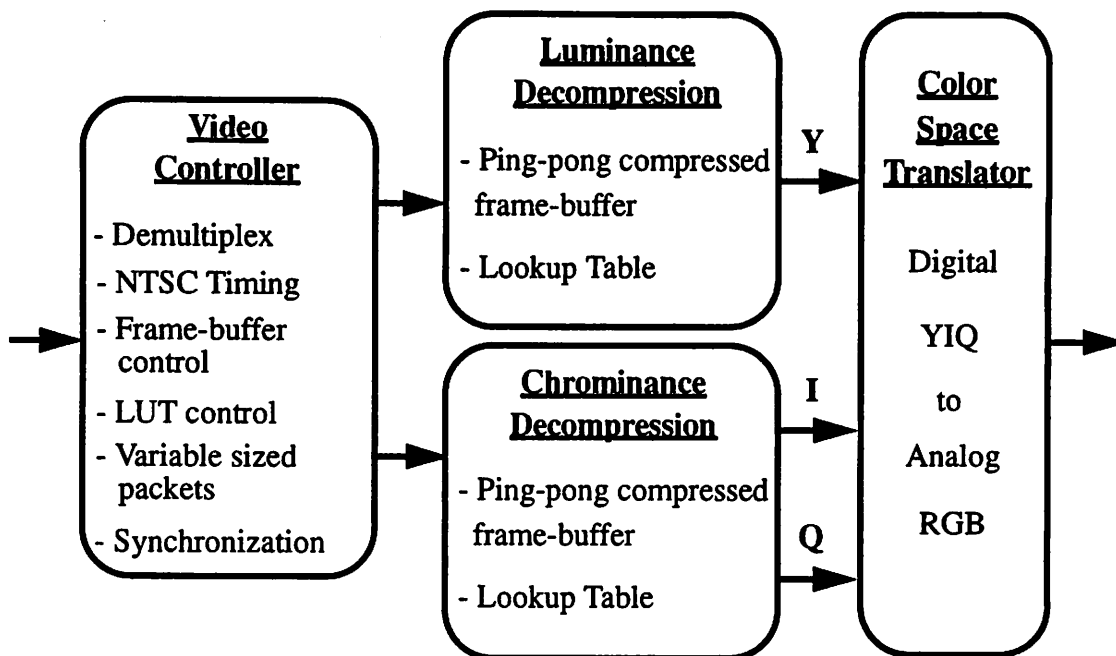


Figure 7-3 : Block diagram of the video decompression.

7.2.1 Luminance Video Decompression

For this implementation, the image is segmented into 4x4 blocks (i.e the vector size is 16) and there are 256 entries in the codebook. The original image on the encoder side is represented in the RGB domain using 6-bits for each color plane - using 6-bits to represent video data instead of 8-bits results in very little visual distortion on this low-resolution display. Color information is transformed to the YIQ domain and each plane is individually coded with separate codebooks. The I and Q color components (called the chrominance components) are sub-sampled in both the horizontal and vertical dimensions. Therefore, the YIQ representation (Y:1, I:1/4, Q:1/4) gives 2:1 compression over the RGB representation (R:1, G:1, B:1). On each plane, VQ results in 12:1 compression since only 8-bits are transmitted for each 4x4 block (choosing 1 out of 256 codes) instead of 16 x 6 bits for the true data. A total of 24:1 compression is achieved (32:1 if the quantization from 8bit to 6bit representation is taken into account) and therefore to support full-

motion color video at 30 frames/sec, this system consumes a bandwidth of 690 Kbits/sec on the wireless link.

Figure 7-4 shows a block diagram of the luminance decompression chip. The incoming compressed video data (VQ codewords for the image) is stored using a ping-pong addressing scheme. For example, when the compressed video coming from the RF link is being stored in BANK0, the compressed data stored in BANK1 is read out to be decompressed using the lookup table (LUT). After a full frame of compressed video is assembled in BANK0, the R/\overline{W} signal (signal *WBANK*) of the two frame-buffers toggles, resulting in data being read from BANK0 to the lookup table while new compressed video data is written into BANK1. This ping-pong addressing scheme provides an asynchronous interface between the radio clock and the video system and provides immunity against bursty channel errors; if a frame of compressed video is dropped or if higher priority data is sent over the link (such as text/graphics data or speech which require data to be sent with minimal latency), the complete compressed frame that is already stored in the terminal is decompressed and displayed until a new frame of compressed video is assembled in the other frame-buffer. Also, since the refresh rate of the color display is 60Hz while the image is updated only at 30Hz, the ping-pong frame-buffer is actually required unless the bandwidth on the link is doubled.

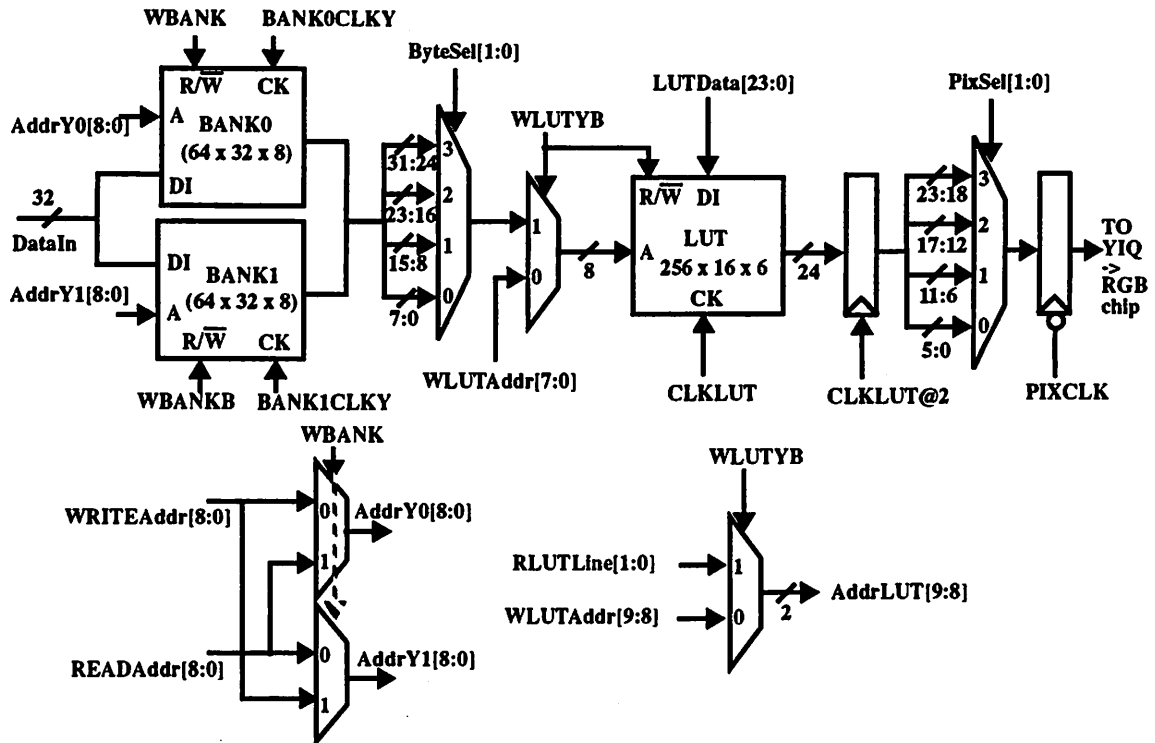


Figure 7-4 : Block diagram of the luminance decompression chip.

The video is stored in a compressed rather than uncompressed format to reduce the amount of memory in the system by a factor of 24. Rather than decompressing and storing the data once and displaying the decompressed image twice (since as mentioned above, the refresh rate is 60Hz while the image is updated at 30Hz), the image is stored in a compressed format and is read out and decompressed twice; that is, there is no decompressed frame-buffer. The compressed frame-buffer is read out in a parallel fashion, where four 8-bit VQ codewords are read in parallel (once again the access pattern of data is known and is exploited) though only one is used at a given time; this once again enabled operation at supply voltage as low as 1.1V. In this implementation, the video frame-buffer was clocked at 156kHz while meeting the throughput rate of 2.5MHz. The output of the frame-buffer is multiplexed at 4:1, and is used to index the lookup table which generates the decompressed data.

The chip uses an addressing scheme that eliminates the need for an output line buffer which is typically used to convert a block data format to the raster format required by the display. Figure 7-5 shows the numbering of codewords in the frame-buffer and the ordering of pixels inside each block. Each time a codeword is accessed from the frame-buffer, only 4 pixels are read out from the lookup table, creating a raster output which can be sent directly to the display. That is, pixels P0-P3 corresponding to each codeword between CW0 and CW31 are read from the lookup table and then P4-P7 are read once again for codewords CW0 through CW31, and then P8-P11, and finally P12-P15. Thus, each codeword is accessed four times per image. This approach increases the frequency of codeword access relative to a scheme which reads each codeword only once (and stores P0-P15 in a line-buffer), but since the codewords are accessed at a much lower frequency relative to the lookup table data, and since the line-buffer access has been eliminated, the overall power due to memory access is reduced by approximately a factor of 1.5. This is an example of architectural restructuring to reduce the number of operations.

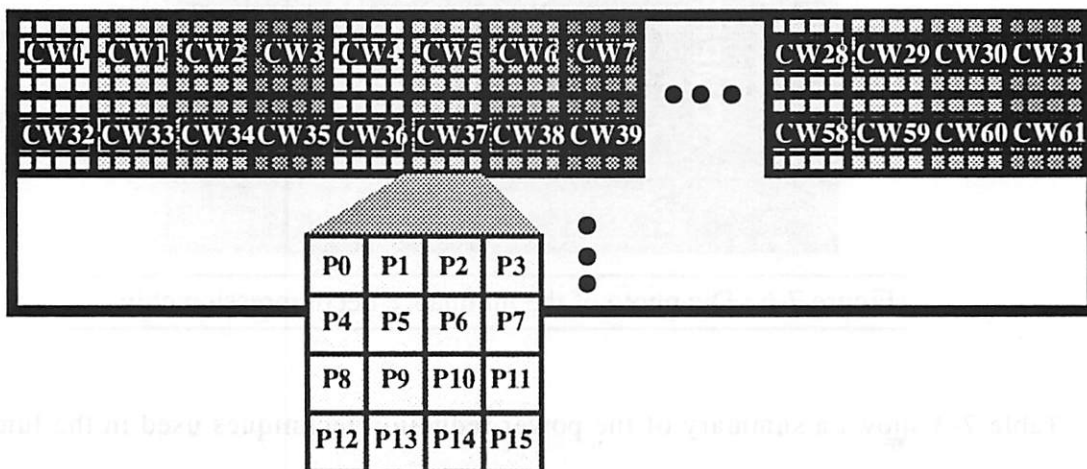


Figure 7-5 : Addressing of the frame-buffers and lookup table.

The control of the luminance decompression chip (address generation, clock generation, multiplexor selection, etc.) is performed by the video controller chip. The multiplexors for read/write address lines are integrated in the decompression chip. RLUTLine[1:0] controls the row of pixels that are read for a given codeword. The lookup table is programmable over the radio link

through the video controller chip. Figure 7-6 shows the die photograph of the luminance decompression chip. The chip consumes only $115\mu\text{W}$ running at 1.5V to support a video throughput rate of 2.5MHz for the 4" color display.

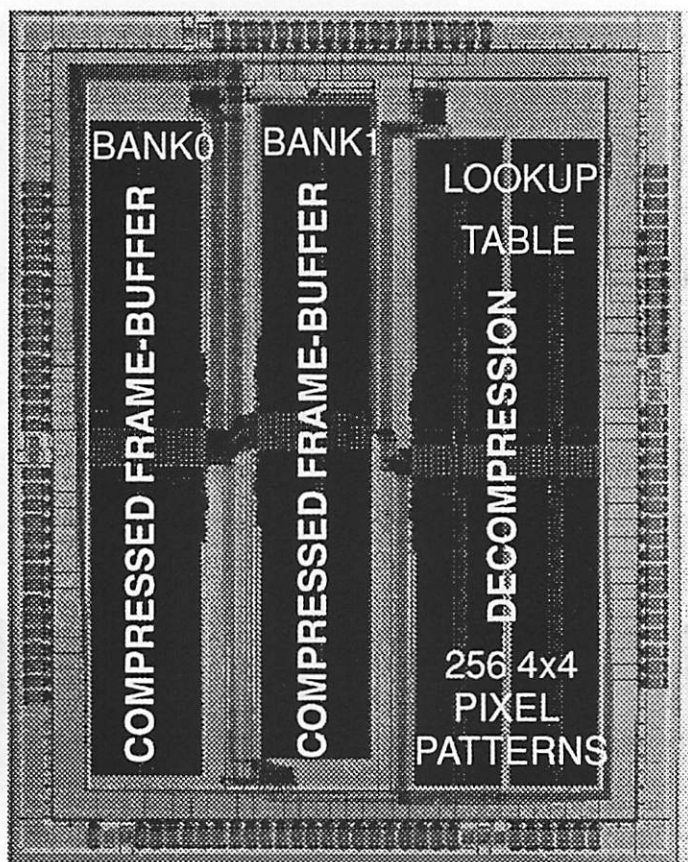


Figure 7-6 : Die photo of the luminance decompression chip.

Table 7-3 shows a summary of the power reduction techniques used in the luminance decompression chip.

Table 7-3 : Summary of power reduction techniques applied to the luminance decompression chip.

Design Approach	Power Reduction	Comments
Algorithm Selection	x5-10	Due to operation reduction Comparison of VQ decoding to typical DCT algorithms

Table 7-3 : Summary of power reduction techniques applied to the luminance decompression chip.

Design Approach	Power Reduction	Comments
Supply Voltage Scaling (through parallelism)	21	1.1V operation vs. 5V operation
Memory Restructuring	1.5	Elimination of Output Line Buffer Memory
Self-timed Outputs Drivers	x2	Elimination of glitching on the output of data busses. This number is for just the memory output busses
Bit Swing Reduction	x3.7	This reduction is only for the memory array. The global bus has rail to rail swing. Comparison is for 1.1V swing vs. 300mV swing

7.2.2 Chrominance Decompression Chip

The color video decompression chip is implemented using a very similar architecture to the luminance architecture. Figure 7-7 shows the block diagram of the chrominance decompression chip. Since the chrominance components, I and Q, are each sub-sampled by a factor 4, the amount of frame-buffer memory required is reduced by a factor of 4 for each component. Also, since they use the same addressing scheme, the I and Q data are stored in the same physical frame-buffer and the data is interleaved; i.e the 32-bit frame-buffer word is organized as: CWI0 CWI1 CWQ0 CWQ1. The 4:1 mux to select the codeword in the Y decompression chip is replaced by two 2:1 muxes, one for I (to select between CWI0 and CWI1) and one for Q (to select between CWQ0 and CWQ1). The chrominance (IQ) decompression chip also has two separate (256x16x6) lookup table memories. The IQ data is stored in the memory in a sign-magnitude format to reduce switching activity when accessed. It is possible to implement the chrominance chip using two of the luminance chips. The design of the chrominance chip was finished and functionally validated, but the two chip solution was chosen to minimize the number of chips that had to be fabricated.

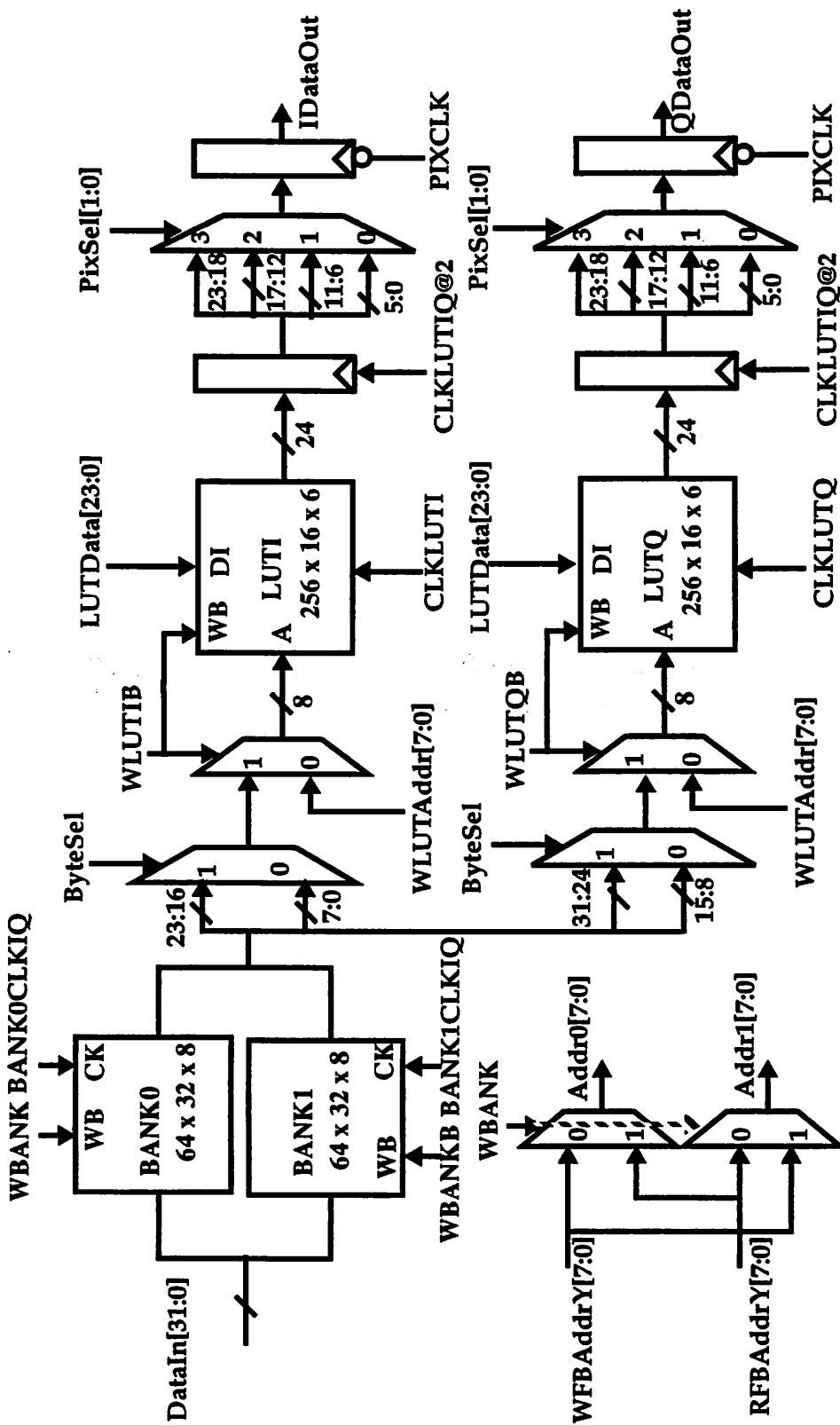


Figure 7-7 : Chrominance decompression chip block diagram.

7.2.3 Video Controller

The video controller performs all the control functions for the video decompression module. It generates all of the timing for the NTSC display, interfaces to the radio, controls the frame-buffers and lookup tables and performs synchronization for the system. A summary of the functions performed is outlined below:

NTSC sync generation for the 4" display: The LQ4RA01 4" color display is responsive to a standard composite sync signal with negative polarity of the same amplitude level as that of the video composite signal. The standard sync found in NTSC format has extra timing information such as an extra half line in one field (to distinguish between even and odd fields), and pre and post equalization half-line pulses during vertical sync that are not a necessity for proper operation of the LCD. A significant simplification of this protocol which involves block sync (with no equalization pulses) and the elimination of the half-line can be used to obtain a sync that still provides adequate synchronization information. The implemented sync signal is simply a scaled digital combination of VSYNC (vertical sync) and HSYNC (horizontal sync). The original and modified sync signals are shown in Figure 7-8.

Decodes and demultiplexes the radio data: similar to the function performed in the protocol chip (described in the previous chapter), the video controller chip decodes packets from the radio and demultiplexes between a frame-buffer FIFO and LUT FIFO. Contained inside each frame-buffer packet (that is sent over the frame-buffer FIFO) is the encoding of type information (TYPE = 0 => data is for the Y frame-buffer, and TYPE= 1 => data is for the IQ frame-buffer). Similarly, inside the LUT packet is encoding information about the LUT data type (Y, I or Q). Figures 7-9a and 7-9b show the protocol for the data coming over the lookup table and frame-buffer FIFOs.

Controls reading and writing of the frame-buffer memories for both Y and IQ decompression chips: it generates the read and write addresses for the ping-pong frame-buffers. It also generates the multiplexor control signals that selects the output of the frame-buffers (ByteSel

in Figure 7-4). It also performs the R/\overline{W} control (i.e controls when the frame-buffers switch between reading to writing) and generates the clocks for the SRAMs.

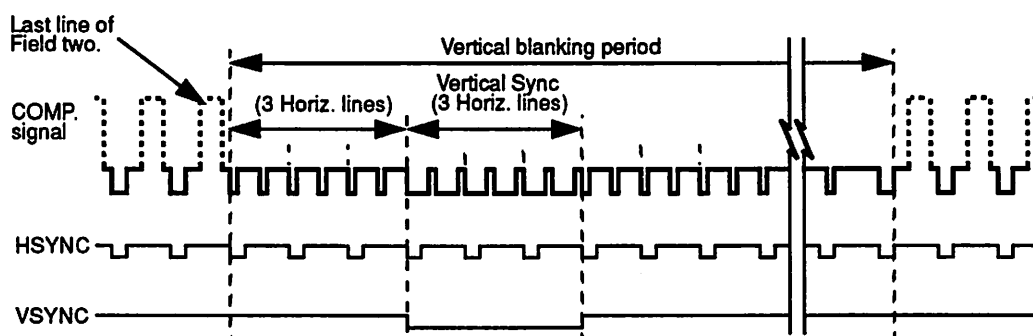
Controls loading and reading of the lookup table memories for Y, I and Q data: it generates the write address (WLUTAddr in Figure 7-4) and part of the read address (RLUTLine in Figure 7-4) for the lookup table memories. It also generates the multiplexor control signals that selects the output of the LUT (PixSel in Figure 7-4).

Support for variable sized packets: a decompressed frame of video can be broken into multiple packets and the size of the packets is variable, providing a flexible platform to test the effects of packet sizes. Also, this allows a form of inter-frame coding in which only the differences between the current frame and the frame corresponding to two image ago is sent. This is effective only if the BER is fairly low for the reasons explained earlier.

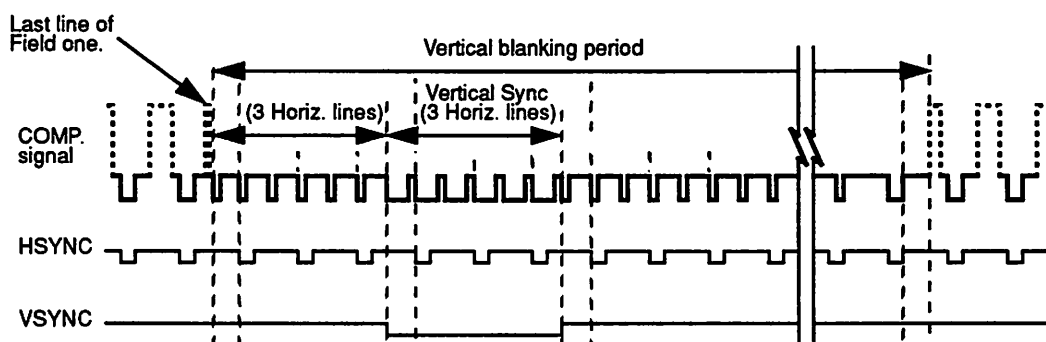
Matches pipeline delays in the system: since the system is pipelined, the output sync signals for the display must be delayed to avoid offsets of pixel data on the display. The timing signals needed by the color space converter are also generated in this chip.

Figure 7-10 shows the die photograph of the video controller chip.

Field One (Odd) Composite Timing



Field Two (Even) Composite Timing



(Both Fields) Modified (Simplified) Timing

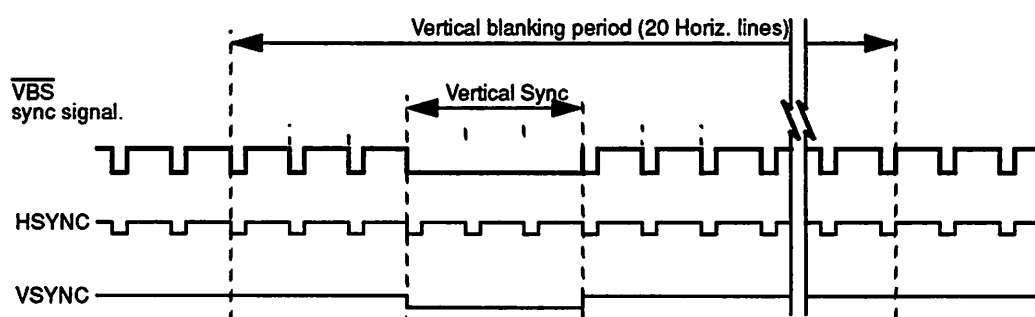


Figure 7-8 : Simplification of the NTSC system timing.

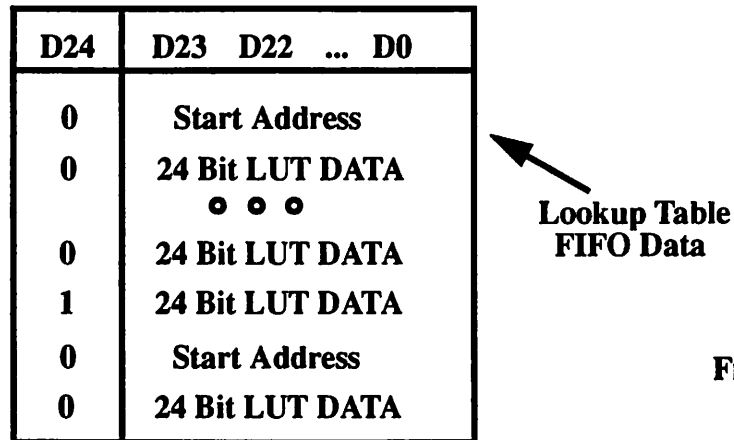


Figure a

- End of Packet Flag (EOP):
Bit 24 = Last Data Entry in Packet
- Start Address:
Bit 10: 9: Encoding of packet type (0 0 = Y, 0 1 = I, 10 = Q)
Bit [9 :0]: Write Lookup Table Address

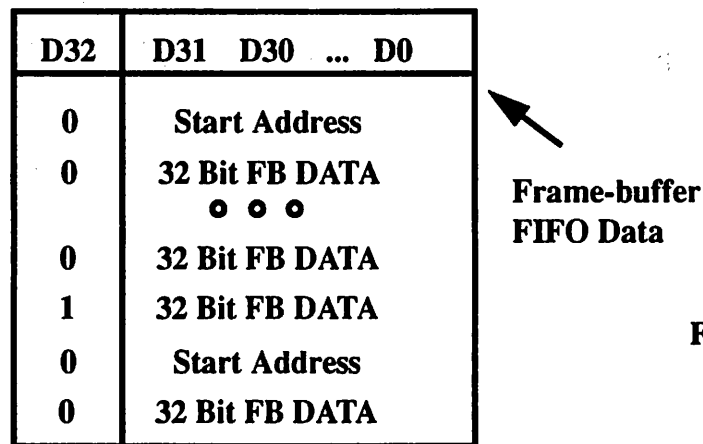


Figure b

- End of Packet Flag (EOP):
Bit 32 = Last Data Entry in Packet
- Start Address:
Bit 10: 1 = Flip Frame-buffer at EOP , 0 = Don't flip at EOP
Bit 9: Packet Type #, 1 = IQ & 0 = Y
Bit [8:0]: Write Frame-buffer Address (bit 8 is used only for the Y frame-buffer)

Figure 7-9 : Protocol for the lookup table and Frame-buffer FIFOs.

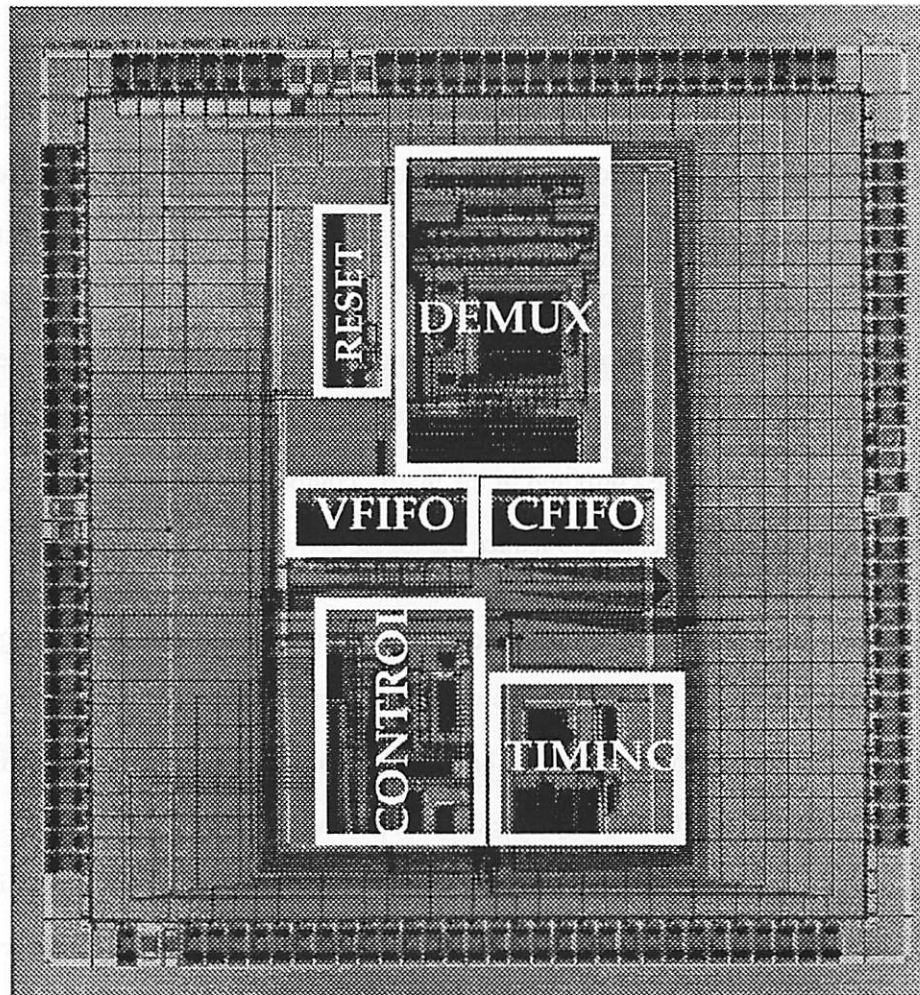


Figure 7-10 : Die photograph of the video controller chip.

7.2.4 Color Space Converter and Digital to Analog Converter

The digital YIQ information from the video decompression chips are sent to a color space converter which converts it to analog RGB to drive the 4" color LCD display from SHARP. The digital YIQ is first converted to digital RGB and then a triple digital to analog converter directly drives the display.

Digital YIQ to Digital RGB Conversion

In the YIQ to RGB translation, which involves multiplication with constant coefficients, the switching events are minimized at the algorithmic level by substituting multiplications with hardwired shift-add operations (in which the shift operations degenerated to wiring) and by optimally scaling coefficients. As described in Chapter 4, the multiplication of multiple coefficients with the same input was exploited to minimize number of shift-add operations. In this way, the 3x3 matrix conversion operation degenerated to 8 addition. The implementation was fully parallel and therefore there was no controller. For I/O communication (between the decompression chips and the color space chip) and in the matrix computation, sign-magnitude representation is chosen over two's complement to reduce the toggle activity in the sign bits.

At the architecture level, time-multiplexing was avoided as it can destroy signal correlations, increasing the activity. Figure 7-11 shows two alternate schemes for transmitting the I and Q data from the decompression chips to the color space converter chip. On the left is a fully parallel version in which I and Q have separate data busses. Also shown is the data for I and Q for a short segment in time. As shown, the data is slowly varying and therefore has low switching activity in the higher order bits. On the right is a time-multiplexed version in which there is a single time-shared bus in which the I and Q samples are interleaved. As seen from the signal value on the data bus, there is high switching activity resulting in higher power. Figure 7-12 transition activity for time-multiplexed activity vs. fully parallel architecture.

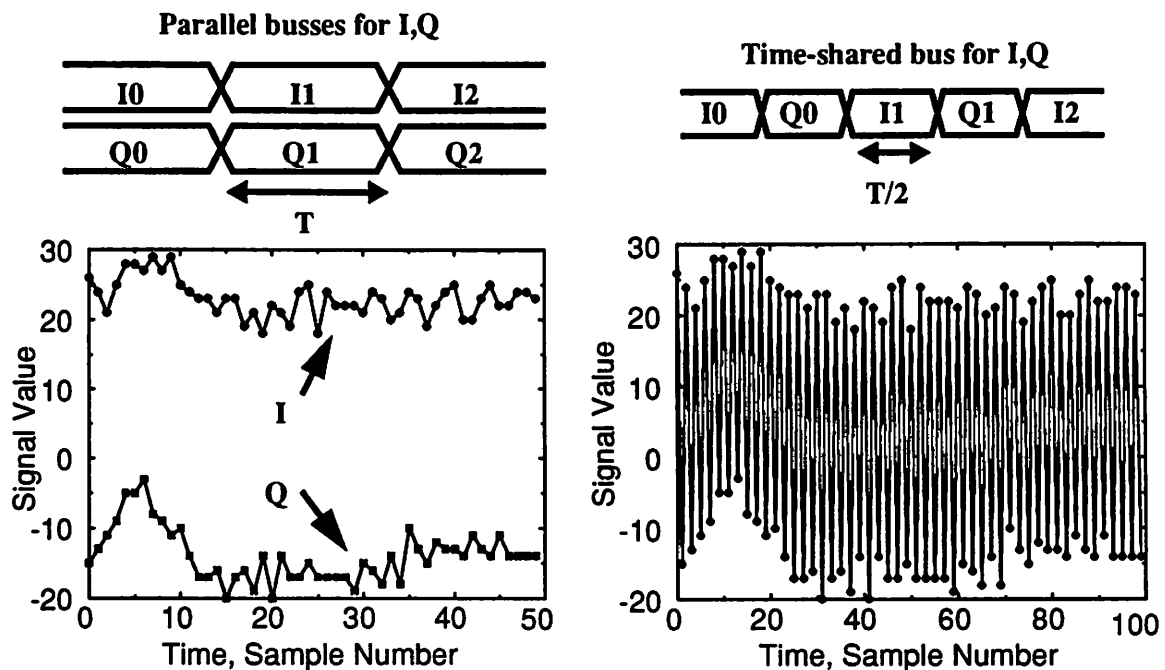


Figure 7-11 : Time-multiplexing can destroy signal correlation increasing activity.

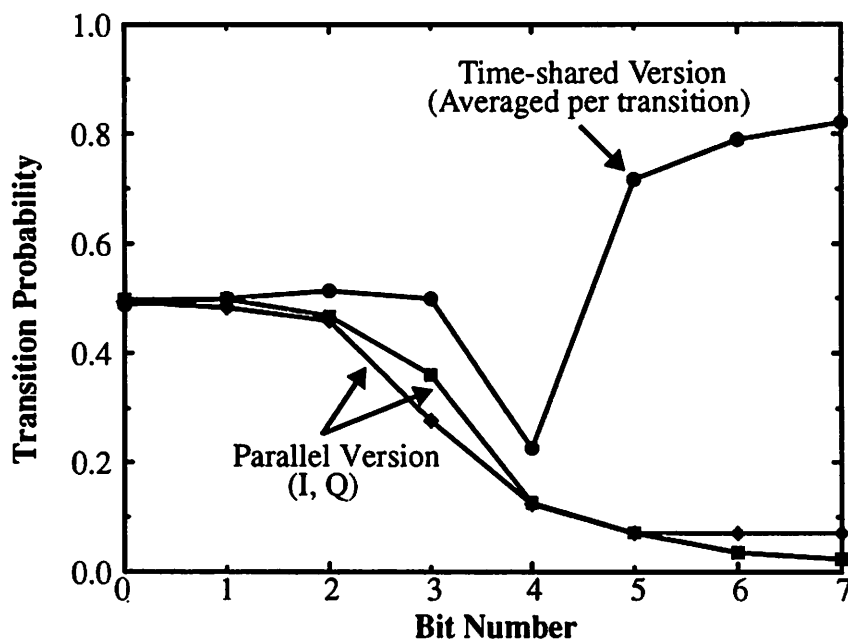


Figure 7-12 : Transition activity for time-multiplexing activity.

The digital YIQ \rightarrow digital RGB consumes only $100\mu\text{W}$ at 1.5V. This is more than three orders of

magnitude lower power compared to existing commercial solutions. Table 7-4 shows the power reduction achieved from various techniques compared to a commercial color space converter.

Table 7-4 : Summary of power reduction techniques applied to the YIQ -> RGB conversion

Design Approach	Power Reduction	Comments
Frequency Reduction (14MHz -> 2.5MHz)	5.6	The display resolution used was small and therefore the throughput requirement was much lower.
Supply Voltage Scaling (Parallelism)	11	1.5V operation vs. 5V operation
Optimized Cell Library	x2-3	Minimum sized devices, Single Phase Clocking.
Hardwired Shift-add Coefficient Optimization	7	The commercial implementation allowed programmability of coefficients (NTSC, PAL, etc.). Fixed application allowed optimization of the coefficients for minimal computational complexity - no real multipliers.
Fully Parallel Implementation	1.5-2	Hardware assignment that keeps uncorrelated data on different units.
Sign-Magnitude over Two's Complement Representation	1.2	For this application data was correlated, but there was still some statistical improvement.
Integrate Processing and DAC's	1.4	The DAC (as described in the next sub-section) was integrated onto the same chip and eliminated I/O power.
Bitwidth Reduction	1.3	Exploiting the low resolution of the display only 6 bits per plane was used instead of the typical 8bits.

Low-voltage Digital to Analog Converter

A low-voltage, low-throughput 6-bit DAC has been developed to drive the 4" SHARP LCD. The LCD display takes pixel data in the analog R,G,B format which has voltage levels compatible with the NTSC format (i.e $V_{pp} = 0$ to 0.7V). The DAC, shown in Figure 7-13, has an architecture based

on a conventional non-weighted current switched array. Based on a decoded 6-bit digital word, an appropriate number of current sources are turned ON and summed. The output voltage is obtained by passing current through an external resistor. Since the settling time requirement is quite low and since the LCD display has a high impedance capacitive load, the external resistor was chosen to be approximately $1K\Omega$ rather than 75Ω . This reduces the power consumption by more than an order of magnitude since the average current drawn from the supply is reduced by more than order of magnitude. The row decoding and column decoding logic is identical to the one presented in [Miki86]. The decoding logic was implemented using minimum sized low-power standard cells.

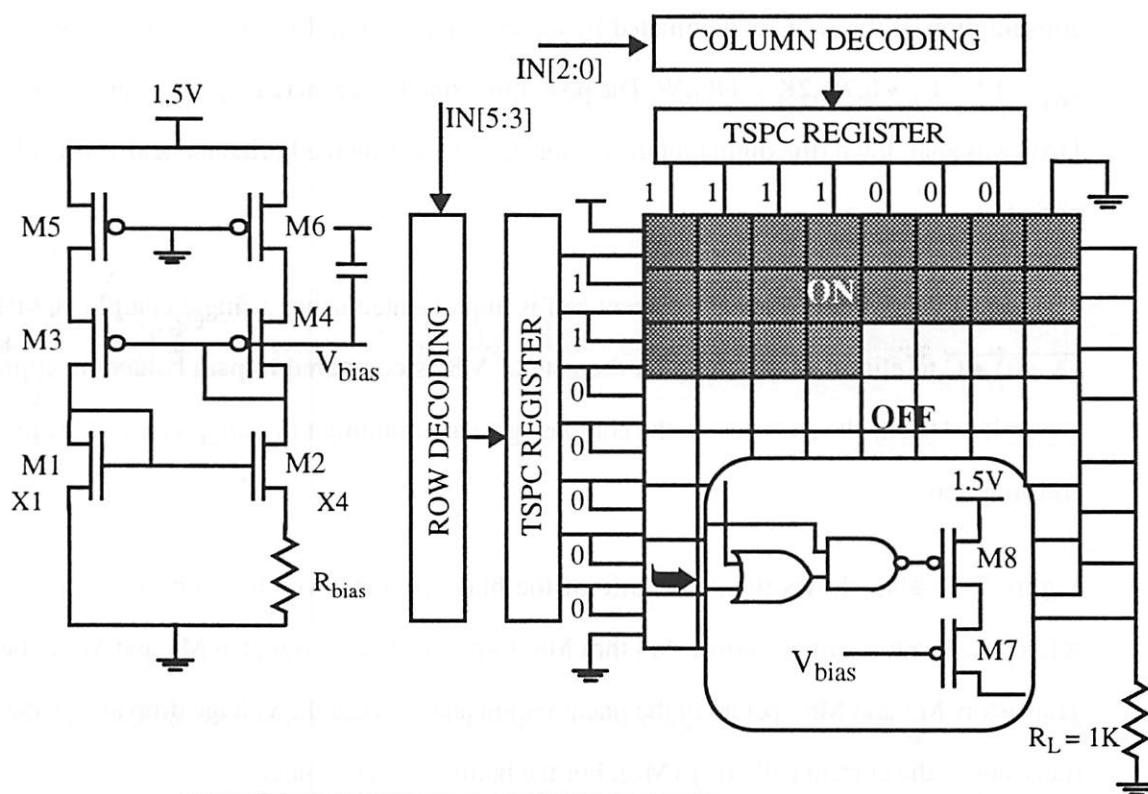


Figure 7-13 : Low-voltage 6-bit Digital to Analog converter.

A current source that operates at a reduced supply voltage of 2.7V has been developed [Miki92]. The current source used in this DAC is similar. However, due to the modest DAC throughput requirements for this 4" display, and to statistically reduce the power consumption by a factor of 2 (on average only half of the current sources will be ON) a single ended architecture is used instead

of the differential scheme. The current cell consists of stacked PMOS devices (M7 and M8) with the top transistor M8 being digitally switched, thus operating in the linear region. Therefore, effectively, the output resistance of the current source is the output resistance of a single transistor (M7) degenerated with a source resistor. To increase the output resistance of M7, the length of the device was made non-minimum. In order to operate the DAC down to a supply voltage of 1.5V and to meet the 0.7V_{pp} requirement for the output, the W/L of M8 was made large to keep the voltage dropped across M8 to less than 100mV. Also, the bias voltage was set up so that the $V_{gs} - V_t$ for M7 was approximately 200mV, allowing supply voltages to be as low as 1.2-1.3V. The power consumption of the DAC is dominated by the analog power and for a 1.5V supply: $P_{avg} = V_{dd} \cdot I_{avg} = 1.5 \cdot 1/2 \cdot 0.7/1.2K = 440\mu W$. The power measured in the actual system was lower since the DAC was shut down (the digital input was forced to 0) during the horizontal and vertical blanking periods.

The digital decoding inside each current cell is implemented using a single complex CMOS gate $(A + B) \cdot C$ to eliminate glitching on the gate of M8 as compared to path balancing approaches [Fournier91]. The device sizes on the complex gate are minimum to minimize the dynamic power consumption.

Figure 7-13 also shows the schematic of the bias circuitry which is a bootstrapped current reference. The top current mirror, M4 thru M6, forces the bias currents in M1 and M2 to be equal. Transistors M5 and M6 operate in the linear region and emulate the voltage drop across the switch transistor in the current cell array (M8). For the bottom current source:

$$V_{gs1} = V_{gs2} + I_{out} R_{bias} \quad (\text{EQ 135})$$

$$V_{tn} + \sqrt{\frac{2 \cdot I_1}{K_n \cdot \frac{W}{L}_1}} = V_{tn} + \sqrt{\frac{2 \cdot I_2}{K_n \cdot \frac{W}{L}_2}} + I_2 \cdot R_{bias} \quad (\text{EQ 136})$$

Assuming $V_{tn} = V_{tp}$, and $I_1 = I_2$ (forced by the top current mirror M3-M6),

$$I_{ref} = \frac{2}{K_n R_{bias}^2} \left(\frac{1}{\sqrt{\frac{W}{L1}}} - \frac{1}{\sqrt{\frac{W}{L2}}} \right)^2 \quad (\text{EQ 137})$$

Note that the reference current ($I_{ref} = I_1 = I_2$) is to first order independent of supply variations (this is due to the bootstrapped techniques used). The bias current is set by sizing M1 and M2 and by choosing R_{bias} . The current source can operate at down at low supply voltages even with a process that has a standard threshold voltage. The minimum operating voltage for this current reference is given by:

$$V_{ddmin} = V_{ds6} + |V_{tp4}| + V_{dsat4} + (V_{gs2} - V_{tn}) + I_{ref} R_{bias} \quad (\text{EQ 138})$$

In the above equation, V_{ds6} is very small ($< 100\text{mV}$ since M6 is in the linear region and by device sizing), V_{tp} for the MOSIS $1.2\mu\text{m}$ CMOS process is typically around 0.9V , and the last two terms can be made small ($100\text{mV} - 200\text{mV}$) by device sizing and choice of bias current. Therefore, the current source can operate down to the $1.2\text{-}1.5\text{V}$ range. Table 7-5 shows a summary of the power reduction techniques that were applied to the DAC design. .

Table 7-5 : Summary of power reduction techniques applied to the DAC

Approach	Power Reduction	Comments
Reduction of Load Resistance	16	The output resistance was chosen to be $1.2\text{K}\Omega$ since the load into the LCD is capacitive. This is much higher than the normal 75Ω resistance
Single Ended Architecture	2	Due to the relaxed speed requirements, a single-ended architecture was sufficient and statistically the power was reduced by a factor of 2
Voltage Reduction	3.3	A linear reduction in power consumption from 5 to 1.5V

Figure 7-14 shows the die photo of the color space converter and triple DAC.

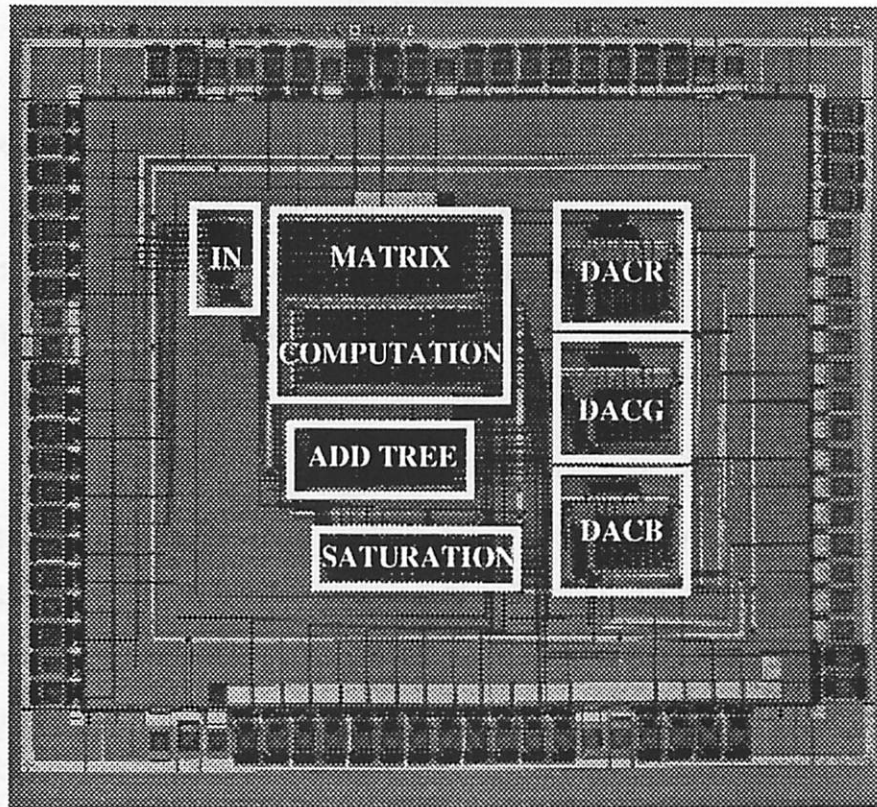


Figure 7-14 : Die photo of the color space converter and digital to analog converter.

7.2.5 Chipset Summary

Table 7-6 shows a summary of the video decompression chipset which is implemented using a standard mosis 1.2 μ m CMOS process. The minimum operating supply voltage for each chip is also shown along with the power consumption at the worst case voltage for this system implementation which is 1.5V. At the worst case supply voltage, the entire chipset consumes less than 2mW to perform the video decompression for the 4" active matrix SHARP display.

Table 7-6 : Summary of the video decompression chipset implemented in 1.2 μ m CMOS.

Chip Description	Area (mmxmm)	Number of Transistors	Minimum Supply Voltage	Power at 1.5V
Video Controller	6.7 x 6.4	31400	1.1V	150 μ W
Luminance Decompression	8.5 x 6.7	~250000	1.1V	115 μ W
Chrominance Decompression	8.5 x 9.0	~400000	1.1V	100 μ W
Color Space Conversion and Triple DAC	4.1 x 4.7	12500	1.3V	1.1 mW

7.2.6 Video Decompression Test Board

A test board using the video decompression chipset has been designed and tested. Figure 7-15 shows a block diagram of the video test board. There are 5 custom PGAs (a video controller chip, three luminance decompression chip - one for Y, I and Q, and a color space and triple-DAC chip), a crystal oscillator which generates the master clock (2.5MHz) from which all other timing signals are derived, and a power supply chip to generate the supply voltage for the custom chips (1.3V-1.5V). The video controller takes a serial compressed video stream (for test purposes from an Aerial board on SPARC station) and drives the Y, I, and Q decompression chips. The decompressed busses (Y,I, and Q) output from the decompression chips go to the color space converter which generates the analog signals that drive the LCD display. The video controller generates the composite sync for the LCD display. The LCD display requires 5V and -8V power supplies. Figure 7-16 shows the output of the video decompression chips operating at a supply voltage of 1.3V.

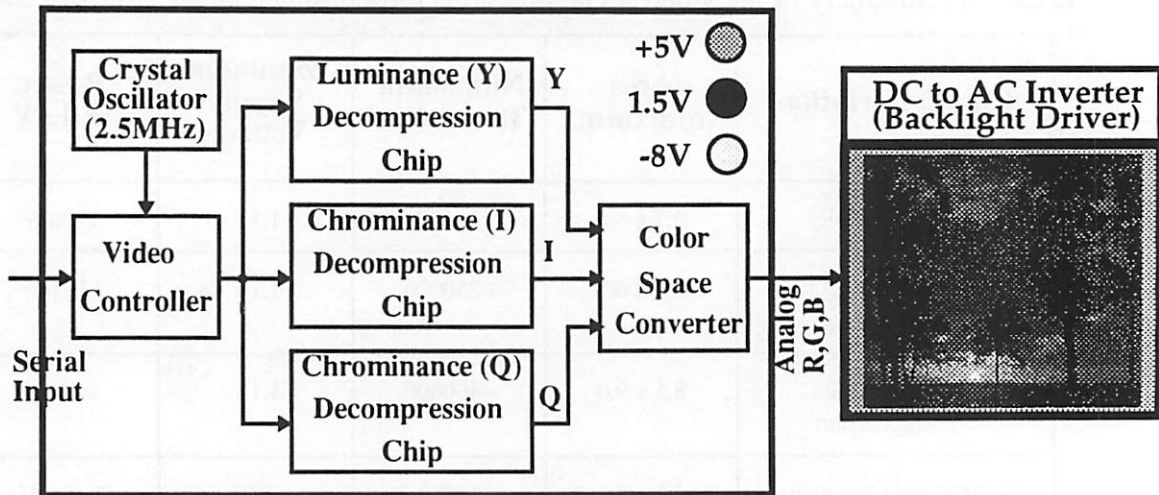


Figure 7-15 : Block diagram of the video decompression test board.

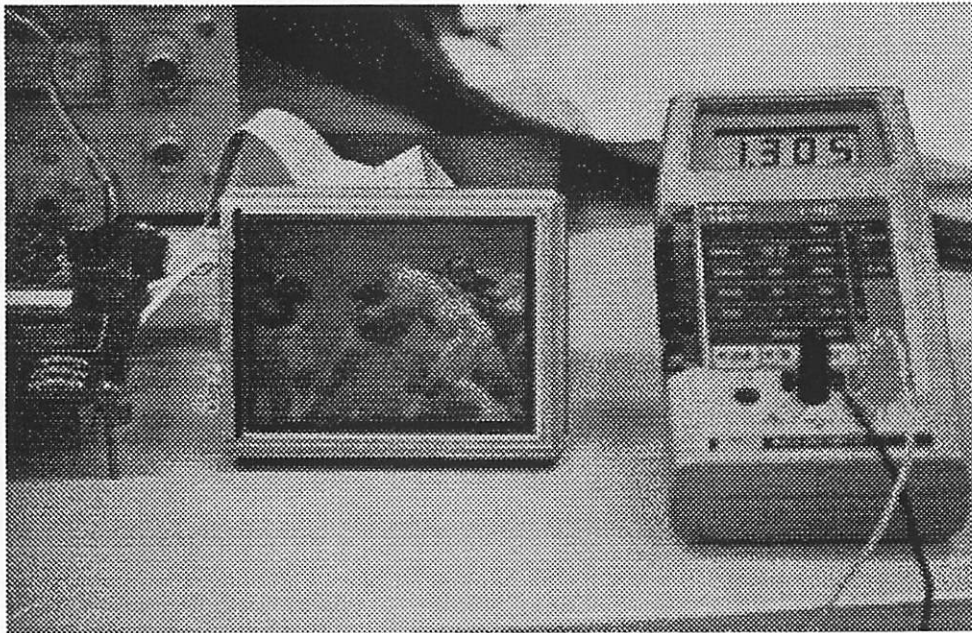


Figure 7-16 : Output of video decompression chips running at a supply voltage of 1.3V.

7.3 Summary

A video decompression has been described in this chapter that consumes two to three orders of magnitude lower power compared to existing commercial solutions that perform the same functionality. The chipset, consisting of four chips, performs radio interfacing, frame-buffering, video decompression, color translation, and digital to analog conversion all at a power level of less than 2mW operating at a supply voltage of 1.5V. It is clear from this design example that there is no single magical technique that can be used to achieve orders of magnitude reduction of power consumption; instead a system level solution is required that optimizes the algorithms, the architectures, the logic and the circuits.

At the algorithmic level, the vector quantization algorithm was chosen since the computational complexity was significantly lower than conventional DCT based algorithms. In the matrix multiplication, YIQ->RGB, coefficient optimization and bitwidth reduction resulted in significant reduction of the computational complexity over an implementation that uses true multipliers. In addition to minimizing the number of operations, the data representation was optimized for reducing the switching activity for memory access and in the logic of the matrix multiplication. At the architecture level, parallel techniques were used to scale the supply voltage to 1.1V while meeting video throughput constraints. By optimizing the memory architecture for the decompression and frame-buffer, the output line buffer typically used in video systems to convert block scan to raster scan was eliminated reducing the number of memory operation. Time-multiplexing was avoided since it can destroy correlations and increase the switching activity; that is, uncorrelated data was processed in different hardware units. At the circuit level, the digital logic was implemented using a low-power cell-library which uses minimum sized devices, optimized clocking, and optimized layout. As described in Chapter 6, the memory circuits reduced power by using reduced swing bitlines and self-timing to eliminate glitching. The digital to analog circuitry conserves power by using a larger output load resistance to reduce load currents, a single ended architecture to reduce the average power by a factor of 2, and voltage scaling down to 1.5V.

CHAPTER 8

Low-power Programmable Computation

Figure 8-1 shows a block diagram of the InfoPad I/O terminal, that was described in the previous chapters, emphasizing the I/O functionality that is suitable for implementation on a programmable processor.

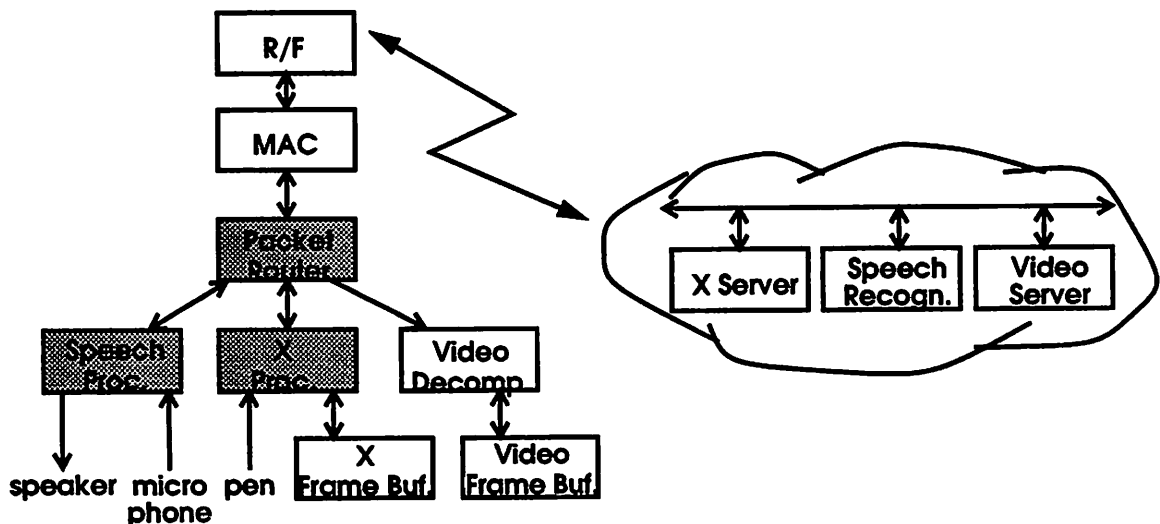


Figure 8-1 : Energy efficient programmable computation required in portables.

Even with the minimalist approach to computation adopted by Infopad, there are parts of functionality that are usually best implemented using general-purpose microprocessors, programmable DSPs, and programmable processor cores embedded in ASICs. Examples include media access and data link layer protocol processing for wireless RF communication between the portable terminal and a basestation, the DSP processing for speech coding, and processing for parts of the graphics display server (X server for example) that may reside on the portable terminal.

The need for the use of software programmable components is even more in other portable devices - such as most PDA type devices - that do not adopt as extreme a partitioning of computation between the network servers and the portable device as is adopted in Infopad. For example, the InfoPad system requires that the X-server computation is completely performed on the backbone network. Another partitioning scheme could be to move just the X-server computation to the terminal [Weiser93] (while the client applications still run on the backbone network), which then will require a programmable processor; however, as described earlier this approach will require extra bandwidth for protecting error sensitive control data of X-protocol. Implementation using software may be needed because: (i) the algorithmic and logical (control) complexity of the application may preclude dedicated hardware and make software the only practical choice, or (ii) the application may not operate continuously or different functionality may be needed at different times so that a time-multiplexed software implementation is more cost effective.

8.1 Architectural Approaches to Low-power

The primary focus of this chapter is on architectural techniques to improve the energy efficiency for parts of the portable devices that need to be implemented using software programmable components such as microprocessors, DSPs, and embedded processor cores. So far the main approaches for lower power consumption and increased battery life for general purpose computers have been restricted to using a limited amount of voltage scaling in the form of using 3.3V parts

instead of 5V parts (often coupled with lower clock speed), and straightforward shutdown of the system power supply and/or clock - various notebook and laptop computers on the market illustrate these techniques. However, much higher reductions in power consumption are possible by using more sophisticated architectural and implementation strategies than straightforward shutdown and the use of 3.3V components. For example, a proper addressing of the problems of *when* to shutdown, and *how* to scale the voltages can result in substantial improvement in energy efficiency with no or little loss in performance.

Here CMOS technology is assumed and that the average power consumption of a CMOS gate is dominated by the switching component and is given by: $P = \alpha CV_{dd}^2 f$. The expression for power consumption suggests several strategies for increasing the energy efficiency (reducing the power consumption while maintaining the computation speed):

Activity based system shutdown: Many computations are “event-drive” in nature with intermittent computation activity triggered by external events and separated by periods of inactivity - examples include X server, communication interfaces etc. An obvious way to reduce average power consumption in such computations would be to shut the system down during periods of inactivity. Shutting the system down - which would make power consumption zero or negligible - can be accomplished either by shutting off the clock ($f=0$) or in certain cases by shutting off the power supply ($V_{dd} = 0$). However the shutdown mechanism has to be such that there is no or little degradation in speed - both latency and throughput are usually important in event-driven computations.

Supply voltage reduction: Not all computation implemented as software is “event-driven” in nature - data-flow functions such as DSP are “continuous” in nature. Obviously, shutdown is not an effective mechanism for these systems. An alternative strategy is to operate at the lowest possible supply voltage, as is suggested by the quadratic dependence of power on supply voltage V_{dd} . Unfortunately, operation at reduced supply voltage comes at the expense of reduced circuit

speed. Fortunately, however, if only throughput and not latency is the metric of speed - as is true for many “continuous” applications such as speech coding - the reduction in circuit speed can be compensated for by architectural techniques like pipelining and parallelism that increase throughput so that the net result is a more energy efficient system operating at a lower voltage but same throughput. Compiler techniques for effective parallelization and pipelining play an important role in the success of this strategy to energy efficiency.

Switching activity reduction: In addition to the above two strategies that are specific to event-driven and continuous computations, there is a range of architectural strategies that are applicable to both types of computation. Such strategies in general try to make the system more energy efficient by reducing the switching activity α . As described in Chapter 4, reduction in α can be accomplished in a variety of ways - computation restructuring, communication restructuring, optimizing the memory storage architecture and hierarchy, change the data encoding etc.

8.2 Shutdown Techniques

An obvious mechanism for saving energy is to shut down parts of the system hardware that are idle because they are waiting for I/O from outside the system or from other parts of the system. While it is shutdown, the system consumes near zero power. Shutdown is a particularly relevant mechanism for event-driven interactive computation, such as is found in the graphics server and other user-interface related functions of InfoPad like portable devices. As shown in Figure 8-2, such computations are in one of two states: they are either blocked while waiting for an I/O event, or are performing computation. When running on a dedicated CPU, the application will alternate between a *blocked* state where it stalls while waiting for external events such as a key press or a mouse click, and a *running* state where it will execute instructions to perform computation. If $T_{blocked}$ and $T_{running}$ be the average time spent in the *blocked* and the *running* states respectively, then one can improve the energy efficiency by as much as a factor of $1 + T_{blocked}/T_{running}$ provided the system is shutdown whenever it is in the *blocked* state.

There are two main problems in shutdown - *how* to shutdown, and *when* to shutdown. The first problem is addressed by mechanisms for stopping and restarting the clock ($f=0$) or for turning off and on the power supply ($V_{dd}=0$). The second problem is addressed by policies such as “shut the system down if the user has been idle for 5 minutes”. Although the two problems are not really independent because the decision about when to shutdown depends on the cost (in time and power) of shutting down and restarting the system, the focus here is primarily on the problem of deciding when to shutdown while being cognizant of the available shutdown mechanisms.

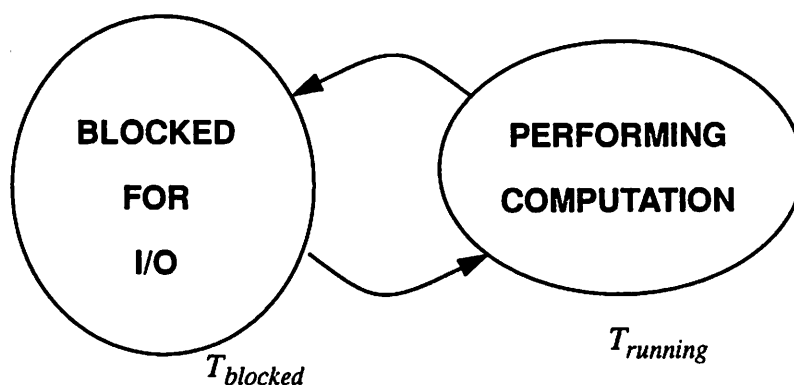


Figure 8-2 : Event-driven applications alternate between *blocked* and *running* states.

Simple shutdown techniques, for example shutting down after a few seconds of no keyboard or mouse activity, are already used to reduce power consumption in current notebook computers. However, the event-driven nature of modern window systems, together with efficient hardware shutdown mechanisms provided by newer microprocessors and system controllers, suggests the possibility of a more aggressive shutdown strategy where parts of the system may be shutdown for much smaller intervals of time while waiting for I/O events. Such a shutdown mechanism is presented here for use on a portable X-terminal type device where a simple algorithm derived from analysis of actual traces is used to predict the length of the time spent in *blocked* state in order to minimize the impact on interactive speed, while the more frequent system shutdowns result in dramatic reductions in effective computation energy.

8.2.1 Conventional Shutdown Approaches

The portable computers available now use various shutdown techniques that are all variants of the following basic scheme: “Go to Reduced Power Mode after the user has been idle for a few seconds/minutes”. Figure 8-3 illustrates the philosophy underlying the conventional approaches to shutdown. A drawback of this straightforward policy is apparent - the system continues to waste energy while it idly waits to check for lack of user activity for a few seconds/minutes. Our experimental traces with an X server (Section 8.2.2) showed that while the X server spends 96-98% of its time in the *blocked* state, the average time spent on each visit to the *blocked* state is short (\ll a second). The conventional shutdown schemes will therefore fail to exploit the large reduction in energy that is otherwise possible.

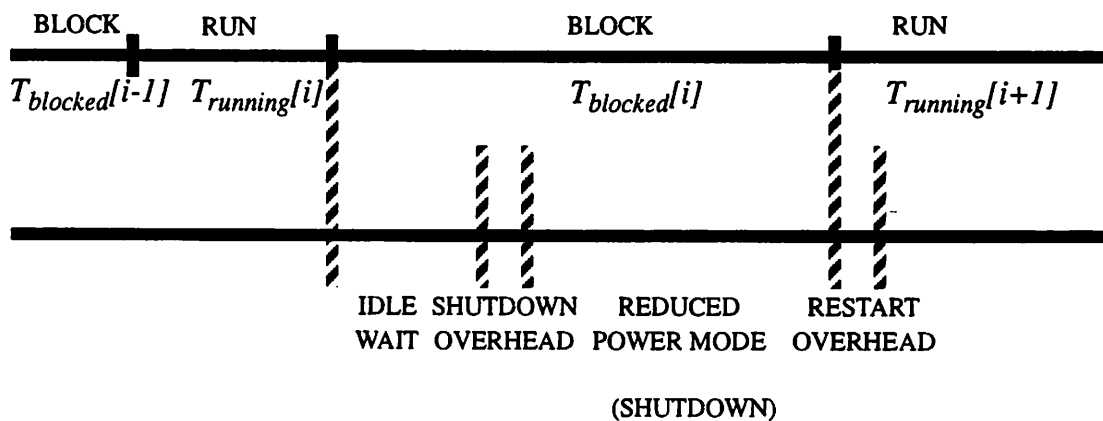


Figure 8-3 : Conventional shutdown approaches.

Typical of the conventional approaches to shutdown are the schemes used in Apple’s popular Mac Powerbook series of portable computers, which have three different types of reduced power modes based on shutting down parts of the system [Apple92]. A Power-Management IC controls the process of entering and exiting these shutdown modes by monitoring the input devices and the battery voltage, controlling the contrast of the LCD display.

Rest Mode

A technique termed *Power Cycling* is used in the rest mode on most models of PowerBooks. The computer enters the rest mode after 2 seconds of idle time upon which the processor registers are saved and the processor is powered down. However, the I/O devices remain on, the screen cursor continues to blink, and the keyboard continues to be scanned. After 1/60 second (16.7 ms) the power to the main processor is restored. If there has been no I/O activity, the processor is again shutdown for another 1/60 second.

The power consumption is reduced by up to 90%, i.e. by as much as x10, *while in the rest mode*. However, because the rest mode is entered only after 2 seconds of idle time, the effective reduction in power consumption is much less dramatic. Analysis of experimental battery life data reported in [Berkoff93] for various usage scenarios of a PowerBook suggests that enabling the rest mode reduces the power consumption of the processor and logic section alone by x6 whereas the power consumption of the entire system, including the disk and display backlight, is improved by x2. Our experiments, described later, suggest improvements in the processor power consumption in the neighborhood of x1.5-x2 if this strategy is used for a processor running an X server.

The impact of power cycling on interactive applications is negligible.

Sleep Mode

The sleep mode is entered after the computer has been idle for a user selected period of time, typically in the range of a few minutes. The computer exits the sleep mode and reverts back to the normal mode on the occurrence of an external event, such as a key press or a modem ring-detect. A decrease in computers responsiveness compared to the rest mode is traded-off against an increase in energy conservation by shutting down the peripheral functions as well: the I/O ports, the disk ports, the display, the sound circuits. Power is retained only to the RAM and the Power-Manager IC.

Since the disk goes to sleep too, the effect on interactive applications is substantial - spinning the disk down and back up takes a substantial time. Further, the break-even point when it is worth spinning-down the disk and spinning it back up (because spinning-up the disk takes much increased power) is about 15 seconds [Berkoff93], which implies that sleep mode is effective only for moderately long periods of idle time.

Shutdown Mode

This mode is typically entered on an explicit command from the user. It has the worst impact on the responsiveness of the computer but saves the maximum power - on most PowerBook models the entire computer, including the Power Manager IC, is turned off - only a tiny amount of power is drawn for a parameter RAM. The computer does not respond to external events at all while in this mode, and one needs to restart it.

8.2.2 Predictive Shutdown Approaches

The straightforward shutdown schemes described above either show only a moderate overall improvement in energy consumption with negligible loss of computer responsiveness as in the rest mode, or show a higher degree of improvement but at the cost of much decreased computer responsiveness, as in the case of sleep mode and shutdown mode. In this section we explore a shutdown mechanism where we try to predict the length of idle time based on the computation history, and then shut the processor down if the predicted length of idle time justifies the cost - in terms of both power and responsiveness - of shutting down [Srivastava94]. The basic philosophy behind the predictive approach can be summarized as follows: "Use computation history to predict whether $T_{blocked}$ will be large enough ($T_{blocked} > T_{cost}$) to justify a shutdown". Our analysis, which is based on real-life traces, suggests that our predictive shutdown approach leads to a much higher reduction in effective processor power, than is obtained with straightforward non-predictive shutdown schemes, with only a small loss in responsiveness.

Helpful Trends in Computing & Communications

Two developments in computing and communication motivate more sophisticated shutdown mechanisms. First, newer microprocessors, such as Intel 486SL, and AT&T Hobbit, provide power management support integrated with the system control, thus enabling implementations of shutdown that are efficient both in terms of time and hardware cost. Power management features typically include the ability to shut down the clock and/or power supply to various parts of the system, and efficient mechanisms to store and restore the processor state for power down.

Second, the integration of computing and communication is resulting in a paradigm where increasingly the computer will become a device to access remote computation and information servers across a network. This is particularly the case with the proposed wireless devices such as the InfoPad. With the actual application-specific computation being done primarily on network servers, the computation that is left on the personal device is increasingly related to the graphic user interface and windowing system functionality. A good example of this would be a wireless multimedia device which, besides audio and video services, provides the capability to run X system based remote client applications. In such a case the general purpose programmable computer in the portable device would primarily be running the X display server (or a part of the server), and may be some important local clients such as the window manager - similar to conventional X terminals. Such software is event-driven in nature, where the events arise because of user interaction. Except for extremely graphics intensive applications, a vast majority of time such software just idles while waiting for user input. Shutting down the hardware while the software is waiting for external events can give rise to potentially huge savings in computation energy.

Potential for Reduction in Computation Energy by Shutdown

Adopting the "X Terminal" model of computation described above, we studied the potential for energy reduction in the case of an X display server. Our analysis below, based on experimentally

obtained traces of X server state, suggests that potential energy savings as high as x30 to x60 are possible.

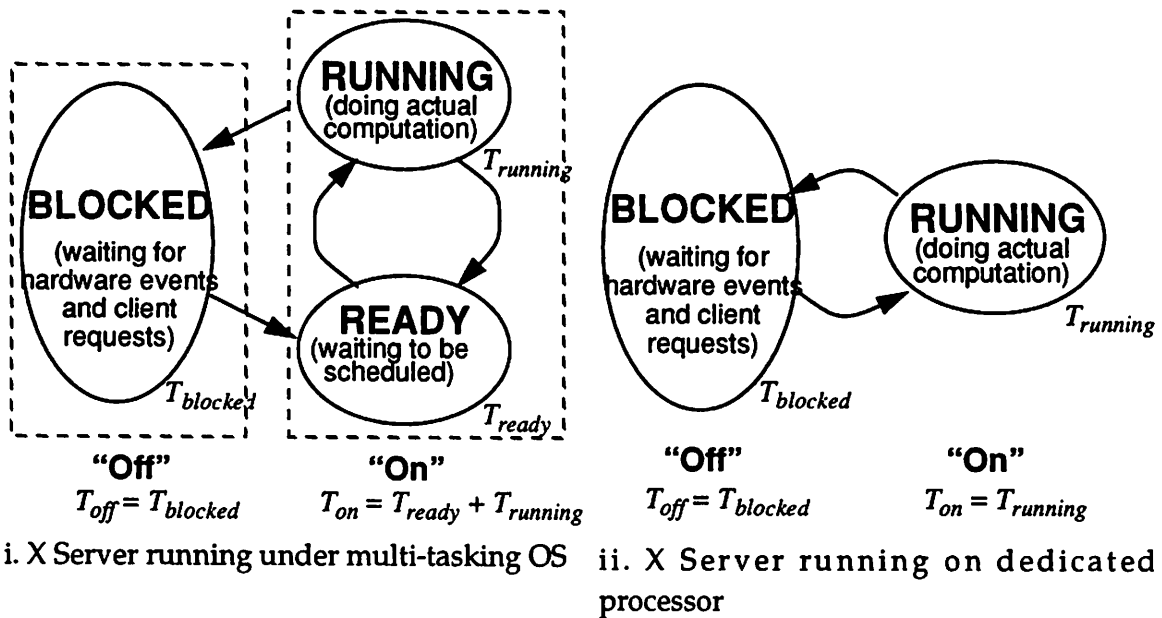


Figure 8-4 : States of X Display Server Process

Figure 8-4 shows that the X server process running under a multitasking operating system, such as UNIX, is in one of the three states:

- a. It may be *blocked* or stalled while waiting for either an hardware event (key press or mouse activity) or requests from one of the existing client applications or a connection request from a new client.
- b. It may be *running* on the processor doing some computation.
- c. It may be *ready* to run but is waiting for the scheduler of a multi-tasking time-shared operating system to schedule it to run.

Let $T_{blocked}$, $T_{running}$, and T_{ready} be the time spent in the three states. If the X server were to run on a dedicated processor, as would be the case in our X Terminal model of computation, the server will never be in the *ready* state - it would always be either *blocked* or

running - i.e. $T_{ready} = 0$. The hardware does nothing while the server is in the *blocked* state and can therefore be shut down by stopping the clock or by saving the state and then power down. The fraction $T_{blocked} / (T_{blocked} + T_{running})$ of the total time (in this case $T_{ready} = 0$) that the server spends in the *blocked* state corresponds to the maximum possible reduction in computation energy.

In order to estimate $T_{blocked} / (T_{blocked} + T_{running})$ under the condition that $T_{ready} = 0$, and thus the potential energy reduction in the X Terminal model of computation, we instrumented the X11R5 server from MIT, running under SunOS on a SPARCstation 2, to measure $T_{blocked}$ and $T_{running} + T_{ready}$. Unfortunately, T_{ready} is non-zero due to the multi-tasking time-shared nature of SunOS, and there is no easy way to measure $T_{running}$ and T_{ready} separately without modifying the kernel. Even though we ran our experiments on an unloaded workstation with no local X clients except for the console window and the window manager, there are background daemon processes over which we did not have any control. However, it is important to realize that the fraction $T_{blocked} / (T_{blocked} + T_{running} + T_{ready})$, which we can easily measure, is a lower bound on the fraction of energy that can be saved under ideal conditions. So our results err on the safe side - the estimate of the potential energy reduction that will result from using $T_{blocked} / (T_{blocked} + T_{running} + T_{ready})$ will thus be pessimistic.

From now on we refer to the *blocked* state as the *off* state, and the *running* and *ready* states jointly as the *on* state. Further, $T_{off} = T_{blocked}$ and $T_{on} = T_{running} + T_{ready}$. Our instrumented X11R5 server measures T_{off} and T_{on} , and we use the ratio $T_{off} / (T_{off} + T_{on})$ as a lower-bound on the maximum possible reduction in energy under ideal shut down. Figure 8-5 shows a sample trace of the time spent by the X server in the *off* and the *on* states over a small stretch of time, and Table 8-1 shows the results obtained by analyzing several traces obtained from real X sessions. These X sessions consisted of running our instrumented X11R5 server on a SPARCstation 2 together with the window manager *olvwm* and console

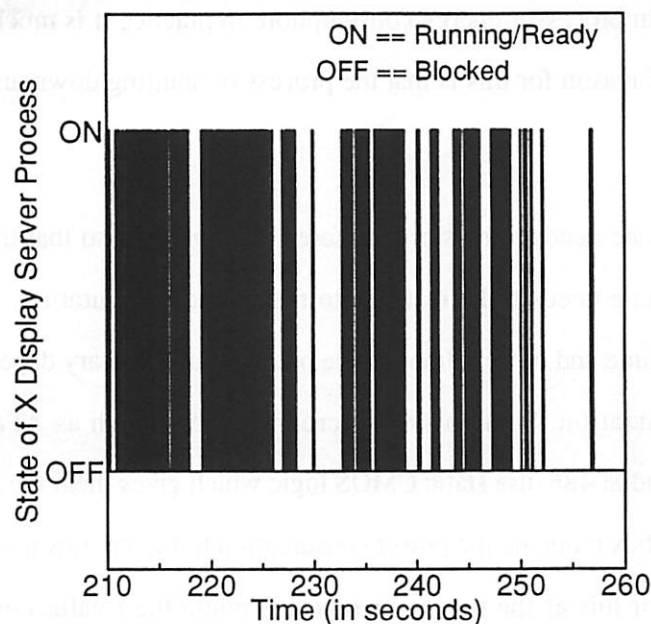


Figure 8-5 : Sample trace of X server state under UNIX.

	Trace		
	1	2	3
Trace Length (sec)	5182.48	26859.9	995.16
T_{off} (sec)	5047.47	26427.4	960.82
T_{on} (sec)	135.01	432.5	34.34
$T_{\text{off}}/(T_{\text{off}}+T_{\text{on}})$	0.9739	0.9839	0.9655
Maximum Energy Reduction (conservative estimate)	x 38.4	x 62.1	x 29.0

Table 8-1 : Analysis of X Server Traces for Potential Energy Reduction

window *contool* running locally, and several typical X clients, such as *xterms*, *FrameMaker*, *xclocks*, *mailtool*, *cm* (calendar manager) etc., running remotely. As is evident from these traces the X server spends most of its time, ranging from 96.5% to 98.4%, in the *off* state suggesting that energy reductions ranging from x29 to x62 under ideal shutdown conditions.

Shutdown Overhead

Although the X server trace analysis in the previous section suggests that there is a tremendous

potential reduction in processor energy consumption, in practice it is much harder to realize this reduction. The chief reason for this is that the process of shutting down and restarting has a cost associated with it.

Enough processor state needs to be stored before shutting down so that the computation can be restarted, and the state needs to be restored to restart the computation. This process requires additional compute time and power although the precise numbers vary depending on the hardware and software organization. Some newer microprocessors, such as AT&T Hobbit and some versions of 80386 and 80486, use static CMOS logic which gives them the ability to shutdown by stopping the clock, thus reducing the power consumption to the microwatts range. Very little state needs to be saved for this as the processor registers retain their values even when the clock is stopped. This type of shutdown can be accomplished in a few microseconds. However, in other cases the entire processor state may need to be stored in the memory - for example if the processor uses dynamic logic (as most processors do) or if one wants to conserve even more energy by doing a power down instead of just stopping the clock. The overhead now increases substantially - to hundreds of microseconds to several milliseconds - as work similar to a context switch needs to be performed. At the penalty of more overhead, even more energy can be saved by storing the state on the disk as opposed to the main memory as it is no longer necessary to refresh the main memory which is typically DRAM. The overhead now increases to hundreds of milliseconds to several seconds.

The overhead due to shutdown creates problems. If the overhead is large enough, then there is problem of deciding *when* to shutdown. The time spent in the blocked state must be long enough to justify the overhead. If, after deciding to shut down, the blocking interval turns out to be too small, then one has to pay not only a power penalty, but more importantly, an effective slowing down of the computation speed because now the computer has to block for an interval higher than necessary. This slowing down translates into increased latency which, once it increases beyond a certain point, has an adverse impact on the interactive behavior of applications like X server.

Energy Reduction by Predictive Shutdown

The ideal situation, of course, would be to make the overhead zero or very small - but as the previous discussion suggested, overhead really is a function of the hardware and type of shutdown. The next best thing would be an a priori knowledge of the length of the blocking interval right at the beginning. Unfortunately, this is physically not possible.

One therefore has to resort to a heuristic to decide when to shut down. The approach used is based on the recent computation history and a prediction is made whether the idle time would be long enough to break-even with the shutdown overhead. Results demonstrate that for reasonable values of shutdown overhead, the predictive approach allows much higher energy savings to be achieved compared to the straightforward non-predictive approach, while the degradation in interactive performance is negligible.

Restricting our analysis to X server running on a dedicated processor, the server process starts in the *on* state, and makes alternate transitions from *on* to *off*, and from *off* to *on* state.

Let $T_{on}[i]$ and $T_{off}[i]$ be the time spent by the X server in the *i*-th visit to the *on* and the *off* state respectively, for $i = 1, 2, 3, \dots$. Relating to our previous terminology, T_{on} is the sum of $T_{on}[i]$ over all *i*, and similarly T_{off} is the sum of $T_{off}[i]$ over all *i*.

Further, let T_{cost} be the time overhead associated with the process of shutting down. We interpret this to mean that once we shutdown, it takes time at least equal to T_{cost} before the computation can begin. Thus, if it turns out that the event that wakes up the X server and hence restarts the computation occurs before the expiry of this T_{cost} , a penalty is paid in terms of increased latency over the case with no shutdown.

Conventional idle-time based shut-down mechanisms, such as Apple's *Power Cycling*, are non-predictive in nature. In our model such a scheme involves making a decision to shutdown if the time spent in the *off* state after the most recent entry (say, the *i*-th entry) has exceeded a certain

idle-time threshold, which is 2 seconds in Apple's scheme. In other words we decide to shutdown on i -th entry to the *off* state once $T_{off}[i]$ has exceeded 2 seconds. Of course, once we decide to do this, the computation cannot be restarted for at least T_{cost} time, so that if $T_{off}[i]$ would have been less than $2 + T_{cost}$, a penalty is paid in increased latency. The intuition behind such a scheme is that if the idle time has exceeded 2 seconds then most likely the user going to remain idle for a long time. However, this scheme has two disadvantages. First, no shutdown is done for the first 2 seconds (or whatever is the idle time threshold that is chosen), and power is wasted during that period. Second, this scheme is able to take advantage of only relatively long idle periods - as our analysis showed, the average $T_{off}[i]$ is less than a second, and thus relatively short idle periods are more common.

Prediction of $T_{off}[i]$

To address the deficiencies of the conventional idle-time based shutdown scheme, we designed and studied a predictive scheme for deciding when to shut down. The idea is that a simple heuristic rule is used that, on the i -th entry into the *off* state, predicts whether $T_{off}[i]$ is going to be long enough to justify shutting down. Specifically, the heuristic rule predicts whether $T_{off}[i] \geq T_{cost}$, and if so it is decided to shut the processor down.

The heuristic rule uses the computation history to make the prediction. In our case the previous values of $T_{on}[i]$ and $T_{off}[i]$ form the obvious computation history. In particular, $T_{on}[1] \dots T_{on}[i]$, and $T_{off}[i] \dots T_{off}[i-1]$ can be used to predict $T_{off}[i]$. Further, the prediction rule itself needs to be simple to evaluate and not require too much state information. Our intuition led to two approaches and the corresponding prediction rules:

- Our first approach was to use regression analysis to arrive at a model for predicting $T_{off}[i]$. We used *Mathematica* to analyze the traces obtained for the X11R5 server running on a SPARCstation 2 and arrived at the following model for $T_{off}[i]$ in terms of $T_{off}[i-1]$ and $T_{on}[i]$:

$$T_{off}[i] = 0.0740018 + 0.553733 T_{off}[i-1] - 0.00947348 T_{off}[i-1]^2 + 1.42233 T_{on}[i] + 1.13883$$

$$T_{off}[i] T_{on}[i] - 1.49143 T_{on}[i]^2$$

A quadratic model with cross terms was used because the scatter plot of $T_{off}[i]$ versus $T_{on}[i]$ in Figure 8-6, and a similar plot of $T_{off}[i]$ versus $T_{off}[i-1]$ shown in Figure 8-7, both suggested a hyperbolic behavior.

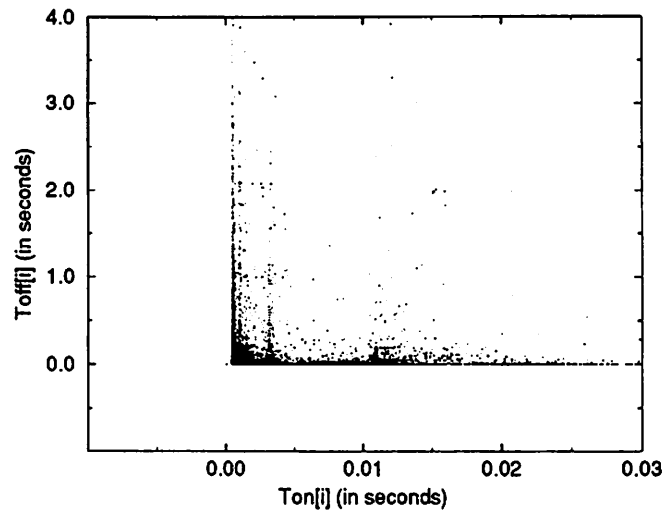


Figure 8-6 : L-shaped $T_{off}(i)$ versus $T_{on}(i)$ Scatter Plot

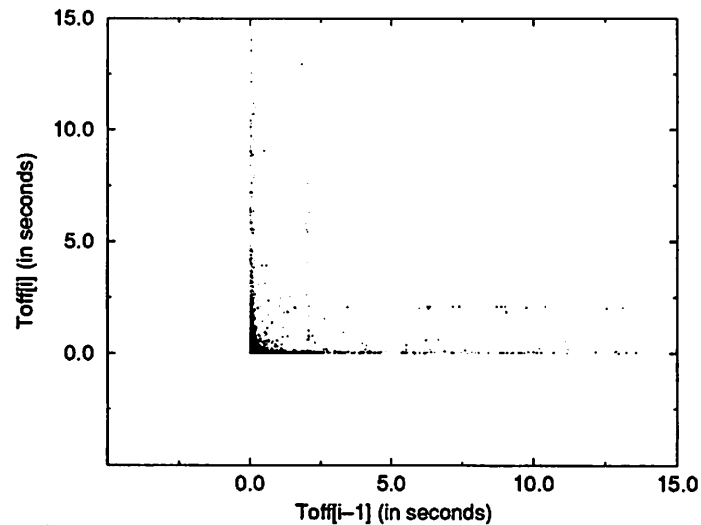


Figure 8-7 : $T_{off}[i]$ versus $T_{off}[i-1]$ Scatter Plot

If the value of $T_{off}[i]$ predicted by the above model is $\geq T_{cost}$, a decision is made to shut the pro-

cessor down.

- The second approach is based on an even simpler intuition - $T_{on}[i]$ corresponds to the most recent history of $T_{off}[i]$, and the scatter plot of $T_{off}[i]$ versus $T_{on}[i]$ in Figure 8-6 is a *L-shaped* plot with the points concentrated along the two axes. This suggests that a large value of $T_{on}[i]$ is followed by a small $T_{off}[i]$ with a very high probability, and that the $T_{off}[i]$ following a small value of $T_{on}[i]$ is fairly evenly distributed. This suggests the following simple filtering (or thresholding) rule as the prediction heuristic:

$$T_{off}[i] \geq T_{cost} \Leftrightarrow T_{on}[i] \leq T_{on_threshold}$$

and, Figure 8-6 suggests that for $T_{cost} = 10$ ms, a reasonable value of $T_{on_threshold}$ is in the range of 10 ms to 15 ms. Note that $T_{cost} = 10$ ms is a very safe upper bound on the shutdown cost if the state is being saved in the main memory - in reality it is more likely to be x10-x100 smaller.

Hit Ratios for Prediction Schemes

The above heuristic rules to predict whether $T_{off}[i] \geq T_{cost}$ are similar in nature to the replacement rules in caches. A good idea of their efficacy can therefore be obtained by measuring the probability with which our prediction is correct, i.e. the *hit ratio*. We used the experimentally obtained traces to simulate the X server running with predictive shut down. Figure 8-8 shows the hit ratio for three schemes: the *model-based* prediction, the *on-threshold-based* prediction with $T_{on_threshold} = 15$ ms, and the *on-threshold-based* prediction with $T_{on_threshold} = \infty$. The last case corresponds to a scheme where we always decide to shut down. The hit-ratio curve corresponding to the *model-based* prediction shows a sudden jump at around $T_{cost} = 160$ ms. This is an artifact of the quadratic model. Finally, note that the hit ratio for the *on-threshold-based* prediction scheme is relatively insensitive to $T_{on_threshold}$ changing from 15 ms to ∞ (the two curves almost overlap), whereas it does degrade with increase in T_{cost} .

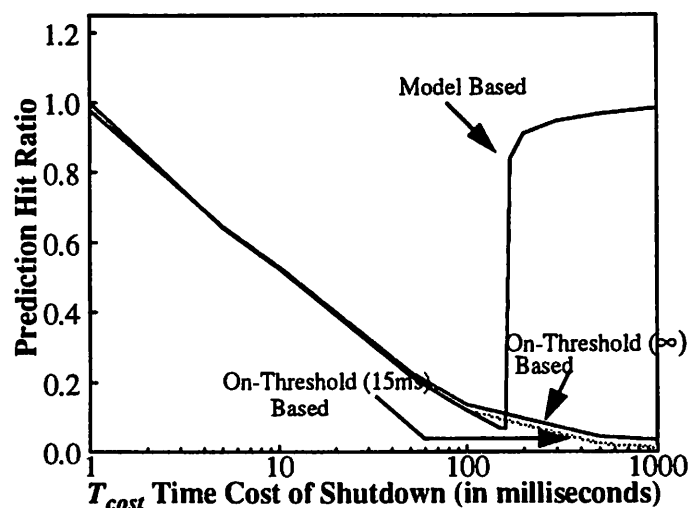


Figure 8-8 : Hit ratio curves.

Impact on Responsiveness - *Slowdown*

The time overhead T_{cost} associated with shutdown can have a negative impact on the responsiveness of the computer due to increased latency. This *slowdown* is difficult to quantify as it involves ill-defined psychological and biological metrics - in fact some studies even suggest second order effects in interactive behavior such as an increase in the user think time as the computer responsiveness degrades [Brady86]. In the absence of any well-defined metric, we used our own simple and intuitive measure of slowdown - it is the factor by which the total length of the X server session is increased due to shutdown being used. This increase occurs because many of the $T_{off/i}$'s are longer than they would have been without shutdown because of the overhead T_{cost} associated with shutdown. Figure 8-9 plots the slowdown resulting from the various schemes for different values of T_{cost} . For comparison we also plot the slowdown resulting from using a conventional *idle-time-threshold* based scheme with a threshold of 2 seconds, similar to Apple's Power Cycling scheme.

As the curves demonstrate, the conventional schemes have almost no slowdown for all values of T_{cost} . However, the *on-threshold-based* prediction scheme performs reasonably well for values of T_{cost} smaller than 10-15 ms - the slowdown is less than 3 % to 4 % which is not noticeable at all.

We verified this by creating a special version of the X11R5 server where an artificial delay, corresponding to T_{cost} , was added on entry to the *off* state. In fact, even a T_{cost} of 100 ms, which corresponds to a slowdown of 50% to 60% for *on-threshold-based* prediction scheme, resulted in a very usable, though noticeably sluggish, X server on the SPARCstation 2. As we mentioned earlier, T_{cost} corresponding to saving state in the main memory and restoring state from the main memory is typically going to range from just a few microseconds for processors with ability to stop the clock, to a few hundreds of microseconds or a few milliseconds for other processors - and for these range of values of T_{cost} the slowdown is negligible for the predictive schemes.

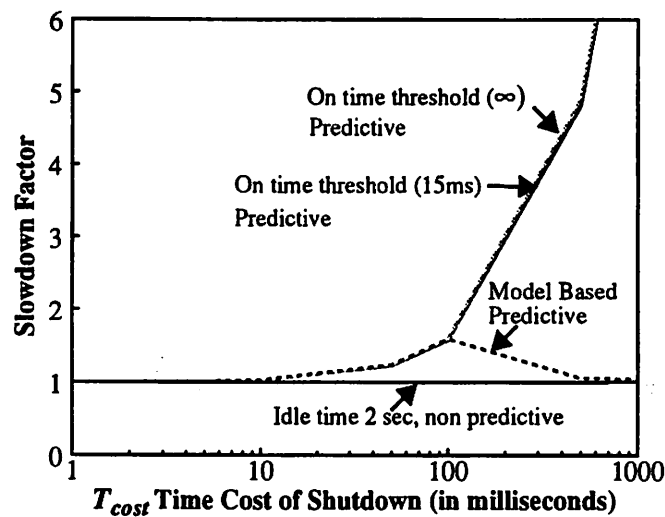


Figure 8-9 : *Slowdown* as a function of T_{cost}

Reduction in Computational Energy

Finally, we evaluated the various predictive, non-predictive, and ideal shutdown schemes from the point of view of reduction in computational energy. The results are summarized in Table 8-2 for three values of T_{cost} - 0 ms, 10 ms and 100 ms - the former being the ideal case, and the latter two being safe upper bounds for storing the processor state in main memory and on disk respectively. The results demonstrate that for $T_{cost} = 10$ ms, a conservative upper bound for most processors, the predictive schemes have much superior energy savings at negligible slowdown. However, for $T_{cost} = 100$ ms, which may correspond to the case where the processor state is saved on the disk, the

		Energy Reduction	% Slowdown
$T_{cost} = 0(\text{ideal})$	Known $T_{off[i]}$	x 38.4	0 %
	Non-Predictive Idle-Time Threshold (2 seconds)	x 1.7	0 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = 15$ ms)	x 20.1	0 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = \infty$)	x 38.4	0 %
	Model-Based Predictive Scheme	x 38.4	0 %
$T_{cost} = 10$ ms	Known $T_{off[i]}$	x 25.7	0 %
	Non-Predictive Idle-Time Threshold (2 seconds)	x 1.7	0 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = 15$ ms)	x 20.1	2.7 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = \infty$)	x 38.4	2.8 %
	Model-Based Predictive Scheme	x 38.4	2.8 %
$T_{cost} = 100$ ms	Known $T_{off[i]}$	x 5.1	0 %
	Non-Predictive Idle-Time Threshold (2 seconds)	x 1.7	.025 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = 1$ ms)	x 2.1	21.1 %
	On-Threshold-Based Predictive Scheme ($T_{on_threshold} = \infty$)	x 38.4	59 %
	Model-Based Predictive Scheme	x 38.4	59 %

Table 8-2 : Summary of Energy Reduction

predictive schemes have noticeable degradation of interactive performance.

8.3 Architecture-Driven Voltage Reduction

Not all programmable computation, even on user-interaction dominated portable devices, is event-oriented - many functions that are not event-driven and instead execute continuously also require implementation as software running on microprocessors or embedded core processors on an ASIC. In fact embedded software for DSP core processors on an ASIC has emerged as the dominant method of implementing speech coding, speech compression, text-to-speech synthesis, simple speech recognition, modem functionality, and many other signal processing functions in portable devices such as cellular phones, PDAs etc. Given the “continuous” nature of signal

processing functions, shutdown strategies such as the predictive shutdown technique of the previous section are not effective. The architecture driven voltage reduction techniques presented in Chapter 3 is however very effective for such applications. Unfortunately, as described earlier and as shown in 8-10(a), operation at reduced supply voltage comes at the expense of increased gate delays, and consequently slower clock frequency for a given circuit. However, due to certain nice attributes present in most DSP algorithms - their performance is determined by throughput alone and not latency, and their ample inherent concurrency - often make it possible for one to use architectural techniques such as parallelism and pipelining to increase the computation throughput which can then be traded-off against the loss in speed due to voltage reduction for a net gain in energy efficiency for unchanged throughput.

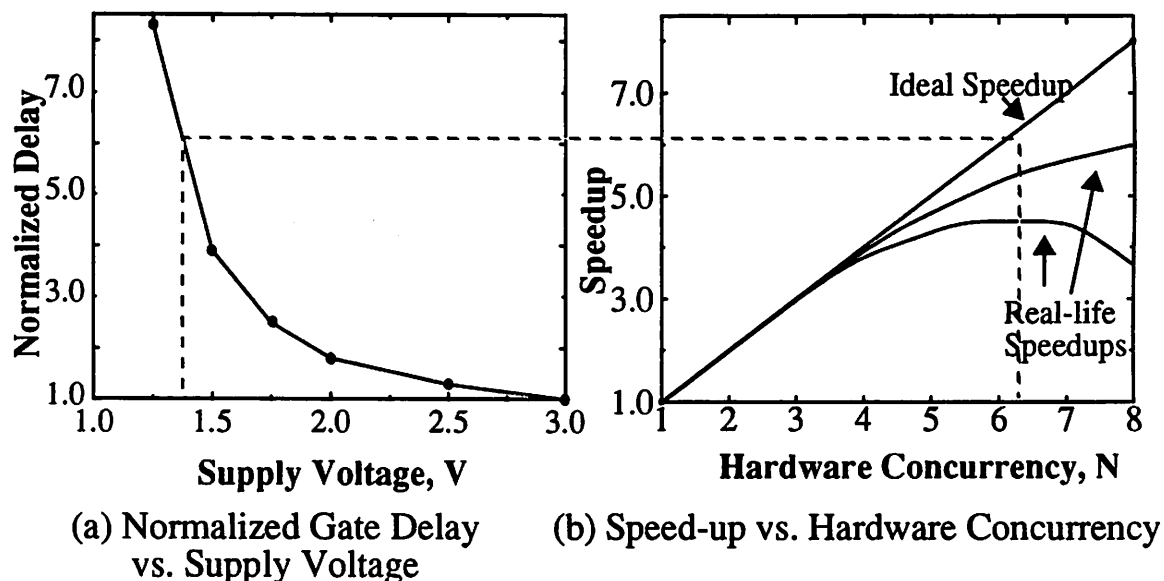


Figure 8-10 : Trade-off between voltage and hardware-concurrency.

Previous chapters have described techniques for energy efficient computation in ASICs that use dedicated architectures for implementing DSP algorithms through aggressive voltage scaling (down to 1-1.5 Volts) and concurrency exploitation; here architecture-driven voltage reduction will be applied to software programmable computation. While the supply voltages of many microprocessors have indeed come down to 3.3V or 3V from 5V, the choice of 3.3V has been

driven by the fact that for this level of voltage scaling the degradation in system performance (clock rate) is not very significant. Architecture-driven voltage reduction, on the other hand, reduces voltage even further into the realm where there is a significant reduction in clock rate, but uses architectural concurrency to keep the system throughput at its original level.

8.3.1 Voltage-Concurrency Trade-Off and Architectural Bottlenecks

As described in Chapter 3, the basic idea in architecture-driven voltage reduction is to compensate for the loss of speed due to increased gate delay when operating at a lower voltage, with the increase in speed due to increased hardware concurrency. The power consumption is reduced at a fixed computation speed (throughput). Hardware concurrency can be due to parallelism or pipelining or a mix of the two. For example, hardware that performs N operations in parallel has concurrency of N . Similarly, a hardware with N pipeline stages so that N successive operations are overlapped also has a concurrency of N . Compiler transformations such as parallelization and software pipelining play an important role in restructuring the computation so that the hardware concurrency is fully exploited.

Figure 8-10 illustrates this trade-off between voltage and hardware concurrency by plotting the curve for increase in gate delay (normalized to gate delay at 3.0V) next to hypothetical curves for increase in throughput (speedup) due to hardware concurrency, and using the same scale for normalized gate delay and speedup. With a hardware concurrency of 6 - for example by using a six processor parallel computer - an ideal speedup of $\times 6$ can be obtained as shown by the plot in 8-10b. If the voltage is now reduced from 3.0V to a level where the gate delay increases by $\times 6$, the clock frequency will have to be reduced by $\times 6$ as well. Each of the processor will therefore slow down by $\times 6$ so that the net speedup with the six processor machine operating at the reduced voltage will be 1 compared to the uniprocessor machine operating at 3.0V. In general, the strategy would be to reduce the voltage to a level where the normalized gate delay increases by the same

factor as the speedup. The throughput of the concurrent hardware operating at the reduced voltage level will then be the same as the throughput of the non-concurrent hardware operating at the original voltage of 3.0V. For our example the reduced voltage level at which the gate delay increases by x6 is 1.3V, as the dotted lines in Figure 8-10 show. Assuming that the switched capacitance for the six processor machine was x6 higher than for the uniprocessor machine, the power is reduced by a factor of $(1/6)*(3/1.3)^2*(6) = 5.3$.

Consider a more detailed analysis. Let $S(N)$ be the speedup for a hardware concurrency of N from 8-10(b). Obviously, for non-concurrent hardware $N=1$, and $S(1)=1$. Let $V(d)$ be the voltage for a normalized gate delay in Figure 8-10(a), with $V(1)=3.0V$ being the reference point. The initial hardware has $N=1$, $S(1)=1$, and $V(1)=3.0V$. Let $f(1)$ and $f(N)$ be the clock frequencies for the initial and the final hardware, let $C(1)$ and $C(N)$ be the switched capacitances, and let $P(1)$ and $P(N)$ be the power consumptions.

If the hardware concurrency is due to parallelism, then one can operate each of the N parallel hardware units at a frequency $f(N)=f(1)/S(N)$ so that the effective throughput is unchanged. Further, parallel hardware units will require some overhead circuit so that the total capacitance can be expressed as $C(N)=(N+\epsilon(N))*C(1)$ where $\epsilon(N)$ is the overhead capacitance. Assuming unchanged switching activity, it follows that:

$$\frac{P(1)}{P(N)} = \frac{C(1)V(1)^2f(1)}{C(N)V(S(N))^2f(N)} = \frac{S(N)}{N + \epsilon(N)} \times \left(\frac{V(1)}{V(S(N))} \right)^2 \quad (\text{EQ 139})$$

Similarly, if the hardware concurrency is due to N -level pipelining, then the hardware can be operated at a reduced voltage of $V(S(N))$ so that the clock frequency will be $f(N)=N*f(1)/S(N)$, which in turn gives an effective throughput that is unchanged. Let the overhead capacitance due to pipelining be $\epsilon(N)*C(1)/N$, so that the total capacitance is $(1+\epsilon(N)/N)*C(1)$. Assuming unchanged switching activity, the ratio of the power consumption is again given by an expression identical to

that in Equation 139.

An increase in energy efficiency is obtained whenever $P(1)/P(N)$ is > 1 . In the ideal case of linear speedup ($S(N)=N$) and no capacitive overhead ($\epsilon(N)=0$), it is clear from Equation 139 that $P(1)/P(N)$ is indeed > 1 because $V(1) > V(S(N))$. In the ideal case one can get arbitrarily large improvements in energy efficiency by continuing to increase hardware concurrency and decrease voltage until the devices stop working.

However, Equation 139 points to two fundamental architectural bottlenecks that prohibit an arbitrary increase in energy efficiency by reducing voltage. First, the speedup is not linear in most cases - as shown in 8-10(b), real examples typically show a speedup that starts saturating, and even decrease, as N is increased. In parallel hardware this may be due to lack of enough parallelism in the computation, or due to effects like bus contention. In pipelined hardware the speedup may not be linear because of granularity of pipelining, or because of the existence of pipeline interlocks and feedback cycles in the computation. The second architectural bottleneck occurs because $\epsilon(N)$ is not zero in real world. In parallel hardware, capacitive overhead is contributed by data multiplexing/demultiplexing, bus arbitration etc., while in pipelined hardware the capacitive overhead is contributed by pipeline registers.

Together these two architectural bottlenecks - nonlinear speedup $S(N)$ and capacitive overhead $\epsilon(N)$ - place a restriction on the increase in energy efficiency that can be obtained.

8.3.2 Exploiting Concurrency in Programmable Computation

From the preceding discussion it is clear that the energy efficiency of a computation can often be increased by trading voltage reduction with increased concurrency. There are two independent issues in this trade-off: the concurrency that is available in the hardware, and the concurrency that is inherent in the computation algorithm. A key role is played by compiler optimizations and transformations in exposing and mapping the algorithmic concurrency to the hardware

concurrency. This problem has been an important area of research in computer science for many years, but with a different goal - increasing the computation speed. Nevertheless, the results are equally relevant to our goal of energy efficiency.

Unlike computation implemented on ASICs, the computation done in software is relatively general-purpose, and efficient compiler transformations and parallelization techniques to exploit the inherent concurrency are not as easily available except in special cases such as DSP and numerical computation. Applications like windowing systems, database access etc. are control intensive and logically more complex than the dataflow oriented DSP and numerical applications. Besides the obvious potential of pipelining execution of successive instructions, here are three common types of concurrency that are available in software computation:

- **Instruction Level Parallelism**

The fine-grained instruction level parallelism inherent in a code is discovered by a dataflow analysis, either dynamically in hardware, or statically in software. Research [Butler91] has shown that instruction level parallelism in excess of 17 per cycle exists in the SPEC suite, a set of applications that are the *de facto* standard compute-bound benchmarks. Further, with properly balanced hardware, an execution rate of 2 to 5.8 can be sustained. Compiler optimization techniques, such as software pipelining, loop unrolling, and branch prediction, help in enhancing the available instruction level parallelism.

- **Thread Level Parallelism**

Many applications have a natural coarse-grained parallelism that is suitable for mapping as separate thread of executions. A good example of this would be a multi-threaded X server that handles different clients via multiple threads. Even on a uniprocessor a multi-threaded organization can improve speed by reducing unnecessary blocking. When multiple processors are available then the thread level parallelism can be exploited by mapping different threads to different processors. The thread level parallelism is usually specified by the user - compilers that can restructure and/or partition an algorithm into multiple threads do not exist, except in special domains such as DSP [Hoang92]. Unfortunately, very few existing applications are multi-

threaded, although this has started to change with the availability of threads in various operating systems.

- **Process Level Parallelism**

Process or task level parallelism is inherent in the assumption behind multi-tasking operating systems like UNIX. Multiple processes, usually corresponding to different applications and often different users, time-share the CPU. Since processes often have to block for I/O, time-sharing results in improved CPU utilization (computation throughput) at the cost of increased latency for compute-bound tasks.

Complementary to the algorithmic concurrency is the concurrency available in the hardware.

There are three main ways of increasing the concurrency in a software programmable computer:

- **Increase the number of processors**

This is the approach taken by MIMD computers such as Sequent and Sun SPARC-10s. The overhead comes in the form of more complex hardware for sharing data between processors as well as more complicated operating system. This approach can exploit process and thread level parallelism, although the latter will require a more sophisticated OS.

- **Increase the number of functional units**

This is the approach taken by SuperScalar microprocessors and VLIW processors. Of course this comes at an overhead in the form of a more complex instruction issue unit. This approach is meant to exploit instruction level parallelism.

- **Increase the levels of pipelining**

This is the approach taken by SuperPipelined microprocessors. This too comes at an overhead in the form of an increased pipeline bubble. Thus approach is also meant at exploiting instruction level parallelism.

The success of a compiler in utilizing the intrinsic concurrency of an algorithm to exploit the available hardware concurrency is a major determinant in achieving energy efficiency through architecture-driven voltage reduction.

8.3.3 A Power Consumption Model for MIMD

To evaluate the effect of hardware parallelism on power, a model to estimate the power consumption needs to be developed. For this we need to define the following quantities:

Let $S(N)$ = computation speed when the MIMD system has N processors. The speed metric $S(N)$ could represent either throughput or latency. Also, let $V(N)$ = Lowest supply voltage at which we can run the N processors while maintaining the same throughput as with the uniprocessor system. Assume that the uniprocessor is running at a reference voltage of 3V (i.e. $V(1) = 3V$). Given the speedup factor (based on the number of processors and algorithmic constraints), the supply voltage $V(N)$ can be determined from the function shown in 8-10(a) which is obtained by characterizing the semiconductor process. For example if $N = 4$, and if $S(4)=4$, then from 8-10(a), $V(4)$ is approximately 1.5V.

Ignoring static power consumption due to leakage currents, the power consumed by CMOS circuits is given by $C V^2 f$. Therefore the power consumption of a uniprocessor is:

$$P(1) = \left(C_{processor} + C_{interconnect} + C_{memory} \right) V(1)^2 f \quad (\text{EQ 140})$$

Where $C_{processor}$, $C_{interconnect}$ and C_{memory} are determined for a specified set of hardware modules (e.g. DSP32C or 386SL) and technology (PCB interconnect => 3-4 pF/inch routing capacitance). For example, for a DSP32C processor, the average capacitance switched is given by $1.2W / ((5.25)^2 25\text{Mhz}) = 1.75\text{nF}$ (the power of 1.2W, obtained from data sheets, does not include the I/O power). For interconnect, considering both the bus and pin components, and assuming an average switching activity, the capacitance was determined to be 0.15nF.

Now consider extending the model presented in Equation 140 to multiple processors. Ideally, speedup grows linearly with the number of processor; however, due to interprocess communication overhead, the speedup is typically not linear. Similarly, parallelism will introduce capacitance overhead. The overhead is often attributed to an increase in interconnect capacitance

(longer wires on the PC board/MCM), loading capacitance (more processor on the shared bus), control capacitance, interprocess communications overhead (e.g. more memory I/O transactions), etc. Taking these factors into account, the power consumption as a function of the number of processors is given by:

$$P(N) = [NC_{processor} + NC_{interconnect}(N + \%C_{over}) + C_{memory}(N + \%C_{over})] \frac{fV(N)^2}{S(N)} \text{ (EQ 141)}$$

Where $\%C_{over}$ is the interprocessor communications overhead. Note that the physical interconnect capacitance, the second term, increases linearly with the number of processors. This is a reasonable assumption since for PCB interconnect technology, the processors can be placed right next to each other (in a linear fashion) and routing can be performed on multiple layers (unlike ASICs where the routing is confined typically to 2 metal layers).

Note that a minimum bound on the supply voltage, V , can be set by one of the following:

- Noise immunity requirements.
- $S(N)$ stops increasing with N . Using the curve in 8-10(a), a maximum bound on N can be translated into a minimum bound on V .
- There may be a maximum bound on N due to operating system and other software considerations. This too can be translated into a corresponding minimum bound on V .

Intuitively, a model of speedup in MIMD will have the following characteristics:

- For small values of N the speedup will be close to linear.
- At higher values of N , the speedup will start saturating and reach a peak.
- Finally, for even larger values of N , the speedup may actually start falling off. This may happen because, for example, the arbitration overhead for access to a shared bus may just grind the system to a halt.

Such a “hump” like speedup curve is characteristic of shared bus centralized memory systems. For our purposes, there is no rational reason to parallelize beyond the point of saturation. In other

words, we are only interested in the initial part of the curve. One such model of speedup, using analytical results from [Stone87], is given by the following:

$$S(N) = \frac{N}{1 + \frac{N-1}{r}} \quad (\text{EQ 142})$$

where r is a constant. As N gets large, the speedup will saturate to r . The value of r depends on, among other things, the ratio of computation to communication, the granularity and number of interacting tasks, and the communication network.

8.3.4 Example: CORDIC on a MIMD using Thread Level Parallelism

First, we will consider signal processing applications running on programmable processors. For such applications, it is typically throughput (and sometimes latency) which is the design constraint rather than trying to compute as fast as possible. For example, a speech codec has to compress and decompress speech at a sample rate of 8Khz; however, once this throughput requirement is met, there is NO advantage in making the computation faster. As described earlier, by making the computation faster (using more parallelism for example), the supply voltage can be dropped and hence the power can be reduced while still meeting the functional throughput. Typical applications include filters, speech processing, robotics and image processing.

Most examples in this class exhibit a substantial amount of concurrency. For example, all DSP applications are executed in an infinite time loop, giving rise to temporal concurrency. In addition, several exhibit spatial concurrency. For low-power operation (as we will see), it is very important to detect and exploit concurrency since we can run at reduced supply voltages. For example, temporal concurrency can often be exploited by pipelining, resulting in dramatic speedup and hence lower power for a fixed throughput. For application that have a lot of recursion (feedback loops), it often necessary to apply a series of transformations (like loop unrolling) to achieve

speedup. Compilers have been developed to effectively exploit concurrency in DSP applications [Hoang92].

To study the power trade-offs of DSP multi-processor implementations, a cordic algorithm (obtained from [Hoang92]) is studied that converts cartesian to polar coordinates iteratively in 20 steps. It takes as input an (X,Y) coordinate, as well as an array of correction angles. The loop iteratively calculates the corresponding amplitude and phase. Since each iteration is dependent on the results of the previous iteration, the computation is sequential in nature.

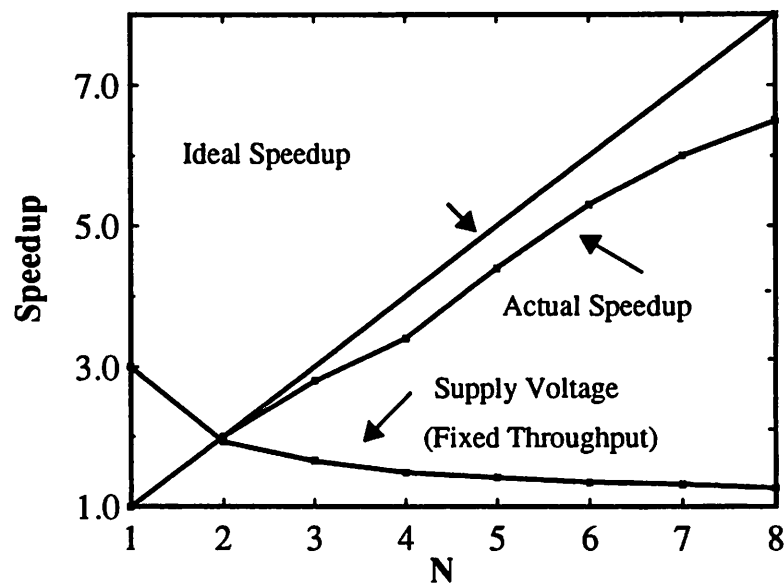


Figure 8-11 : Speedup and Vdd vs. N

A scheduling algorithm which only exploits spatial concurrency would perform poorly on this example. However, optimizing using transformations like pipelining the loop and assigning successive loop iterations to successive processors, significant speedup can be achieved. The analysis is based on the DSP32C processor power numbers. 8-11 shows the ideal speedup (linear with processors) and the actual speedup obtained. 8-11 also shows the lowest supply voltage $V(N)$ at which the various multiprocessor implementations can run while meeting the same functional throughput as the uniprocessor version. Figure 8-12 shows the communications overhead as a function of the number of processors.

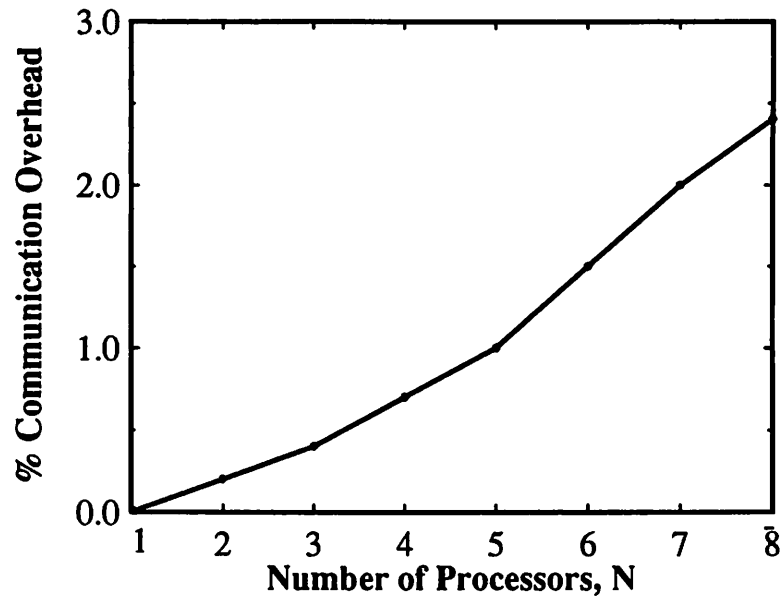


Figure 8-12 : % communications overhead vs. N.

Figure 8-13 shows the power consumption as a function of N. We see that a factor of 3.3 reduction in power can be accomplished by reducing the supply voltage from 3V to 1.5V. The power starts to increase with more than 6 processors since the overhead circuitry (interconnect capacitance) starts to dominate, resulting an optimum number of processors for power.

8.3.5 Low Power Computers - Multiple Slow CPUs More Energy Efficient than a Single Fast CPU?

The discussion in this section shows that if energy efficiency is the consideration, it is better to use concurrent (parallel or pipelined) slower hardware operating at a lower voltage than to use non-concurrent faster hardware at a higher voltage. To put it differently, multiple slower CPUs are more energy efficient than a single fast CPU for applications with enough algorithmic concurrency. This suggests that from a power perspective it may be better for future microprocessor chips to consist of multiple slow and simple CPUs operating at a slow clock frequency and a low voltage - a conclusion which is just the opposite of the current trend towards

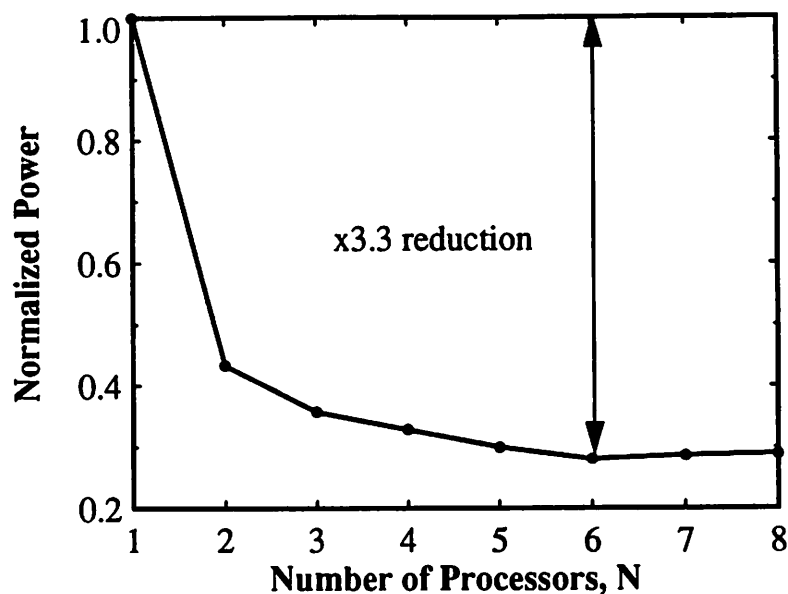


Figure 8-13 : Power vs. N.

complex CPUs operating at extremely high frequencies (many 100s of MHz).

8.4 Summary

Two architectural approaches for energy efficient programmable computation were described: predictive shutdown, and concurrency driven supply voltage reduction. A significant reduction in power consumption can be obtained by employing these architectural techniques. For example, parallel processors operating at a reduced voltage can substantially improve the energy efficiency of a fixed throughput computation (by a factor of 4 in the example we described). For applications where continuous computation is not being performed (like X-server, etc.), an aggressive shut down strategy based on a predictive technique has been presented which can reduce the power consumption by a large factor compared to the straightforward conventional schemes where the power down decision is based solely on a predetermined idle time threshold.

CHAPTER 9

Conclusions and Future Directions

Portable operation is becoming increasingly important for many of the most exciting new electronic products. The strict limitation on power dissipation which this imposes, must be met by the designer while still meeting ever higher computational requirements. To meet this need, a comprehensive strategy is required at all levels of the system design, ranging from algorithms, through architectures and logic styles to the underlying technology. An integrated methodology and the associated CAD tools for low power CMOS design has been presented here which incorporates optimization all levels of system design.

For a properly designed CMOS gate, power is consumed primarily in charging and discharging parasitic capacitors and is given by $\alpha C_L V_{dd}^2 f$ where α is node transition activity, C_L is physical load capacitance, V_{dd} is supply voltage and f is operating frequency.

The choice of supply voltage clearly has the greatest impact on the power-delay product, which is the amount of energy required to perform a given function. It is only necessary to reduce the supply voltage to quadratically improve the power-delay product. Conventional low-power

approaches have exploited the velocity saturated nature of sub-micron devices to apply a limited amount of voltage scaling from 5V down to 3.3V - the "critical" voltage at which there is little degradation in circuit speed relative to 5V. Unfortunately, further reduction in supply voltage is associated with a reduction in the circuit speed, with the gate delays increasing drastically as the voltage approaches the threshold voltages of the devices. However, if the goal is to reduce the power consumption for realizing a fixed functionality (or fixed throughput), then an architectural voltage scaling strategy, that involves compensating the increased gate delays at reduced voltages with increased level parallelism, should be used to scale the power supply voltage down to 1-1.5V (for a standard CMOS technology) while maintaining system throughput. Not only can such an approach that trades silicon area for lower power consumption be applied for signal processing applications and other dedicated applications that have a fixed computational rate, but also for those general purpose applications where the goal is to increase the MIPS/Watt for a fixed MIPS level. Therefore, future power conscious microprocessor chips should use multiple slow and simple CPUs operating at a slow clock frequency and a low voltage - a conclusion which is just the opposite of the current trend towards complex CPUs operating at extremely high frequencies.

To scale power supply voltages beyond the 1-1.5V scaling presented in this work, it is necessary to increase the leakage component of power by scaling the device threshold voltages to levels that are much lower than the values currently used which typically range from 0.7V to 1V. Device threshold voltages should be scaled to less than 300mV, so as to make the subthreshold leakage power and switching power equal. The use of low V_t devices will increase the overall power consumption for circuitry that is powered down using gated clocks since sub-threshold current will continue to flow even though no switching power is consumed. Special circuitry will be required to dynamically adjust the leakage power for such applications - that is, increase the leakage when active and decrease it when idle. One approach is to dynamically adjust the substrate bias to modulate the threshold voltages. Another approach is to use a process with a high V_t and a low V_t and use the high V_t devices to turn off devices when idle to eliminate leakage power.

High efficiency DC/DC conversion is an integral part of low-power system design. The freedom to scale the power supply voltages to arbitrary levels depends strongly on the ability to provide high efficiency power supply conversion at arbitrary levels under varying loads. Also, adaptive power supply circuits which minimize power by dynamically changing the voltage levels for applications that require processing of bursty input data. Also critical to the design of multi-voltage systems, in which different modules are optimized to operate at their own “optimal” supply voltage, is the use of level conversion circuits.

Besides architecture driven voltage scaling, another approach to low power design is to reduce the switching activity to the minimal level required to implement a given function. To reduce the average number of transitions per operation to the minimal possible level, it is necessary to exploit the temporal and spatial correlations present in the data being processed while optimizing at all levels of the system design ranging from the algorithms and architectures used to the logic, circuit and physical design. When minimizing transition activity, it is not enough to just reduce the number of transitions due to the static behavior of the circuit that is determined strictly from the boolean logic but also those transitions that occur due to the dynamic nature of the circuit.

The choice of algorithms has the greatest impact on minimizing total switched capacitance. Just by redefining the approach to solve a problem, the computational complexity (# of operations to perform a given function) can be significantly altered. For example, conventional approaches to video compression have used standards such as MPEG or JPEG that are based on the Discrete Cosine Transform and utilize complex encoding and decoding strategies. These standards are clearly not designed for low-power hardware implementation due to the computational complexity. For example, if only video decompression has to be performed on a portable device, then by choosing the VQ decompression algorithm (i.e redefining the way to perform video decompression), the complexity is significantly reduced resulting in more than an order of magnitude reduction in switched energy.

At the algorithmic level, it is also very important to optimize the data representation (binary vs. gray-coding, two's complement vs. sign-magnitude, etc.). Data coding has been used in communication systems to control the statistics of the transmitted data symbols with the goal of removing undesired correlation among information bits (scrambling, e.g. for encryption) or introducing controlled correlation (redundancy, e.g. for spectrum shaping of transmitted signal, timing recovery, error correction in unreliable channels). Coding for reduced switching activity falls under the second category - to introduce controlled correlation such that the total number of bit transitions is reduced. Interestingly, the use of coding to reduce the switching activity is just the opposite of the use of some types of coding in communication channels to ensure that the transmitted signal has enough transition activity so that timing recovery and other functions can be reliably performed.

Next to algorithmic optimization, architecture optimization has the greatest impact on the switching activity. A key conclusion on the architectures that minimize transition activity is that time-multiplexing of hardware can destroy naturally occurring temporal correlations that exist in data. Therefore resource sharing should be avoided as much as possible. When the amount of computational resources is limited, then an optimized assignment and scheduling must be used which keep uncorrelated data on different hardware units. The sequencing of operations should be optimized to reduce the switching activity by altering the dynamic range. Race free topologies should be used to minimize spurious transitions that occur due to timing skew. Self-timed circuits are useful only when the overhead is minimal, such as is the case for memory circuits.

At the logic and circuit level, gated clocks should be carefully used to eliminate wasteful logic and bus transitions. At the physical design level, the place and route should be optimized such that signals that have high switching activity can be assigned short wires while signals that have low switching activity can be allowed to have long wires. This is different from current approaches which typically use placement schemes to minimize area.

Clearly, design tools which encapsulate the above methodologies are needed. The techniques presented to reduce power consumption are by no means orthogonal, making manual optimization very difficult. For example, as discussed in Chapter 5, an optimization that reduces the number of operations and therefore switched capacitance can increase the critical path dictating a higher voltage. Therefore, high-level design tools are required to automatically find computational structures that result in the lowest power consumption. Such design tools will require fast estimation of power consumption for a given topology from a high-level of abstraction, which allows the exploration of several structures without having to map high-level descriptions to low-level implementations. The only viable approach to estimate interconnect and control without mapping a high-level design to layout, is to develop statistical models based on a representative benchmark set. The efficient application of high-level optimization techniques requires an “intelligent” ordering of optimizing transformations (an efficient search strategy) and it was shown that a combination of heuristic and probabilistic strategy produces efficient solutions. The synthesis system developed here was used to optimize power for several design examples and the experimental results indicate that more than an order of magnitude reduction in power is possible over current-day design methodologies while maintaining the system throughput.

The various low-power techniques developed have been demonstrated in the design of chipset for a portable multimedia terminals that allows users to have untethered access to multimedia information servers. Six chips were designed to provide the interface between a high speed digital radio modem and a commercial speech codec, pen input circuitry, and LCD panels for text/graphics and full-motion video display. The chips provide protocol conversion, synchronization, error correction, packetization, buffering, video decompression, and D/A conversion at a total power consumption of less than 5 mW - which is three orders of magnitude lower than existing systems. This power reduction was possible due to the system level approach used that involved system partitioning, minimizing the number of operation, choice of data representation, architecture driven voltage scaling, activity driven hardware assignment, use of gated clock,

optimized cell-library, and activity driven place and route.

9.1 Future Directions

9.1.1 Design Methodologies

We have demonstrated circuitry for multimedia applications that works at power supply voltages as low as 1.1V using a standard process that has a worst case threshold voltage of 1V. This is by no means a lower bound on the operating power supply voltage. In fact, inverter circuits have been demonstrated in 1974 that operate from a 200mV power supply voltage [Swanson72]. The lower bound on power supply voltage needs to be investigated for implementation of large and *high* throughput digital systems by scaling the threshold voltages of the devices. Threshold voltage scaling will also require new circuit and power management techniques. DSP circuits that operate continuously, can tolerate a fixed low threshold voltage, but circuitry that is frequently powered down such as the multiply unit of a microprocessor cannot tolerate a fixed low threshold voltage. Adaptive bias circuitry to turn off devices when idle need to be developed. Recently, some work has been done in this area for small design examples, but this problem requires further study and demonstration on larger system designs. Other techniques such as gating the power supply with high threshold devices should also be investigated.

Future systems will use multiple internal voltages and therefore will require high efficiency on-chip DC/DC conversion circuits. These circuits will be required to provide adaptive digitally controlled output voltage levels, that are much lower than the lowest published 1.5V DC/DC converter, that range from 200mV to 1V while maintaining efficiencies greater than 95%. Since the proposed design methodology enables power levels that are in the 10's of mW range for complex signal processing, the power consumption of the converter control circuitry, should be kept in the μ W range - orders of magnitude lower than current approaches.

Circuit approaches that utilize low-swing logic and transmission need to be developed to reduce power beyond voltage scaling. The lower limit on the amount of swing required to distinguish logic levels needs to be investigated and the trade-off between increased sense amplifier power and reduced logic power needs to be analyzed.

There are enormous opportunities for further research in minimizing the overall switched capacitance. This includes the extensive use of dynamic power management techniques, the analysis of logic families implemented in different technology (low V_t), optimizing data representation and bus coding (mult-level logic, sign-magnitude for programmable processor implementation), carry propagation free structures (e.g carry-save or RNS), optimizing control structures, optimized memory architectures, and exploitation of signal correlations.

9.1.2 CAD Tools for Portable System Design

Portable systems are composed of a mixture of computational engines (including dedicated custom hardware, software running on programmable hardware, and other reconfigurable components), I/O drivers (A/D and D/A converters) and I/O devices (including LCD display, sensors, pen input, speaker/microphone, radio), and power supplies.

There is a need for a CAD system that enables a systematic system approach for the fast and efficient design of portable devices. A system level design environment, SIERRA, has been recently developed but does not consider the constraints of portability [Srivastava92]. The ideas developed in SIERRA can be extended to include power constraints. A low-power system design environment will involve the development of high-level estimation tools that allow for an efficient exploration of the design space for implementation onto multiple targets given various design constraints on power, throughput, area, etc., the development of a CAD system for the synthesis of low-power custom ASICs, efficient module generators that exploit reuse of existing modules and efficient implementation computation on programmable hardware. Some specific tools that will be developed as an extension to the existing system will include:

-
- Development of a CAD system for the estimation and optimization of power consumption for signal processing systems at all levels of system design. This will include architectural and algorithmic transformations, data coding, system and chip level partitioning, assignment/scheduling, module selection, logic design (minimization and gated clock), and physical design (activity driven place and route, transistor sizing, etc.). The CAD system described in Chapter 5 did not exploit signal correlation to reduce power. The inclusion of signal correlations into both estimation and optimization tools is essential for maximum power reduction.
 - Parameterized module generators for low-powered portable systems: e.g generation of high efficiency, DC-DC converters for different supply voltages and power requirements, generation of interface circuitry such as A/D and D/A converters for different display resolution and drive capabilities.
 - The development of a compiler that targets power minimization - for both general purpose processors and programmable DSP processors.

9.1.3 New Applications

Thus far, increasing the clock rate has been the primary design metric; however, the increasing demand for portable computing has rapidly elevated the minimization of power consumption to be an equally or more important design parameter. It is important to continue developing new design methodologies and implementing them on new applications. For example, one of the short term goals is to extend the InfoPad terminal capabilities to include up-link color video from the terminal to the base station which will enable video conferencing. In addition, the two separate displays of the current terminal for video and graphics will be replaced by a single low-power display for color video and color text/graphics. The goal is to apply several of the techniques developed in this work as well as new techniques on a low-power chipset that supports video conferencing; this chipset will include circuitry to digitize analog video data from a CCD imager, perform compression and error correction, and video decompression. Also, on the downlink, a raster-operation processor will take graphics commands and display color text/graphics on the same

LCD as the video.

In the long term, other applications such as battery operated bio-medical implantable devices, low-power remote sensors, low-power programmable computation, and low-power I/O devices such as LCD displays and CCD imagers will be considered.

Bibliography

- [Aho77] A. V. Aho, J.D. Ullman, Principles of compiler Design, Reading, Addison-Wesley, 1977.
- [Algoritmica91] Algoritmica, Special Issue, Simulated Annealing, A. Sangiovanni-Vincentelli, ed. Vol. 6, No. 3, pp. 295-478, 1991.
- [Alidina94] M. Alidina, J. Monteiro, A. Ghosh, M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power", 1994 International Workshop on Low-power Design, April 1994, pp. 57-62.
- [Apple92] Apple Computer Inc., "Power Manager IC", and "Reduced Power Modes", in Chapter 20 of *Technical Introduction to the Macintosh Family*, 2nd Edition, Addison Wesley, October 1992.
- [Bakoglu90] H.B. Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley, Menlo Park, CA, 1990.
- [Bell91] T. Bell, "Incredible shrinking computers", IEEE Spectrum, pp. 37-43, May 1991.
- [Benson90] D. Benson, Y. Bobra, B. McWilliams, et al., "Silicon Multichip Modules", Hot Chips Symposium III, Santa Clara, CA, August 1990.
- [Berkoff93] B. I. Berkoff, "Taking Charge: PowerBook-Battery Management", MacUser, February 1993.
- [Brady86] J. T. Brady, "A Theory of Productivity in the Creative Process", IEEE CG&A, May 1986.
- [Breiman84] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone: "Classification and Regression Trees", Wadsworth & Brooks, Monterey, CA 1984.
- [Brodersen88] R.W. Brodersen et. al., LagerIV Cell Library Documentation, Electronics Research Laboratory, University of California, Berkeley, June 23, 1988.
- [Brodersen93] R. W. Brodersen, A. P. Chandrakasan, S. Sheng, "Design Considerations for Portable Systems", IEEE International Solid-state Circuits Conference, pp. 168-169, February 1993.
- [Burd93] T. Burd, InfoPad Documentation, 1993.
- [Burd94] T. Burd, Low-power Cell Library, M.S. Thesis, U.C. Berkeley, June 1994.
- [Burghardt94] Frederick L. Burghardt, Architecture and Implementation of the InfoPad Network Prototype, Masters Report, ERL UCB EECS, June 16, 1994.
- [Burr91] J. B. Burr and A. M. Peterson, "Energy Considerations in Multichip Module-based Multipro-

- cessors", ICCD91, pp. 593-600.
- [Burstein92] A. Burstein, A. Stoelze, R.W. Brodersen, "Using Speech Recognition in a Personal Communications System," in Proc. 1992 International Communications Conf., Chicago, IL, June 1992.
- [Burstein93] A. Burstein, Low-power Cell-library Documentation, 1993.
- [Butler91] M. Butler, T. Yeh, Y. Patt, et. al., "Single Stream Parallelism is Greater than Two", Proceedings of the 18th International Symposium on Computer Architecture, May, 1991.
- [Callaway92] T. Callaway and E. Swartzlander, Jr., "Optimizing Arithmetic Elements for Signal Processing", VLSI Signal Processing, V, pp. 91-100, IEEE Special Publications, 1992.
- [Chandrakasan92] A. P. Chandrakasan, S. Sheng, R. W. Brodersen, "Low-power digital CMOS design", IEEE Journal of Solid State Circuits, pp. 473-484, April 1992.
- [Chandraksasn94] A. P. Chandrakasan, A. Burstein, R. W. Brodersen, "A Low-power Chipset for Multimedia Applications", IEEE International Solid-state Circuits Conference, pp. 82-83, February 1994.
- [Chao94] K. Chao, D. Wong, "Low-power Considerations in Floorplan Design", 1994 International Workshop on Low-power Design, April 1994, pp.45-50.
- [Chu87] K. Chu and D. Pulfrey, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic", IEEE Journal of Solid-State Circuits, pp. 528-532, August 1987.
- [Cirit87] M. Cirit, "Estimating Dynamic Power Consumption of CMOS Circuits", IEEE International Conference on Computer Aided Design, pp. 534-537, 1987.
- [Clarke85] R.J. Clarke, *Transform Coding of Images*, Academic Press, NY, 1985.
- [Comsat82] Comsat General Integrated Systems, Super-Filsyn Users Manual, March 1982.
- [Dahle91] D. Dahle, "Designing High Performance Systems to Run from 3.3V or Lower Sources", Silicon Valley Personal Computer Conference, pp. 685-691, 1991.
- [Davari88] B. Davari *et al.*, "A High Performance 0.25 μ m CMOS Technology", IEDM, pp. 56-59, 1988.
- [Deng94] A. Deng, "Power Analysis for CMOS/BiCMOS Circuits", 1994 International Workshop on Low-power Design, April 1994, pp. 3-8.
- [Eager91] J. Eager, "Advances in Rechargeable Batteries Pace Portable Computer Growth", Silicon Valley Personal Computer Conference, pp. 693-697, 1991.
- [Fang92] W.C. Fang, C.Y. Chang, B.J. Sheu, "A Systolic Tree-Searched Vector Quantizer for Real-Time Image Compression," *VLSI signal processing IV*, New York: IEEE Press, 1990.
- [Feig90] E. Feig, "On the multiplicative complexity of the Discrete Cosine transform", 1990 SPIE/SPSE symposium of Electronic Imaging Science and Technology, Santa Clara, CA, February, 1990.
- [Fournier91] J.M. Fournier and P. Sena, "A 130MHz 8-b CMOS Video DAC for HDTV Applications", IEEE Journal of Solid-State Circuits, pp. 1073-1077, July 1991.
- [Garey79] M. Garey and D. Johnson, *Computers and Intractability*, Freeman and company, 1979.
- [Gersho92] A. Gersho, R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.
- [Ghosh92] A. Ghosh, S. Devedas, K. Keutzer, and J. White, "Estimation of Average Switching Activity", Proc. DAC, 1992.
- [Glasser85] L.A. Glasser and D.W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley Publishing Co., Reading, Mass., 1985.
- [Goertzel68] G. Goertzel, "An algorithm for the Evaluation of Finite Trigonometric Series", *Amer. Math. Monthly*, Vol. 65, No. 1, pp. 34-35, 1968.
- [Goldberg91] D. Goldberg, "What every computer scientist should know about floating-point arithmetic", *ACM Computing Surveys*, Vol. 23, No. 1, pp. 5-48, 1991.
- [Gray84] P. Gray, R. Meyer, *Analysis and Design of Analog Integrated Circuits*, Second edition, John Wiley & Sons, 1984.

-
- [Haroun89] B.S. Haroun, M.I. Elmasry, "Architectural Synthesis for DSP Silicon Compilers", IEEE Transaction on CAD for IC, Vol. 8, No. 4, pp. 431-447, 1989.
- [Hartley89] R. Hartley, A. Casavant, "Tree-height Minimization in Pipelined Architectures", IEEE IC CAD, pp.112-115, 1989.
- [Heller77] W.R. Heller et al. "Prediction of wiring space requirements for LSI", IEEE/ACM Proc. 14th Design Automation Conference, pp. 20-22, 1977.
- [Hirendu93] V. Hirendu and M. Pedram, "PCUBE: A Performance Driven Placement Algorithm for Lower Power Designs", Proc. of Euro-DAC '93, pp. 72-77.
- [Hoang92] P. D. Hoang, "Compiling Real-Time Digital Signal Processing Applications onto Multiprocessor Systems", Ph. D. Thesis, EECS Department, University of California, Berkeley, June 1992.
- [Hodges88] D. Hodges and H. Jackson, Analysis and Design of Digital Integrated Circuits, McGraw-Hill, Inc., 1988.
- [Jacobs90] G.Jacobs and R.W. Brodersen, "A Fully Asynchronous Digital Signal Processor Using Self-Timed Circuits", IEEE Journal of Solid-State Circuits, pp. 1526-1537, December 1990.
- [Jain85] R. Jain, J. Vandewalle, and H. J. DeMan, "Efficient and accurate Multiparameter Analysis of Linear Digital Filters Using a Multivariable Feedback Representation", IEEE Transaction on CAS, CAS-32:225-235, March 1985.
- [Kakumu90] M. Kakumu and M Kinugawa, "Power-Supply Voltage Impact on Circuit Performance for Half and Lower Submicrometer CMOS LSI", IEEE Transactions on Electron Devices, Vol 37, No. 8, pp. 1902-1908, August 1990.
- [Karp72] R. Karp, "Reducibility among combinatorial problems", in R.E. Miller and J. W. Thatcher, Complexity of Computer Computations, Plenum Press, New York, pp. 85-103.
- [Kimura91] N. Kimura, J. Tsujimoto, "Calculation of Total Dynamic Current of VLSI Using a Switch Level Timing Simulator (RSIM-FX)", CICC, 1991.
- [Kingsbury94] B. Kingsbury and J. Wawrzynek, Personal Communications.
- [Kurdahi89] F.J. Kurdahi, A.C. Parker: "Techniques and Area Estimation of VLSI Layouts", IEEE Trans. on CAD, Vol. 8, No. 1, pp. 81-92, Jan 1989.
- [Landman93] P. E. Landman, J. M. Rabaey, "Power Estimation for High Level Synthesis," Proceedings of EDAC '93, Paris, Feb. 1993, pp. 361-366.
- [Leiserson91] C.E. Leiserson, J.B. Saxe, "Retiming Synchronous Circuitry", Algorithmica, Vol. 6., No. 1, pp. 5-35, 1991.
- [Messerschmitt88] D. Messerschmitt, "Breaking the Recursive Bottleneck", 1988, in Performance Limits in Communication Theory and Practice, 1988, J.K. Skwirzynski, pp. 3-19.
- [Miki86] T. Miki, Y. Nakamura, M. Nakaya, S. Asai, Y. Akasaka, Y. Horiba, "An 80-MHz 8-bit CMOS D/A Converter", IEEE Journal of Solid-state Circuits, pp. 983-988, Dec. 1986.
- [Miki92] T. Miki, Y. Nakamura, Y. Nishikawa, K. Okada and Y. Horiba, "A 10bit 50MS/s CMOS D/A Converter with 2.7V Power Supply", 1992 Symposium on VLSI Circuits, pp. 92-93.
- [Monterio93] J. Monteiro, S. Devadas, A. Ghosh, "Retiming Sequential Circuits for Low Power", IEEE International Conference on Computer-Aided Design, 1993, pp. 398-402.
- [Nadel93] B. Nadel. "The Green Machine", PC Magazine, vol 12, no. 10, pp. 110, May 25, 1993.
- [Najm90] F. Najm, "Transition Density, A Stochastic Measure of Activities in Digital Circuits", DAC, pp. 644-649, 1991.
- [Narayanaswamy93] S. Narayanaswamy, InfoPad Documentation, 1993.
- [Parhi89] K.K. Parhi: "Algorithm Transformation Techniques for Concurrent Processors", Proc. of the IEEE, Vol. 77., No. 12, pp. 1879-1895.
- [Patterson80] D.A. Patterson and C.H. Sequin, "Design Considerations for Single-Chip Computers of the Future", Joint Special Issue, IEEE Journal of Solid-State Circuits SC-15, No.1 and IEEE Transactions on Computers C-29, No.2, pp. 108-116, Feb. 1980.
- [Potkonjak91] M. Potkonjak and J. Rabaey, "Optimizing the Resource Utilization Using Transformations",
-

-
- Proc. IEEE ICCAD Conference, pp. 88-91, November 1991.
- [Potkonjak92] M. Potkonjak, J. Rabaey: "Maximally Fast and Arbitrarily Fast Implementation of Linear Computations", IEEE ICCAD, pp. 304-308.
- [Press88] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling: "Numerical Recipes in C", Cambridge University Press, 1988.
- [Rabaey91a] J. Rabaey, C. Chu, P. Hoang, M. Potkonjak, "Fast Prototyping of Data Path Intensive Architecture", IEEE Design and Test, Vol. 8, No. 2, pp. 40-51, 1991.
- [Rabaey91b] J. Rabaey and M. Potkonjak, "Estimating Implementation Bounds for Real Time Application Specific Circuits", European Solid-State Circuits Conference, Milano, Italy, pp. 201-204, September 11-13, 1991.
- [Rabaey94] J. Rabaey, keynote talk at the Berkeley Industrial Liason Program (ILP), 1994.
- [Rabaey95] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, Englewood Cliffs, N.J., to be published in 1995.
- [Rao90] K.R. Rao and P.Yip, *Discrete Cosine Transform*, Academic Press, 1990.
- [Salz89] A. Salz, M. Horowitz, "IRSIM: An Incremental MOS Switch-level Simulator", Proceedings of the 26th ACM/IEEE Design Automation Conference, June 1989, pp. 173-178.
- [Schaper83] L.W. Schaper and D.I. Amey, "Improved Electrical Performance Required for Future MOS Packaging", IEEE Transaction on Components, Hybrids, and Manufacturing Technology, vol. CHMT-6, pp. 282-289, Sept. 1983.
- [Schultz92] D. Schultz, "The Influence of Hardware Mapping on High-Level Synthesis", U.C. Berkeley M.S. report, ERL M92/54.
- [Shen92] A. Shen, A. Ghosh, S. Devedas, K. Keutzer, "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks", Proc. IEEE ICCAD, pp. 402-407, 1992.
- [Sheng92] S. Sheng, A. P. Chandrakasan, R. W. Brodersen, "A Portable Multimedia Terminal", IEEE Communications Magazine, pp. 64-75, December 1992.
- [Shoji88] M. Shoji, CMOS Digital Circuit Technology, Prentice-Hall, 1988.
- [Shung91] C. S. Shung et. al., "An Integrated CAD System for Algorithm-Specific IC Design.", IEEE Transactions on CAD of Integrated Circuits and Systems, April 1991.
- [Sodini84] C. Sodini, P.K. Ko, and J. L. Moll, "The Effect of High Fields on MOS Device and Circuit Performance Devices", IEEE Trans. on Electron Devices, vol. ED-31, pp. 1386-1393, Oct 1984.
- [Sorkin91] G.B. Sorkin, "Efficient Simulated Annealing on Fractal Energy Landscapes", Algorithmica, Vol. 6, No. 3, pp. 367-418, 1991.
- [Srivastava94] M. Srivastava, A. Chandrakasan, R. Brodersen, "Energy efficient programmable Computation", submitted to the IEEE Journal of VLSI Systems.
- [Stan94] M. Stan and W. Burleson, "Limited-weight Codes for Low-power I/O", 1994 International Workshop on Low-power Design, April 1994, pp. 209-214.
- [Stone87] Harold S. Stone, "Multiprocessor Performance", in Chapter 6 of *High-Performance Computer Architecture*, Addison Wesley, 1987.
- [Stratakos94] A. Stratakos, S. Sanders, R. Brodersen, "A Low-voltage CMOS DC-DC Converter for a Portable Low-Powered Battery-Operated System," PESC 1994.
- [Su94] C. Su, C Tsui, and A. Despain, "Low-power Architecture Design and Compilation Techniques for High-Performance Processors", pp. 489-498, Compcon 1994.
- [Svensson94] L. "J." Svensson, J. G. Koller, "Adiabatic Charging Without Inductors", 1994 International Workshop on Low-power Design, April 1994, pp. 159-164.
- [Swanson72] R. Swanson and J. Meindl, "Ion-implanted Complementary MOS transistors in low-voltage circuits", IEEE Journal of Solid-state Circuits, vol. SC-7, pp. 146-153, Apr. 1972.
- [Sze81] S. Sze, *Physics of Semiconductor Devices*, John Wiley & Sons, 1981.
- [Tiwari93] V. Tiwari, P. Ashar, S. Malik, "Technology Mapping for Low Power", pp. 74-79, proceedings
-

-
- of the 1993 Design Automation conference.
- [Trickey87] H. Trickey, "Flamel: A high-Level Hardware Compiler", IEEE Transaction on CAD, Vol. 6, No. 2, pp. 259-269, 1987.
- [Tsui93] C. Tsui, M. Pedram, A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation", pp.68-73, proceedings of the 1993 Design Automation conference.
- [Uyemura92] J. Uyemura, *Circuit Design for CMOS VLSI*, Kluwer Academic Publisher, 1992.
- [VanOostende93] P. Van Oostende, P. Six, J. Vandewalle, and H. De Man, "Estimation of Typical Power of Synchronous CMOS Circuits Using a Hierarchy of Simulators," IEEE Journal of Solid-State Circuits, vol. 28, no. 1, Jan. 1993.
- [Veendrick84] H.J.M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits", IEEE Journal of Solid-State Circuits, Vol. SC-19, pp. 468-473, August 1984.
- [Walker89] R.A. Walker, D.E. Thomas, "Behavioral Transformation for Algorithmic Level IC Design" IEEE Trans. on CAD, Vol 8, No.10, pp. 1115-1127, 1989.
- [Walker91] R.A. Walker, R. Camposano: "A Survey of High-Level synthesis Systems", Kluwer, MA, 1991.
- [Watts89] R.K. Watts (ed.), *Submicron Integrated Circuits*, John Wiley & Sons, NY, 1989.
- [Weiser93] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, Vol. 36, No. 7, pp. 75-85, July 1993.
- [Weste88] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, MA, 1988.
- [Yano90] K. Yano, et al., "A 3.8ns CMOS 16x16 Multiplier Using Complementary Pass Transistor Logic", IEEE Journal of Solid-State Circuits, pp. 388-395, April 1990.
- [Yuan82] C.P. Yuan and T.N. Trick, "A Simple Formula for the Estimation of the Capacitance of Two-dimensional Interconnects in VLSI Circuits", IEEE Electron Device Letters, vol. EDL-3, pp. 391-393, Dec. 1982.
- [Yuan89] J. Yuan and C. Svensson, "High-Speed CMOS Circuit Technique", IEEE Journal of Solid-state Circuits, pp. 62-70, February 1989.
- [Zeger90] K. Zeger and A. Gersho, "Pseudo-Gray Coding," IEEE Trans. on Communications, COM-38, No. 12, pp. 2147-2158, December 1990.
-

Appendix A: Proof of NP-Completeness

Retiming for optimizing critical path can be solved optimally in polynomial time using the Leiserson-Saxe algorithm. However, retiming for power minimization is an NP-complete problem and therefore it is necessary to use heuristic approaches to find a “good” solution in a reasonable amount of time.

We will now prove that retiming for power minimization is an NP-complete problem. More precisely, it will be proven that a highly restricted version in which the only degree of freedom available during optimization is the change in the effective interconnect capacitance is NP-complete. Other more common and complex versions, which take into account the other effects associated with retiming for power minimization such as modifications to the supply voltage, are also NP-complete. The more general versions can be proven without any difficulty by restricting them to the restricted version mentioned above [Garey79].

The proof is presented using a classical component design technique [Garey79] and has three phases:

-
- Retiming for power minimization is formulated as decision problem, i.e. as the problem where the answer has only two options (i) yes and (ii) no.
 - The statement about the fact that the new problem is in NP is established by observing the simple fact that a proposed solution can be checked in polynomial time.
 - A polynomial transformation of an another NP-complete problem (MAX 3-CUT) is presented based on component design technique.

PROBLEM: RETIMING FOR POWER MINIMIZATION:

INSTANCE: Signal flow graph $SFG = (SV, SG)$, available time T , cost of power consumption for each execution unit, registers, memory, and interconnect, positive number P .

QUESTION: Is there a retiming for power minimization of the SFG such that its implementation is at most power consumption P ?

It is easy to see that this problem is in the class of NP problems since a feasible schedule which provides implementation on a retimed graph with the cost P , if exists, can be exhibited and checked for feasibility and cost quite easily.

We shall now polynomially transform the $MAX\ 3-CUT$ problem to the retiming for power optimization problem and finish the NP-completeness proof [Garey79] [Karp72].

PROBLEM: MAX 3-CUT [Garey79]:

INSTANCE: Graph $G = (V, E)$, all edges have equal unit weight, positive number K .

QUESTION: Is there a partition of V into disjoint sets V_1, V_2, V_3 such that the sum of weights of the edges from E that have its endpoints in different sets is at least K ?

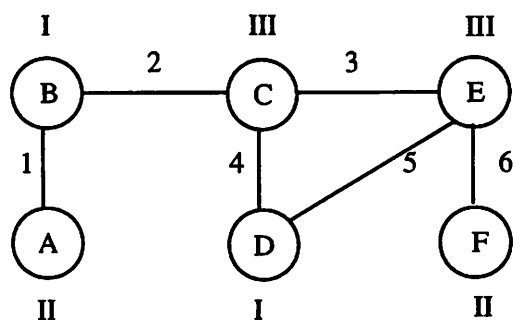
Suppose we are given an arbitrary graph G . We will first polynomially transform G to a corresponding SFG so that finding a solution in polynomial time to the retiming for power

minimization problem implies a polynomial time solution to the *MAX 3-CUT* problem.

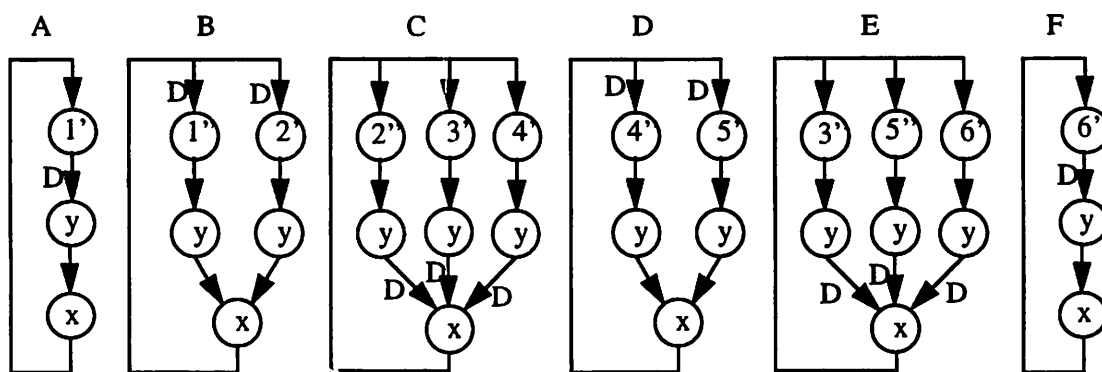
For each node V in G , SFG contains a disjoint subgraph, which contains as many cycles as the node V has incident edges. An illustration is shown in Figure 11-1. For example, subgraph B in SFG contains 2 cycles, corresponding to the 2 incident edges (1 and 2) to node B in G . Each edge i in graph G corresponds to two distinct cycles in sub-graphs which correspond to nodes to which edge i is incident to. We will denote those nodes in SFG with i' and i'' . Cycles with nodes i' and i'' , corresponding to edge i in G , contains three nodes of different types i' or i'' , y , and x . Node x is part of every cycle in the subgraph and acts as enforcer [Garey79]. Each operation takes one control cycle, and the available time is 3 control cycles. Nodes i' and i'' are sending their output results to nodes X' and X'' respectively. Nodes X' and X'' are sending their result to the same node X''' . Nodes X' , X'' and X''' are all of the same type.

Using appropriate structure for the rest of SFG it is easy to enforce the following situation. For execution of nodes X' , X'' and X''' we have the choice of two hardware execution units 1 and 2, and it is possible to execute each of them on either alternative. There is no possibilities for the reduction of voltage by reducing the time of implementation, and all interconnect paths are fixed, with the exception of transfers to nodes X' , X'' and X''' . For example, the enforcer node x enforces all delays in a subgraph to be at the same level. Otherwise, no feasible schedule exists.

Denote the power needed to transfer data between units 1 and 2 by P_{12} . Also let P_t denote the power required for all operations (including memory access, interconnect, control, and clock distribution) when all pairs of operations X' and X'' are executed on different functional units. Notice that we can avoid transfer of data between nodes 1 and 2 only under the condition when both X' and X'' are assigned to the same physical unit, either 1 or 2, since, under this condition, we can also assign operation X''' to this unit and avoid power necessary for transfer of data between nodes 1 and 2. However, we can assign nodes X' and X'' to the same unit only if operations i and i' can be scheduled in the different control steps due to retiming of subgraphs.



Max 3-CUT problem.



Also all i' and i'' ($i = 1, \dots, 6$) are connected in the following fashion

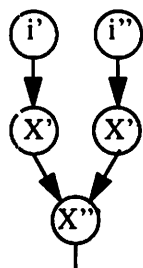


Figure 11-1 : Retiming for power minimization.

The problem has been constructed in such a way that in order that two operations i' and i'' , can send data to the same functional unit, they have to be scheduled in different control steps; i.e. the delays in their subgraphs have to be positioned at different levels. For instance, the delays in subgraphs A and B have been placed at different levels. Therefore, nodes i' and i'' from both subgraphs can send its output results to say node 1 in cycles 1 (subgraph B) and 3 (subgraph A). Putting delays on different levels corresponds to putting nodes, incident to corresponding edge in G , in different disjoint sets.

A retiming where all nodes i and i' has to be scheduled in the same control cycle has the power consumption of P_t . This corresponds to a 0 cut for the *MAX 3-CUT* problem since all nodes are in the same set. Moving one delay will reduce the power cost of the implementation by P_{12} and increase cutset value by the same amount. Therefore, if we can achieve the retiming for power minimization solution with the cost $P = P_t - K * P_{12}$, this corresponds to the solution of the *MAX 3-CUT* problem with cost K . Consequently, we have demonstrated that the *MAX 3-CUT* polynomially transforms to retiming for power optimization.
